NTNU – Trondheim
Norwegian University of
Science and Technology

Exchange of Security Incident Information in the context of Cloud Services

Christian Frøystad

Master of Science in Computer Science

Submission date:	June 2015
Supervisor:	Lillian Røstad, IDI
Co-supervisor:	Martin Gilje Jaatun, SINTEF IKT
	Erlend Andreas Gjære, SINTEF IKT

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

There has been done a lot of research on incident response in general, but not in the context of cloud computing. Grobauer and Schreck [1] has called for more research in this area, pointing out problems including, but not limited to; lack of access to event sources and vulnerability information, insufficient interfaces for access to relevant data, misdirection of incident reports, limited knowledge about architecture, missing knowledge of relevant data sources, and unclear responsibilities.

In this project, the student will study exchange of security incident information in the cloud, create a prototype of a solution, and examine potentially related non-technical challenges to exchange and sharing of security incident information.

[1] Grobauer, B and Schreck, T: Towards Incident Handling in the Cloud: Challenges and Approaches, CCSW '10 Proceedings of the 2010 ACM workshop on Cloud computing security workshop, New York, 2010

Formal supervisor: Lillian Røstad
Co-supervisor: Erlend Andreas Gjære
Co-supervisor: Martin Gilje Jaatun

Summary

Background In recent years, the use of cloud computing has increased significantly. More and more organizations are moving their services to the cloud as there are rather compelling benefits from using cloud computing. Some of these include reduced costs, better agility, and improved reliability. Less attention has been paid to the lack of solutions to well known incident handling problems. While some research have been published in the later years related to security incidents in the cloud, much of this have been focused around digital forensics rather than a practical approach to exchanging security incident information. Sharing security incident information is becoming increasingly important, as attacks becomes more sophisticated, widespread and frequent. Additionally, new laws place new requirements on cloud service providers with regard to notification of both end users and competent authorities. To further complicate the matter Cloud Service Providers (CSPs) use services from other CSPs, creating Cloud Provider Chains. This means that a CSP used by an end user or another CSP could rely on any number of CSPs in a simple chain or a more complex network.

Method Literature study and interviews from the prestudy, and a literature study on incident formats were used to create a specification for an incident exchange interface as well as an incident representation format. Based on this interface and format, a prototype was constructed to act as a catalyst during interviews about incident sharing. Focused interviews, catalyzed by the prototype and a scenario, were used to validate the interface and the format. The approach was also validated against literature.

Results The interviews, as well as the validation against literature, indicates the importance of the solution being agnostic to the tool used in the organization. The usefulness of being able to exchange incident information was confirmed. The prototype presented to the participants did not fit well with their current workflow, but the interface and format is integrable with current tools and workflows. There are challenges related to both legal matters and public relations to take into consideration when sharing incident information.

Conclusion This thesis shows the need for exchanging security incident information and contributes toward achieving this goal by suggesting a way to do this by means of an interface and an incident format. A subscription-based interface between a provider and a subscriber would allow the provider to push security incident information to the subscriber in a timely manner, while still leaving the provider in control of which information is shared and when it is shared.

The most prominent non-technical challenge is one of trust. It is expected that this, at least to some degree, can be accomplished by means of contracts and SLAs. Sharing of security incident information is further complicated by privacy laws, placing restrictions on the information that might be shared. Public perception of a company is another challenge. Sharing of security incident information in a professional way might improve the image of a company, while unprofessional sharing might harm it. Adoption is a major challenge as the solution will only be useful when in use. This thesis has identified three drivers for adoption of such a solution: reduced costs, increased revenue and legal requirements.

Email are currently heavily used in organizations for exchange of security incident information. The solution presented in this thesis could replace email as the channel of incident information exchange without much impact on how the incident handlers work. The interviews incident handlers did point out that if the underlying interface and format were integrated into their current tooling, it would be a useful solution.

Sammendrag

Bakgrunn I de senere år har bruken av nettskyen økt markant. Flere og flere organisasjoner flytter sine tjenester ut i nettskyen da bruk av denne tilbyr en del appellerende fordeler. Noen av disse er reduserte kostnader, større smidighet og forbedret pålitelighet. Det er viet mindre oppmerksomhet til mangelen på løsninger i tilknytning til velkjente problemer rundt hendelseshåndtering. Til tross for at noe forskning, relatert til sikkerhetshendelser i nettskyen, er publisert de senere år, har denne hovedsakelig vært fokusert rundt tekniske bevis i relasjon til straffeforfølgelse [digital forensics] heller enn en praktisk tilnærming til å dele informasjon om sikkerhetshendelser. Deling av informasjon relatert til sikkerhetshendelser har økt viktighet ettersom angrepene blir mer sofistikerte, utbredte og hyppige. I tillegg stiller nye lover nye krav til tjenesteleverandører når det kommer til å varsle både sluttbruker og myndigheter om sikkerhetshendelser. En ytterligere komplikasjon er at tjenesteleverandører kan benytte tjenester fra andre tjenesteleverandører og på den måten danne en kjede av leverandører. Det betyr at en en tjenesteleverandør, som leverer tjenester til en sluttbruker eller en annen leverandør, kan være avhengig av et vilkårlig antall andre leverandører i en enkel kjede eller et komplekst nettverk.

Metode Litteraturstudie og intervju fra forstudiet, sammen med et litteraturstudie om hendelsesformat ble brukt til å lage en spesifikasjon av et grensesnitt for utveksling av hendelser og et format for representasjon av hendelser. Basert på dette grensesnittet og formatet, ble det utviklet en prototype til bruk som katalysator i intervjuer om hendelsesdeling. Fokuserte intervjuer, katalysert av prototypen og senarier, ble brukt for å validere grensesnittet og formatet, representert ved prototypen. Tilnærmingen ble også validert mot litteratur.

Resultat Intervjuene, sammen med valideringen mot litteratur, indikerer viktigheten av at løsningen er agnostisk til verktøyene som benyttes i organisasjonen. Nyttigheten av å kunne utveksle hendelsesinformasjon ble bekreftet. Prototypen passet ikke inn i deltagernes arbeidsflyt, men grensesnittet og formatet kan integreres med deres nåværende verktøy og arbeidsflyt. Det er både juridiske og markedsmessige utfordringer å ta hensyn til i forbindelse med deling av hendelsesinformasjon.

Konklusjon Denne oppgaven viser behovet for utveksling av hendelsesinformasjon og er et bidrag til å oppnå dette målet, ved å foreslå en mte dette kan gjres å benytte et grensesnitt og et hendelsesformat. Et abonnementsbasert grensesnitt mellom leverandør og abonnent gir leverandøren mulighet til å sende hendelsesinformasjon til abonnenten innen rimelig tid, samtidig som han selv vil være i kontroll over hvilken informasjon som deles med hvem og når den deles.

Den mest fremstående ikke-tekniske utfordringen er tillit. Det forventes at tillit, i alle fall i noen grad, kan oppnås ved hjelp av kontrakter og tjenestenivåavtaler. Deling av hendelsesinformasjon kompliseres videre av personvernslovgivning som legger restriksjoner på hvilken informasjon som kan deles. Hvordan et selskap fremstår for offentligheten er en annen utfordring. Deling av hendelsesinformasjon på en profesjonell måte kan forbedre selskapets fremtoning, mens uprofesjonell deling av informasjon kan skadet den. Innføring av løsningen er også en utfordring siden den bare vil være nyttig om den blir brukt. Denne oppgaven har identifisert tre momenter som kan bidra til at løsningen blir tatt i bruk: reduserte kostnader, økt inntjening, og juridiske krav.

Epost er for tiden aktivt brukt i organisasjoner for deling av hendelsesinformasjon. Løsningen som er presentert her, kan erstatte epost som kanal for deling av hendelsesinformasjon uten særlig endring i hvordan hendelseshåndterere arbeider. Hendelseshndtererne som ble interbjuet påpekte at om grensesnittet og formatet ble integrert inn i deres nåværende systemer, ville løsningen være nyttig.

Preface

This thesis, subject code TDT 4900, was conducted at the Department of Computer and Information Science, at the Norwegian University of Science and Technology (NTNU), during the spring of 2015. The formal supervisor from NTNU was Lillian Røstad, while the actual supervision was done by two co-supervisors from SINTEF ICT: Erlend Andreas Gjære and Martin Gilje Jaatun.

Given the size, complexity and multidisciplinary nature of the problem at hand, it has not been possible to give the definitive answer on how to solve incident sharing across organizations. A technical approach is suggested and non-technical challenges are discussed. The solution presented has also been adopted by the A4Cloud project, as mentioned in appendix B.

I should thank my formal supervisor, Lillian Røstad, for approving this task, so that I've been able to work on such an interesting subject. More so, I'd like to thank the co-supervisors, Erlend Andreas Gjære and Martin Gilje Jaatun, for spending time on supervising this project and guiding me through the process of completing this thesis. Special thanks goes to Martin for nice drawings of clouds on the whiteboard.

Further thanks goes to Karin Bernsmed for her feedback on how to approach the interview results, as well as her participation as the transcriber during the interviews. I'd also like to thank the rest of the Information Security group at SINTEF ICT for hints and tricks, as well as educational lunch conversations. Though not mentioned by name, I'd like to thank the participants in the interviews for their valuable contribution to this thesis. Alfredo Reyes receives a large thank you for proofreading this document, both for grammatical and logical errors. Finally I'd like to express my gratitude to SINTEF ICT for providing me with an office, giving me the best possible work conditions.

Table of Contents

List of Tables

List of Figures

Acronyms

API Application Programming Interface. 11, 12, 25, 63

AWS Amazon Web Services. 11

CAPEC Common Attack Pattern Enumeration and Classification. 19

CERT Computer Emergency Response Team. 8, 63, 64, 88, 89

CSIRT Computer Security Incident Response Team. 12, 83, 85

CSP Cloud Service Provider. i, 1, 2, 66, 82–84, 86, 87

CybOX Cyber Observable eXpression. 19, 36, 74

DSRP Design Science Research Process. 3

FIDO Fully Integrated Defense Operation. 91, 92

HMAC Hash-based Message Authentication Code. 30, 57

IDMEF The Intrusion Detection Message Exchange Format. 18, 36

IODEF Incident Object Description Exchange Format. 12, 18, 24, 33, 36, 62, 73, 74, 90, 95

ISP Internet Service Provider. 69

JSON JavaScript Object Notation. 62, 82

MAEC Malware Attribute Enumeration and Characterization. 19

MISP Malware Information Sharing Platform. 90

NIST National Institute of Standards and Technology. 19

PII personally identifiable information. 83

REST Representational State Transfer. 11, 14

RID Real-time Inter-network Defense. 90

SaaS Software as a Service. 11

SLA Service Level Agreement. 36, 77, 84, 87, 88, 90, 92, 95

SSL Secure Socket Layer. 93

STIX Structured Threat Information eXpression. 19, 24, 36, 62, 74, 89–91, 95

TAXII Trusted Automated eXchange of Indicator Information. 89, 91, 93, 96

TLP Traffic Light Protocol. 68, 69, 84, 93

UI user interface. 43, 53, 55, 57

URI Uniform Resource Identifier. 30, 58

UUID Universally Unique Identifier. 26–30, 34

UX user experience. 51, 52

XML Extensible Markup Language. 18, 19

Chapter 1

Introduction

1.1 The need for Security Incident Information Exchange

In recent years, the use of cloud computing has increased significantly. Amazon.com (2015) claims that their cloud service offering is a "*$5 billion business and still growing fast - in fact it's accelerating*". According to Synergy Research Group (2015), the year-over-year growth for the cloud infrastructure services vary from 37 % for Saleforce, to 96 % for Microsoft. It is therefore natural to conclude that more and more organizations are moving their services to the cloud. Microsoft is even planning on moving all their services and infrastructure onto the cloud (Microsoft IT, 2014). Some CSPs use services from other CSPs, creating Cloud Provider Chains. This means that a CSP used by an end user or another CSP could rely on any number of CSPs in a simple chain or a more complex network.

As pointed out by Kalloniatis et al. (2014), there are rather compelling benefits from using cloud computing. Some of these include reduced costs, better agility, and improved reliability. Grobauer and Schreck (2010) describe another side of cloud computing: the lack of solutions to well known incident handling problems. The authors call for more research in several areas, including, but not limited to, no access to CSP controlled event sources and vulnerability information, insufficient interfaces for access to relevant data, inability to add security-specific event sources, and limited knowledge of relevant data sources. While some research has been published in the later years related to security and incidents in the cloud, much of this has been focused around digital forensics rather than a practical approach to exchanging security incident information.

Sharing security incident information is becoming increasingly important, as the attacks are becoming more sophisticated, widespread and frequent.

> "*Our adversaries are amazingly coordinated. They do a far better job sharing information than we do. Its becoming clear that the good guys need to find ways to share actionable information in real time to counter this threat.*" (Horne, 2014)

Another reason for having an established system for exchanging security incident information, is the introduction of the new General Data Protection Regulation in the European Union. This new regulation, to be enforced from 2017, requires notification of a Supervisory Authority as soon as an organization becomes aware of a data breach related to personal information and notification of the end user within defined time frames. There are severe penalties for not complying with these terms (The European Parliament and the Council of the European Union, 2013).

Timely and effective exchange of security incident information could lead to improved security, as relevant information could propagate faster and more information could be utilized in identifying and mitigating any incidents. An incident tagged as uncritical by one CSP, could potentially be critical for another CSP that relies on the first. More severely, it is also possible that the incident would never be detected by the latter, if not being notified. This would not necessarily be caused by incompetence or malpractice, but could be due to lack of access to the necessary information.

1.2 Research Questions

The preceding section presents several challenges in relation to the cloud and incident response, particularly the lack of access to the relevant information is prominent. This thesis is a contribution towards solving this problem, which is approach by examining the following questions:

RQ-1 How to provide the relevant information about an incident in a cloud service to all involved parties in a timely manner through an exchange interface?

RQ-2 Which non-technical challenges arises and how could they be mitigated?

RQ-3 How well does the proposed approach fit into current practice?

1.3 Structure

This first chapter has introduced the need for the project, as well as stated which research questions are to be examined. Chapter 2 presents the methods used to answer the posed research questions. Chapter 3 introduces some more background material on exchanging incident information, as well as going into different incident representations. Chapter 4 presents an interface to be used for exchanging security incident information. Chapter 5 presents an incident representation format based on the findings in chapter 3. A prototype, based on the format and interface, is presented in chapter 6. In chapter 7, results from evaluating the approach and prototype with experienced incident handlers is presented. Chapter 8 contains discussion surrounding the interface, format and the prototype, as well as comparison to earlier approaches in literature and industry, and further work. Finally, the thesis is concluded in chapter 9.

Chapter 2

Method

2.1 Design Science Research

For the sake of reducing the scope of the problem, it was assumed that it is possible for organizations to trust each other at the same level as human trust each other. This assumption is briefly touched upon in the discussion.

The overall strategy for conducting the research in this thesis follows the Design Science Research Process (DSRP). The process for Design Science Research, as outlined by Peffers et al. (2006), is a research model containing six steps. Figure 2.1 shows how the steps of the process relate to each other in general, and figure 2.2 shows the process instantiated for this particular project.

2.1.1 Problem identification and motivation

The first step of DSRP, as described by Peffers et al. (2006), concerns identifying the problem to be solved and the motivation for solving it, as well as the potential value of a solution. The problem and its motivation is discussed by Grobauer and Schreck (2010) and further explored by Frøystad (2014). The problem is the lack of relevant information in handling computer security incidents, and the motivation and expected value of a solution is enhanced security as a result of access to the right information within reasonable time.

2.1.2 Objectives of a solution

The DSRP, by Peffers et al. (2006), goes on to describe the need for objectives for the solution. The objectives need to be related to the problem described, as well as be a result of knowing the state of the problem and current solutions.

The objectives of the solution was identified by Frøystad (2014), by means of literature study and interviews with subject matter experts. This resulted in a qualitative objective of a *simple, deterministic interface allowing for exchange of security incident information across organizations and clouds*.

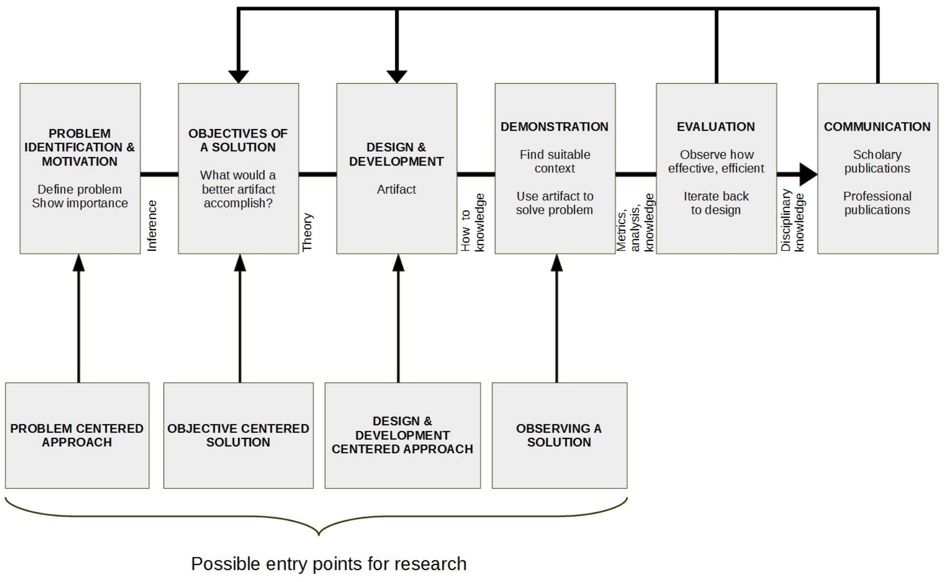

PROBLEM IDENTIFICATION & MOTIVATION
Define problem
Show importance
Inference
OBJECTIVES OF A SOLUTION
What would a better artifact accomplish?
Theory
DESIGN & DEVELOPMENT
Artifact
How to knowledge
DEMONSTRATION
Find suitable context
Use artifact to solve problem
Metrics, analysis, knowledge
EVALUATION
Observe how effective, efficient
Iterate back to design
Disciplinary knowledge
COMMUNICATION
Scholary publications
Professional publications
PROBLEM CENTERED APPROACH
OBJECTIVE CENTERED SOLUTION
DESIGN & DEVELOPMENT CENTERED APPROACH
OBSERVING A SOLUTION
Possible entry points for research

Figure 2.1: The Design Science Research Process (Peffers et al., 2006)

2.1.3 Design and development

Based on the literature study and the interviews conducted by Frøystad (2014), two specifications was created. One for the interface between two exchange parties, and one for a simple incident format to be used when exchanging incident information.

After a preliminary version of the specifications were complete, a prototype was constructed. During development of the prototype, problems with the specifications were discovered and addressed by evaluating the severity of the problem and the impact a solution would have on the specification. When the impact on the specification was low, the solution was applied instantly. If a problem which would result in a solution with high impact on the specification had occurred, a napkin analysis on the reduced quality of specification versus improved implementation would have been conducted. The improvements for the implementation would need to be significant in order to justify reducing the quality of the specification.

2.1.4 Demonstration

Peffers et al. (2006) describe step four as "*Demonstrate the efficacy of the artifact to solve the problem.*" Demonstration was done by having experienced incident handlers participate in a scenario based test of the prototype, and using this as way of interviewing the incident handler about the usefulness of the solution and how it could be further improved. Section 2.2 describes in more detail how the interviews were conducted.

2.1.5 Evaluation

Evaluation was done in two different ways:

- Against literature
- Analyzing results from scenario based test

The evaluation against literature used the prestudy by Frøystad (2014) to identify the relevant literature. Evaluation was also done against the interviews conducted in the same prestudy. This was done by going through each interview and each significant element from papers, to see how the chosen approach fits into that case or those criteria. This is done to examine the validity and novelty of the approach, but also to gain knowledge for further work on the problem and the direction in which it should be headed. An evaluation of which of the problems presented by Grobauer and Schreck (2010) the specifications solve was also done. Finally, the result of the scenario-based test with experienced incident handlers was analyzed. This was used both to evaluate the approach as well as define further work in order to maximize the usefulness of the specifications.

2.1.6 Communication

Communication of the problem and the suggested solution is done through this thesis and the paper in appendix A, which has been submitted to CloudCom 2015.

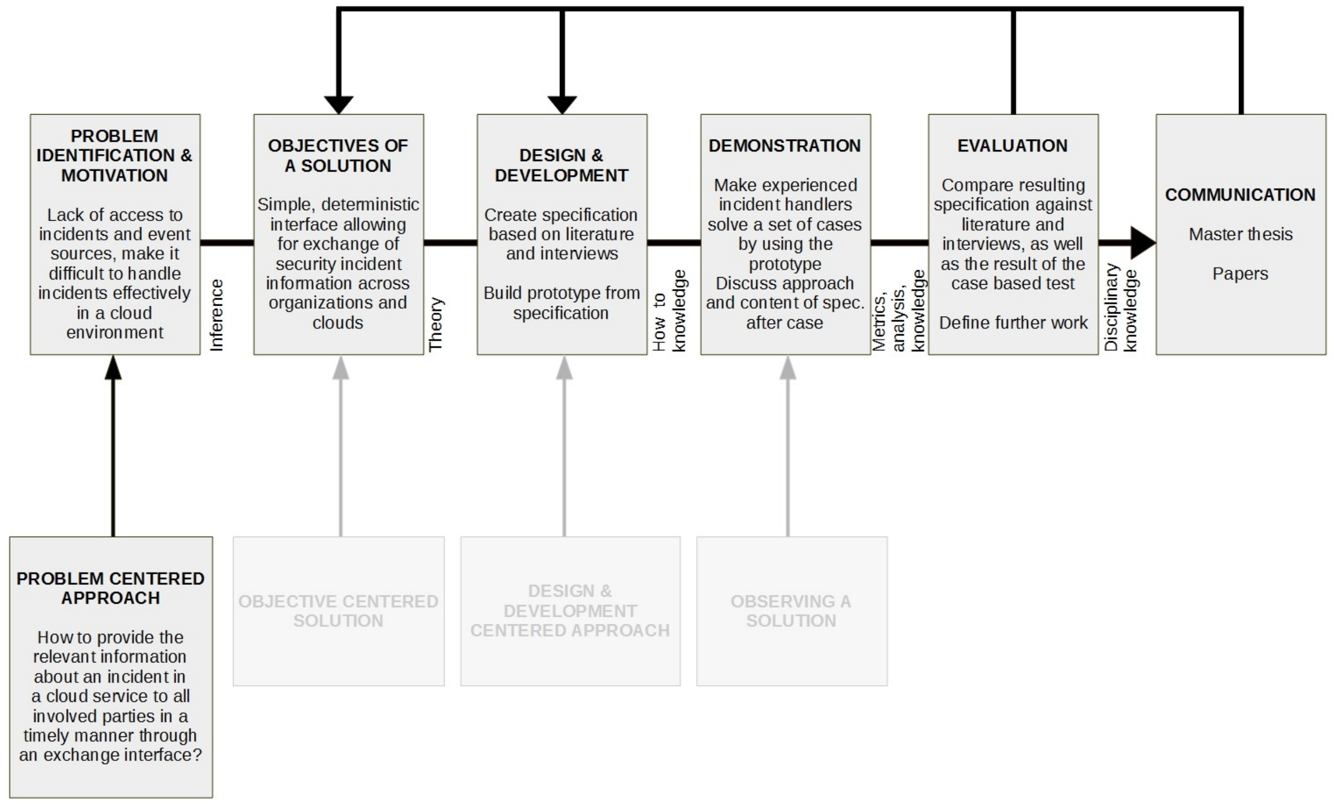

PROBLEM IDENTIFICATION & MOTIVATION
Lack of access to incidents and event sources, make it difficult to handle incidents effectively in a cloud environment
Inference
OBJECTIVES OF A SOLUTION
Simple, deterministic interface allowing for exchange of security incident information across organizations and clouds
Theory
DESIGN & DEVELOPMENT
Create specification based on literature and interviews
Build prototype from specification
How to knowledge
DEMONSTRATION
Make experienced incident handlers solve a set of cases by using the prototype
Discuss approach and content of spec. after case
Metrics, analysis, knowledge
EVALUATION
Compare resulting specification against literature and interviews, as well as the result of the case based test
Define further work
Disciplinary knowledge
COMMUNICATION
Master thesis
Papers
PROBLEM CENTERED APPROACH
How to provide the relevant information about an incident in a cloud service to all involved parties in a timely manner through an exchange interface?
OBJECTIVE CENTERED SOLUTION
DESIGN & DEVELOPMENT CENTERED APPROACH
OBSERVING A SOLUTION

Figure 2.2: The instantiated Design Science Research Process for this project

2.2 Focused Interview

Focused interviews were conducted to test the hypothesis that was the specification for incident format and exchange interface, represented by the prototype. Merton and Kendall (1946) describe what they title a *focused interview* and outline the characteristics of such an interview as follows:

1. "*Persons interviewed are known to have been involved in a particular concrete situation*"
   The interviewees have been involved in incident handling for several years and, at the time of the focused interview, have just finished a set of cases that emulate incident handling using the prototype. The prototype was used to catalyze the interview, by both showing the participant the functionality of the prototype and make him solve a set of defined scenarios relevant to incident handling. The incidents or cases used, can be found in appendix D. During this session, the participant was encouraged to think out loud, and the interviewer would ask additional questions to gain more insight into the participants thoughts and views on a specific subject. For the list of questions made to assist the interviewer, refer to appendix C.1. The interviews were conducted in a semi-structured manner as described by Oates (2006).

   This differs from a usability test as what is tested is neither the user's ability to use the prototype nor the usability of the prototype, but the scenario was rather used to gain in depth information about how the proposed solution would work in a somewhat realistic setting, while at the same time drive the conversation. By presenting the participant with concrete situations and a concrete implementation, it was easier to gain concrete and valuable answers rather than just general thoughts about the subject and the solution.

2. "*The hypothetically significant elements, patterns, and total structure of this situation have been previously analyzed by the investigator*"
   During the prestudy by Frøystad (2014) and the work on this thesis, the situation and the problem has been analyzed and a hypothesis for a solution has been proposed

3. "*Interview guide*"
   An interview guide was constructed, and can be seen in appendix C.2

4. "*The interview itself is focused on the subjective experiences of persons exposed to the preanalyzed situation*"
   The interview guide is constructed to ask for the participants answers and experience, rather than asking him to present the views of others

By catalyzing the focused interview with a practical session, the participant has already been exposed to the prototype and the concept. This was used to facilitate a shorter and more focused talk about the subject at hand, which resulted in more effective interviews - allowing for the total session to take just over an hour per participant.

2.2.1 Participants

The participants in the focused interview are anonymized and while they are both referred to as *he*, they are not necessarily male.

Participant A
Participant A is part of a Computer Emergency Response Team (CERT) in an organization responsible for delivering infrastructure and services to an entire sector. He has more than ten years of experience from incident handling, and heads the virtually composed CERT.

Participant B
Participant B is part of a large CERT responsible for infrastructure and information for a large sector. He has several years of experience with incident handling and information sharing.

2.2.2 Execution

The interview were conducted in a meeting room at the participant's venue. The participant was given access to a computer, which was connected to a larger screen allowing everyone in the room to see his interaction with the system. Four persons were active during the interview: the participant, the interviewer, the transcriber and the tool operator. The participant was the incident handler being interviewed. The interviewer directed the interview, posed the questions and provided the participant with the proper context, as well as answering any questions the participant had. The transcriber transcribed the interview by using the interview guide (Appendix C) as a template. The tool operator emulated different cloud services and sent incidents to the participant upon the interviewer's go-ahead. All participants signed a consent form.

The session started with the interviewer introducing the project and the people in the room, as well as stressing that the participant was not being tested. He went on to explain that the participant's knowledge about and experience from incident handling made his views on the solution was interesting. The participant was encouraged to think out loud, and to ask questions if anything was unclear. Some time was spent introducing the participant to the prototype, allowing him to browse around as he pleased. After the participant confirmed that he was ready to start, the scenarios from appendix D were started. For each scenario, the participant told about his initial reaction to the incident and how he would handle it. The interviewer posed questions as needed from the interview guide in appendix C. After the participant had completed the scenarios, the focused interview was conducted with the questions from appendix C.2 as a base.

2.2.3 Analysis

The transcribed versions of the interviews were studied thoroughly and main subjects of the interviews were identified. This information was structured into the three main subjects; the format, the interface and the prototype or workflow. For each subject, the answers and reflections of the participants are presented as purely as possible, and the author's analysis is placed in separate corresponding sections. The answers were applied to the specification in order to see how well it conformed and also to identify potential areas of improvements.

2.3 Literature review

The literature review continues where Frøystad (2014) left off, basing itself on the work done in that prestudy. During the prestudy, the focus of was mostly on the interface and the ideas around exchanging security incident information. To some degree, there was also a focus on incident formats. Table 2.1 shows the queries used in the prestudy and the number of results for each of the two search engines used.

Search Query	#hits Scopus	#hits Google Scholar
incident sharing	848	403 000
security incident sharing	195	193 000
incident management	128	1 290 000
grobauer cloud	0	372
cloud incident event sources	0	159 000
cloud incident event interfaces	0	24 600
bank fraud information sharing	1	59 600
incident sharing between companies	0	207 000
incident management in supply chain	0	108 000
incident handling supply chain	0	80 100

Table 2.1: Search queries and results for the prestudy (Frøystad, 2014)

Some additional papers were found by examining the references of already identified papers, some were suggested by researchers in the field, and then some were suggested by reviewers of paper sent to a conference.

For each paper found by search, the title and abstract was used to decide if the paper should be further investigated. If those were inconclusive, the introduction and conclusion sections were studied as well. This approach is recommended by Oates (2006, p. 85). Any paper found relevant was read at least once, and notes taken. Any paper suggested by researchers in the field was studied thoroughly, even though it was not always immediately apparent how it related to the task at hand. For information related to incident formats, a overview comparing the representable data was created. All this information was used in creating the specification for a solution.

Chapter 3

Background

During this thesis *cloud provider chain*, *cloud supply chain* and *cloud delivery chain* is used interchangeably. Figure 3.1 show a fairly typical, though simple, example of a cloud provider chain. The bottom left Software as a Service (SaaS) operate their services on the Azure platform, while relying on another SaaS for subscription management. The subscription SaaS does in turn rely on Stripe for payments, Heroku for its platform and Amazon Web Services (AWS) for its database. Heroku also rely on AWS for its infrastructure. A cloud provider chain is therefore a link of cloud providers that either relies on or pass on the services of a prior link.

AWS does not know how the customers of Heroku use the infrastructure AWS provides to Heroku. If proper security is in place, neither does Heroku to a full extent, especially in this particular case where the subscription SaaS has chosen to use AWS directly for the database. Furthermore, the organization operating the bottom left SaaS has limited insight into the preceding links in the cloud provider chain. It is possible that an incident occurring at AWS, would affect the bottom left SaaS - though he would probably not be notified.

3.1 Incident Exchange Concept

This document builds on the work done in an earlier project by Frøystad (2014). This section gives a short introduction to the concept identified during that project.

On the highest level, the solution is a set of interfaces and behaviors an implementation must conform to. More specifically, the solution is a Representational State Transfer (REST) Application Programming Interface (API) that could be implemented by every part of the cloud delivery chain, as well as a common exchange format. Combined, these would allow effortless and deterministic exchange of incident information between the different actors in the cloud supply chain. Figure 3.2 illustrates the conceptual simplicity of the solution. By acting as a formal channel between two parties, the solution allows for human incident handlers to exchange computer security incident information effortlessly as well as allowing parts of or the entire incident handling process to be automated. In

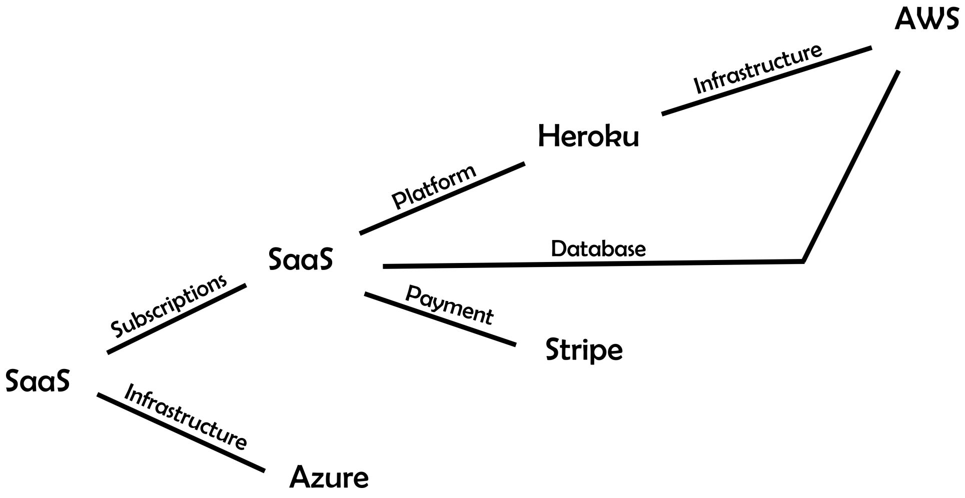


Figure 3.1: The figure shows an example of a Cloud Provider Chain. Items to the left of an edge depends on items at the right side of the edge.

addition, it is possible to guarantee that incident information is sent only to the intended recipient and with better security than what is offered in e.g. email. How the system behaves behind the endpoints, is left to be decided by those implementing the solution in their products or infrastructures. Behind the channel interface (the tin can in figure 3.2) the services are free to use and produce the incident information in any way they desire, or any way their customers demand. This will allow those wishing for automation to implement the solution in an automated manner, while those seeking to have humans controlling the entire information flow would still be able to do so by, e.g., using a help-desk system as a back-end.

Consumers of the API registers which systems, services, incidents, etc., they wish to receive alerts about. The provider of the API would have the final say in which information they are actually allowed to receive. By also allowing the consumers to specify a threshold for receiving incident information, e.g., only when of type *T* and severity *high*, network efficiency would be preserved while also avoiding drowning the consuming Computer Security Incident Response Team (CSIRT) with information they are not interested in.

The proposed solution is not limited to two different clouds connecting, but this simple building block allows for composition into complex networks of services, infrastructure and CSIRTs. The Incident Object Description Exchange Format (IODEF), a format for representing incidents discussed more in section 3.2.1, is capable of keeping track of where the incident occurred and allows for correlation of incidents. This avoids overwhelming the CSIRT with incidents already handled or that are currently being handled, even though the same incident report might arrive through different connections.

Figure 3.3 exemplifies the more complex network structure that might arise. Within *Cloud 1*, C depends on B which depends on A. In *Cloud 2*, E depends on D which depends on B and C from *Cloud 1*. In *Cloud 3*, F depends on E from *Cloud 2* and C from *Cloud 1*. This could result in F receiving incidents from A and B as many as three times for each incident.

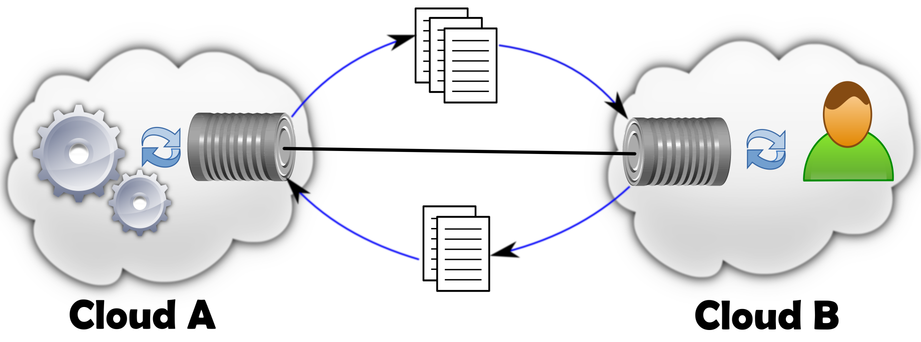


Figure 3.2: Illustration of high level incident exchange concept. (Frøystad, 2014)

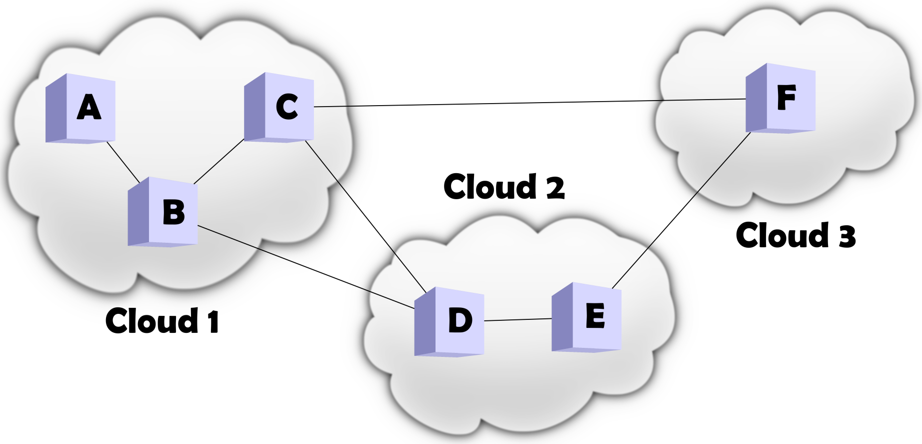


Figure 3.3: Example of more complex incident exchange network. The dependencies goes from left to right, so e.g. F depends on E and C. (Frøystad, 2014)

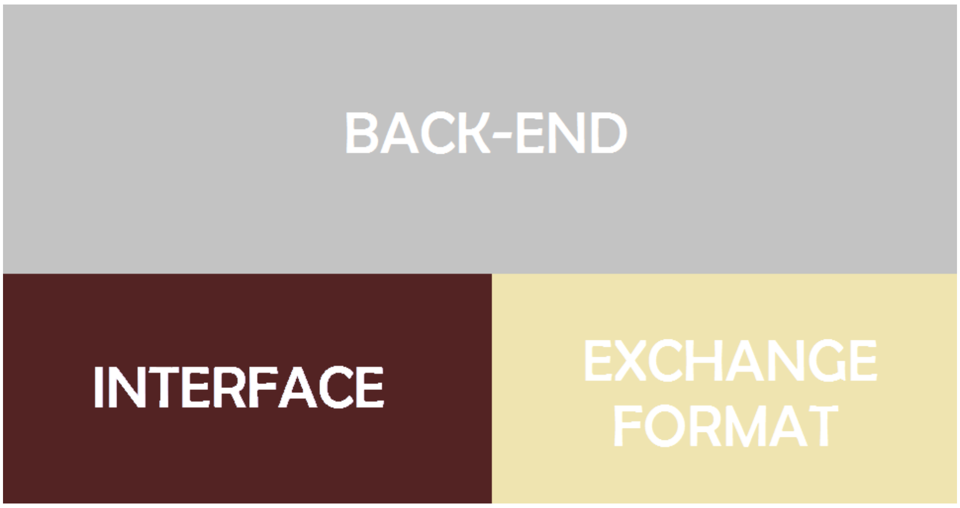


Figure 3.4: Development view of system showing the dependencies for developing the system. (Frøystad, 2014)

The three parts that constitutes the system is displayed in figure 3.4. The interface and the exchange format can be developed somewhat in parallel, while the back-end depends on both the interface and the exchange format being ready. The interface and the exchange format remains the same from provider to provider, while the back-end can differ vastly, as the provider is free to implement the format and the interface in any way suiting his use case.

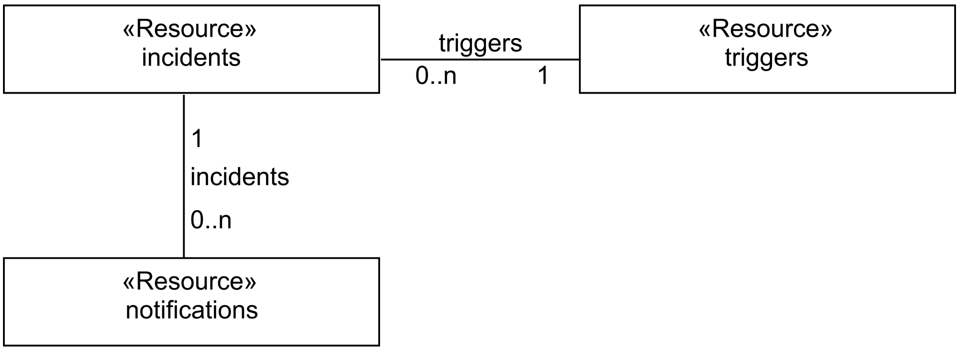


Figure 3.5: Logical view of system showing REST-API resources. (Frøystad, 2014)

Figure 3.5 shows the resources envisioned, with notifications allowing the subscriber to group incidents together, and triggers allowing the subscriber to decide under which circumstances he wants to be notified. The triggers have threshold values that result in notification if surpassed. Triggers can be combined using logical operators such as AND, OR and XOR.

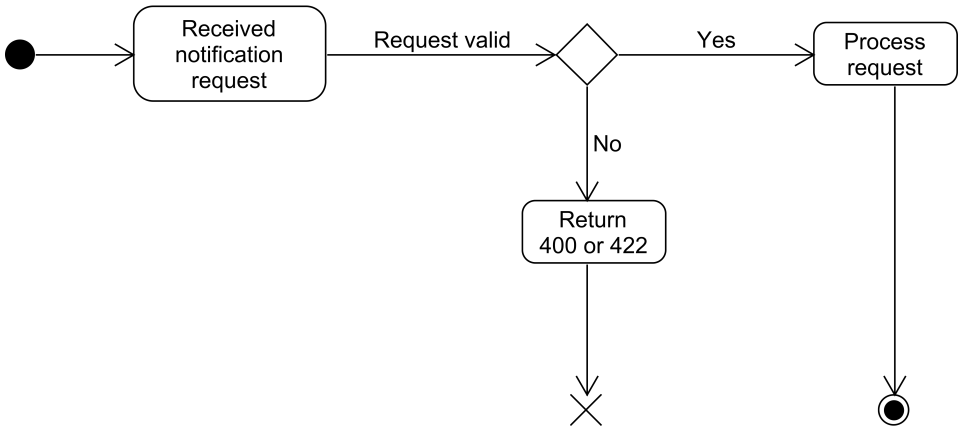


Figure 3.6: Process view of system handling a request for notifications from another system. (Frøystad, 2014)

Figure 3.6 shows how a request for receiving notifications is handled. The subscriber registers what he wants to be notified about, and sends the request to the provider. If the request is valid, the provider processes the request to validate that it is in accordance with the what the provider has decided to share with this particular customer or subscriber. If the request is invalid, the system returns 400 for bad request or 422 if the request is syntactically correct but semantically incorrect.

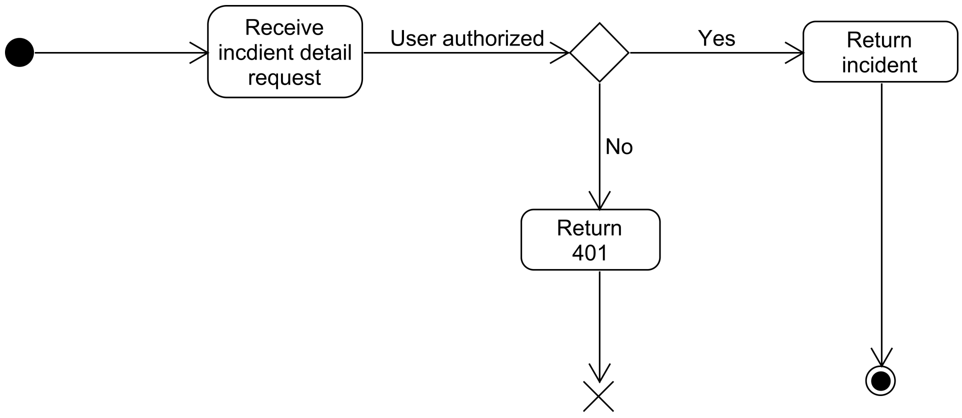


Figure 3.7: Process view of system handling a request for general incident details from another system. It is assumed that the request has passed through a validation step as in figure 3.6 (Frøystad, 2014)

The subscriber might also request details about an incident. Figure 3.7 shows the process involved. When the provider receives the request, the user is authenticated and the provider makes sure he has the right to gain access to what he is requesting, if he has not the necessary authorization a 401 *Unauthorized* status code is returned. Otherwise the requested incident is returned together with a 200 *OK* status code.

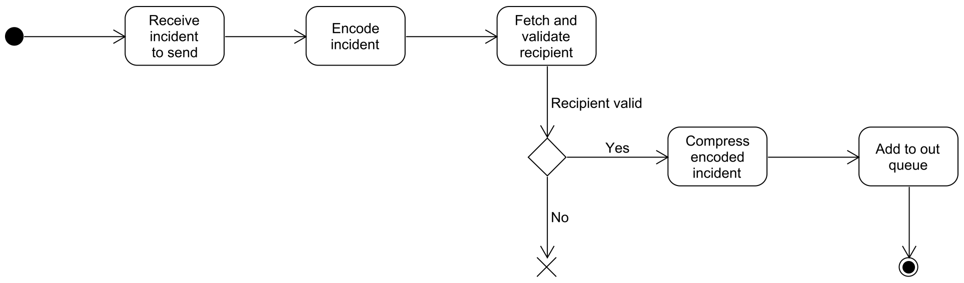


Figure 3.8: Process view of system sending incident information to another system. (Frøystad, 2014)

Figure 3.8 shows how the back-end sends a notification using the defined interface and a common exchange format. The back-end encodes the incident to be sent using a common format. Receivers are fetched from an internal store of subscribers that match the criteria for the incident at hand. Each recipient is validated, the encoded incident compressed and sent to every valid recipient by adding it to the outgoing queue. Invalid recipients do not receive the notification.

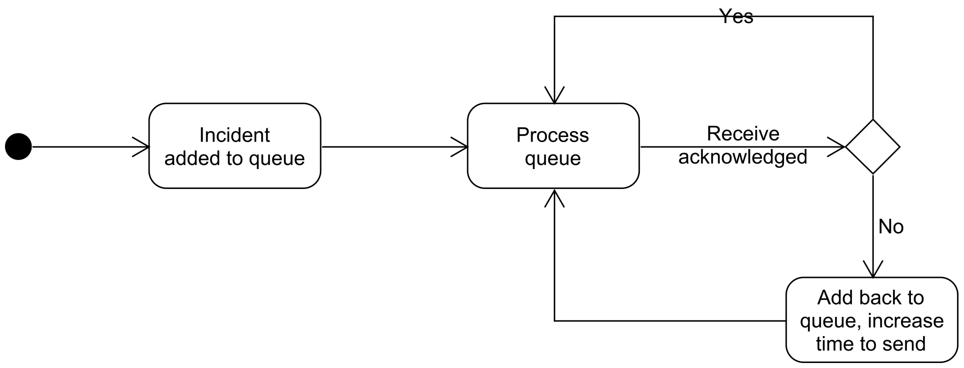


Figure 3.9: Process view of system incident notification queue. (Frøystad, 2014)

Figure 3.9 shows a basic concept of a sending queue. An incident is added to the queue, which is processed asynchronously from the main back-end process. The queue sends its content as early as possible, removing incidents on which it receives acknowledgment that is received and adding back to the queue incidents that are not acknowledged. For each time sending of an incident fails, the time until retry for that particular incident is increased, e.g., by multiplying the former time by 2.

3.2 Incident Representation

This section explores which information different standards, organizations, and research include when describing an incident.

Values	IODEF	STIX	NIST	US-CERT	SM	EU
ID	x	x				
Language	x					
Incident type		x	x			
Status		x	x			
Parent	x	x				
Impact	x	x	x			x
Summary	x	x	x			x
Description	x	x	x	x		x
Occurrence time	x	x		x		x
Detection time	x	x		x		x
Cause	x	x	x			
Source	x	x	x			
Mitigating actions	x	x	x	x		x
Liaison	x	x	x	x		
Vector	x	x	x	x		x
Port	x	#	x	x		
IP Address	x	#		x	x	
Event Log	x	#			x	
Domain	x	#			x	
Operating System	x	#		x	x	
Target	x	x			x	
Processes/Services	x	x			x	
Changed files	x				x	
Prioritizing factors	*		x	x		
General comments	*		x			
Other organizations contacted	*		x			
Incident handler comments	x		x			
List of evidence gathered	x		x			
Incident cost	x		x			
Business impact	x		x			
Contact information for all involved parties	x	x	x			
Mitigating factors	*	x		x		
System function	x	x		x		
Physical system location	x	x		x		
How incident was identified	*	x		x		
Proto Header	*				x	
File, Directory	*	x			x	

Values	IODEF	STIX	NIST	US-CERT	SM	EU
Content Strings	*				x	
Hive	*	#			x	
Key / Key Group	*	#			x	
Environment Variable	*	#			x	
Proto Field	*	#			x	
Session Token	*	#			x	
Number of notified users	*					x
Means of communication	*					x
Identification of provider	x	x				x
Type Data	*	x				x
Use of other providers	*	#				x
End time	x	x				

Table 3.1: Data points in incident reporting formats.
x = Supported or included
* = Can be added, though not standard elements
# = Might be supported, but difficult to say for sure
SM = Floodeen et al. (2013)
EU = The European Commission (2013)

3.2.1 Incident Object Description Exchange Format (IODEF)

Danyliw et al. (2007) describe the current version of IODEF and its data elements, which is listed below and expanded in table 3.1. IODEF was developed in parallel with The Intrusion Detection Message Exchange Format (IDMEF) created by Debar et al. (2007), having a compatible data model, and it appears to eventually having superseded IDMEF as the work on IDMEF was concluded before a final format was completed and standardized. The format is structured, using Extensible Markup Language (XML) for the representation, and thus allows more deterministic exchange of information between two parties. The focus of IODEF is transporting information, not storage or processing, and is therefore not necessarily suited for such applications. In order to be able to represent the necessary details for automatic processing, several extensions to IODEF have been created. These include, but are not limited to: *eCrime* (Cain and Jevans, 2010) and *IODEF Extension for Structured Cybersecurity Information* (Takahashi et al., 2014). Due to its high degree of flexibility and extensibility, IODEF is quite complex. IODEF data elements include:

- Incident id
- Alternative id
- Related activity
- Detect time
- Start time
- End time

- Report time
- Description
- Assessment
- Method
- Contact
- Event data
- History
- Additional data

3.2.2 Structured Threat Information eXpression (STIX)

Structured Threat Information eXpression (STIX) is a format for representing threats, created by The Mitre Corporation (2015). The overall goal is to facilitate sharing of threat information in an automatable manner, and as such, the format is structured and very expressive. The format is developed in cooperation with the Department of Homeland Security and is already used by some organizations. The reasoning behind creating another format, when faced with quite a few preexisting ones, is given to be "*Recognizing limitations in current standardized approaches of representation*" (Barnum, 2012). The goal for the format, as expressed by Barnum (2012), is to represent any information relating to cyber threats – the full spectrum of threat information. This goal of being able to express everything, makes for a very large scope for the format. In order to accomplish this goal, STIX is not only one format, but rather a collection of formats including, but not limited to: Cyber Observable eXpression (CybOX), Common Attack Pattern Enumeration and Classification (CAPEC), Malware Attribute Enumeration and Characterization (MAEC). While STIX is a complex format envisioned to represent most any threat related information, most data fields are made optional in order to reduce the burden on users. While focusing on facilitating automation of threat handling and sharing, the creators also place emphasis on readability for humans. The format uses XML at its core, but the creators are considering allowing other representations of the same scheme. As mentioned, the format is already in use, but not considered final.

While only a small part of STIX, the incident part includes the data points listed in table 3.1.

3.2.3 National Institute of Standards and Technology

Cichonski et al. (2012) provide recommendations on computer security incident handling. Among the topics touched upon, the matter of which information to include in incident reports is discussed. The authors recommend every organization to make their own list of data elements to be collected when an incident is reported. Among the data elements suggested by National Institute of Standards and Technology (NIST), are the following:

- Contact Information for the Incident Reporter and Handler

  - Name
  - Role
  - Organizational unit and affiliation
  - Email address
  - Phone number
  - Location

- Incident Details
  - Status change date/timestamps
  - Physical location of the incident
  - Current status of the incident
  - Source/cause of the incident
  - Description of the incident
  - Description of affected resources
  - Incident category, vectors of attack associated with the incident, and indicators related to the incident
  - Prioritization factors
  - Mitigating factors
  - Response actions performed
  - Other organizations contacted

- General comments

Additionally, some elements directed at the incident handler is included:

- Current Status of the Incident Response
- Summary of the Incident
- Incident Handling Actions
  - Log of actions taken by all handlers
  - Contact information for all involved parties
  - List of evidence gathered
- Incident Handler Comments
- Cause of the Incident
- Cost of the Incident
- Business Impact of the Incident

3.2.4 US-CERT

US-CERT (2015b) has created a list of standard data elements to be collected for each incident. This allows for more standardized incident handling. The data elements to be collected are:

- Contact information for both the impacted and reporting organizations
- Details describing any vulnerabilities involved
- Date/Time of occurrence, including time zone
- Date/Time of detection and identification, including time zone
- Related indicators
- Threat vectors, if known
- Prioritization factors
- Source and Destination Internet Protocol (IP) address, port, and protocol
- Operating System(s) affected
- Mitigating factors
- Mitigation actions taken, if applicable
- System Function(s)
- Physical system location(s)
- Sources, methods, or tools used to identify the incident

3.2.5 Shared mental model

Floodeen et al. (2013) discuss the existence and development of a shared mental model for incident response teams. The authors state that it is generally accepted that faster resolution time of incidents leads to more contained damage. In extension to this, the authors believe that better coordination between actors will lead to faster resolution times. The main question posed in the paper is "*Could coordination be improved by the development of a mental model internalized by the group's technical staff before the incident?*" 90 technicians were asked about the information they would need to collect on different scenarios. The results underline the importance of good tooling and standards: "... *we were expecting to gain more information on their schemas for handling an incident. Instead, the information items that participants listed bore little relevance to actually helping the community handle the incident. It appeared that the teams were citing information items found on a typical cyber security advisory...*", "*In effect the common advisory message format appeared to drive their responses, prompting teams to suggest standard advisory information, rather than based on the information that was important, had value or was* *difficult to work with*" and "*It is as if the mental model they are following is their Incident Ticketing System, not the scenario at hand*" were among the more revealing results.

The project weighted the data elements with regard to both importance and difficulty in obtaining the information, and came up with the following list:

1. Port
2. IP address
3. Event, Log, MSG
4. Domain
5. OS
6. URI, Link, Web Query
7. Process/Services
8. MD5 (detect changes)
9. Proto Header
10. File, Directory
11. Content Strings
12. Hive
13. Key/Key Group
14. Environment Variable
15. Proto Field
16. Session Token

3.2.6 EU: Data Breach Notification

The European Commission (2013) includes a list of data elements required when notifying the competent authority about personal data breach.

- Name of the provider
- Identity and contact details for the data protection officer
- Whether it concerns first or second notification (if not all information is available and an investigation is required in order to obtain the required information, the provider is required to notify the competent national authority within 24 hours with a specified subset of information and provide the rest in a second notification as soon as possible - but no later than three after the initial notification)
- Date and time of incident

- Circumstances of data breach
- Nature and content of data concerned
- Technical and organizational measures applied to the affected data
- Relevant use of other providers
- Summary of incident causing data breach
- Number of users affected
- Potential consequences on users
- Technical and organizational measures taken to mitigate effects
- Content of notification
- Means of communication
- Number of notified users

3.3 Notification

Notifying other involved parties about incidents is not only convenient and useful in fighting the intruders, but for some incidents it is also required by law in an increasing number of countries. California (State of California Department of Justice Office of the Attorney General, 2015), along with 45 more states in the US (Tañà, 2013), have laws requiring notifications in case of a data breach. The European Commission (2013) is also introducing laws to require providers to notify the *competent national authority* within 24 hours of a data breach. There are some notable differences between the function of the laws in the US and within the EU. The main goal of the US version of the data breach notification law is meant to hinder identity theft. The EU version of the data breach notification law has a much broader application area, including:

- Personal data
- Financial information
- Location data
- Internet log files
- Web browsing histories
- E-mail data
- Itemized call lists
- Any information which loss could result in:
  - Identity theft

  - Fraud
  - Physical harm
  - Psychological distress
  - Humiliation
  - Damage to reputation

There are also different quantitative factors in relation to notification requirements. E.g., in California, 500 citizens must be affected in order for the business to be required to notify the supervising authority (State of California Department of Justice Office of the Attorney General, 2015). In the EU, any business is required to notify the competent authority about any and all personal data breaches (The European Commission, 2013).

When it is likely that a breach affects the privacy of an individual, the individual shall be notified without undue delay.

3.4 Summary

Table 3.1 shows the information requested by a few organizations as well as whether the information is representable in IODEF which was identified by Frøystad (2014) as a suitable format for representing all incidents. By including the required elements of IODEF as well as all requested information from each of the examined incident reporting systems of formats, the resulting amount of data elements that needs to be represented amounts to more than 50. That is a substantial amount of data elements, especially considering the results found by Floodeen et al. (2013), that incident handlers request all the information they are able to enter into their incident handling system or advised by their standard. Incident handlers would be collecting large amounts of unrelated and irrelevant information for each incident. On the positive side, a huge format representing all information and incident handlers that input information based on the information rather than the actual case, would ensure that other involved parties received all necessary information. On the other hand, the vast amount of information collected would be a waste of time in many situations and thus increase the cost of doing incident handling.

While it seems like IODEF and STIX are the more powerful and flexible formats, allowing for representing almost anything, this flexibility comes at a cost, as the formats are very complex and require quite some work to implement - both in terms of development as well as required information in the face of an incident. Therefore, a less complex, but still flexible, format will be introduced in the coming chapters. But a simple and flexible format would provide limited benefits if the users were still to rely on exchanging this information by email or phone (Frøystad, 2014). The coming chapters will therefore also introduce a deterministic and structured way of exchanging incident information, as well allow control of which information one wants to receive.

Chapter 4

Interface

This chapter outlines the API used to exchange incidents between actors in the cloud supply chain. The interface presented here is based on the findings of Frøystad (2014).

Note that the format presented here is what was found before the interviews in chapter 7. Based on the findings in the interviews, some changes to the interface might be needed.

4.1 Endpoints

Table 4.1 gives an overview of the endpoints defined in the API. The following sections go into more detail on each endpoint. Endpoints with a plural name, without an *id* specifier, is simply a list consisting of multiple items described in the singular form where an *id* specifier is present. Incident and trigger types can be thought of as classes or templates describing how the incident and trigger objects are to be formatted and handled. These endpoints deviate some from the endpoints defined by Frøystad (2014), in order to simplify the interface and be more in line with current best practice. In addition, all payloads are specified.

4.1.1 Incident type

This endpoint gives the consumer an overview over the available types for incident notification. Each type has a name, a description, an id, and the providers estimate on the consequence of such an incident occurring.

The incident types are specified by the provider, and the provider would need to have conducted assessment on how and when incidents of each type can be provided. This is specified by the provided trigger types. The provider also needs to decide whether an incident of the given type can be automatically pushed to subscribers or if humans needs to manually approve the notification. Declining to send an incident needs to be logged, so that the necessary information will be available for an eventual audit of the provider.

This could be achieved by using dedicated logging utilities such as Transparency Log as defined by Pulls et al. (2013).

Payload

```
{
    "id": UUID,
    "name": STRING,
    "description": STRING,
    "consequence": FLOAT
}
```

Each incident type is assigned a Universally Unique Identifier (UUID), in order to not be confused with other incident types from the current or other providers. A short name is also assigned, making it easier to talk about the incident type and get an indication to what it is about. The description is supposed to give the subscriber enough information to decide if this is an incident type he needs to subscribe to or not, thus the description should cover the important aspects of the type and neither be too long nor too brief. Consequence is a value between 0 and 1, where the provider estimates the consequence of the incident occurring. Given that the provider does not necessarily know how the subscriber uses his system, the consequence will be more about how much damage or how deep into the system a perpetrator could come, rather than how large the consequence would be for the subscriber's system. This could become a way of identifying providers weak spots to facilitate attacks, but it could also become an incentive for providers to ensure security at every corner. An alternative would be for the consequence values to be calculated based upon which services the subscriber uses and for which purpose he has stated the services are used.

4.1.2 Trigger type

This endpoint gives the consumer an overview over the available types of triggers for the incident type in question. This is values to be specified by the consumer, and notification is sent if values are violated for an incident. The provider needs to restrict which triggers are available for each type of incidents, and the values selectable. This will help the provider stay in control of when different types of data are shared with subscribers, while still allowing the subscribers to choose when to receive notifications.

Payload

```
{
    "id": UUID,
    "name": STRING,
    "description": STRING,
    "comparators": ['>', '<', '=', '!=', '<=', '>=']
}
```

Each trigger type is assigned a UUID to be able to tell triggers apart, also between different providers. The name is a short description to make it easier to talk about and recognize the trigger type. A complete description of the trigger type must be provided for the subscriber to decide if the trigger is relevant for his use case and to make an informed choice about how and which threshold to set. The description also needs to include information about how the threshold is to be interpreted. Comparators defines the comparators to use in relation to the threshold value. Even though thresholds are restricted to have the type float, this allows for easy integration with most scenarios if an adequate description is assisting the subscribers in interpreting the values. E.g., true and false values are translated to 1 and 0, system states are enumerated, etc.

4.1.3 Notification Trigger

This is what triggers a notification. Each incident type associated with the notification is assigned one or more triggers. A notification trigger consists of trigger_type and threshold.

Payload

```
{
    "id": UUID,
    "type": {
        "id": UUID,
        "name": STRING,
        "description": STRING,
        "comparators": ['>', '<', '=', '!=', '<=', '>=']
    },
    "method": AND / OR / NONE,
    "threshold": FLOAT,
    "comparator": STRING
}
```

Each trigger is given a UUID in order to not be confused with other triggers, potentially from other providers. Every trigger needs to be of a type defined by the provider, as such the trigger needs to be assigned a trigger type. The method defines the relation of the trigger to the next trigger in the list. AND has presence over OR. When the subscriber creates a trigger, he needs to define a threshold, this is done by setting the desired value of the threshold field. The comparator defines how the threshold relates to the value computed by the service provider.

4.1.4 Notification Incident

This is a combination of an incident type, incident triggers and the accompanying threshold values defined by the subscriber. A notification incident is part of defining a notification type. A notification type can hold one or more notification incidents.

Payload

```
{
    "id": UUID,
    "name": STRING,
    "type": {
        "id": UUID,
        "name": STRING,
        "description": STRING,
        "consequence": FLOAT
    },
    "triggers": [
        {
            "id": UUID,
            "type": {
                "id": UUID,
                "name": STRING,
                "description": STRING,
                "comparators": ['>', '<', '=', '!=', '<=', '>=']
            },
            "method": AND / OR / NONE,
            "threshold": FLOAT,
            "comparator": STRING
        },

    ]
}
```

Each incident has a UUID, a designated type and a set of triggers as described above.

4.1.5 Notification Type

A notification type is a set of incident types and triggers specified by the consumer. A notification type holds at least one incident type and one incident type holds at least one trigger, but each could hold an infinite number of incident types or trigger types. The triggers can be assigned methods on how they relate to the other triggers as AND, OR, or NONE. The trigger method operates on the current trigger and the next trigger in the list. The none operator can only be used for the last trigger in a list.

Incident → Trigger 1 AND Trigger 2 OR Trigger 3

AND has higher precedence than OR, so the above statement is interpreted as (Trigger 1 AND Trigger 2) OR Trigger 3

Payload

```
{
    "id": UUID,
    "name": STRING,
    "endpoint": URI,
    "incidents": [
        {
            "id": UUID,
            "name": STRING,
            "type": {
                "id": UUID,
                "name": STRING,
                "description": STRING,
                "consequence": FLOAT
            },
            "triggers": [
                {
                    "id": UUID,
                    "type": {
                        "id": UUID,
                        "name": STRING,
                        "description": STRING,
                        "comparators": ['>', '<', '=', '!=', '<=',
                            ↪  '>=']
                    },
                    "method": AND / OR / NONE,
                    "threshold": FLOAT
                },

            ]
        },

    ]
}
```

Each notification is assigned a UUID to be interoperable with other systems. A name is given for ease of use. A notification holds a number of incidents, as described above.

4.1.6 Notification Validation

This is an endpoint the subscriber can use to validate that a received incident notification is correct and was actually sent from the claimed sender.

The payload is the notification the subscriber wants to validate. He receives either 200 OK or an error:

```
{
    "error": ERRORCODE,
    "error_msg": STRING_DESCRIBING_ERROR
}
```

4.2 Sent Incident Notification

When the provider pushes an incident notification to the endpoint defined by the subscriber, he uses the following format.

```
{
    "id": UUID,
    "type": {
        "id": UUID,
        "name": STRING,
        "endpoint": URI,
        "incidents": [
            {
                "id": UUID,
                "name": STRING,
                "type": {
                    "id": UUID,
                    "name": STRING,
                    "description": STRING,
                    "consequence": FLOAT
                },
                "triggers": [
                    {
                        "id": UUID,
                        "type": {
                            "id": UUID,
                            "name": STRING,
                            "description": STRING,
                            "comparators": ['>', '<', '=', '!=',
                                ↪ '<=', '>=']
                        },
                        "method": AND / OR / NONE,
                        "threshold": FLOAT
                    },

                ]
            },

        ]
    },
    "generated": TIMESTAMP,
    "sent": TIMESTAMP,
    "sender": PROVIDER.ID,
    "hash": HMAC,
    "incidents": [incident-objects ...]
}
```

Each notification sent, is assigned a UUID, so that the receiver has a way of easily separating notifications. The type referes to the notification type, created by the subscriber that initiated the notification. Timestamps for when the notification was generated and when it was sent is embedded. An identification of the sender is also included. This allows the subscriber to validate the received notification by sending it, in its entirety, to the validation Uniform Resource Identifier (URI) at the provider. This makes it important that the subscriber has a way of validating from which providers he subscribes to notifications; else, a malicious provider might claim to be another provider and thus falsely validate a notification. Therefore, only the sender id is provided, so the subscriber must consult his records of subscriptions to find the correct validation URI. Each notification also contains a hash value, computed with Hash-based Message Authentication Code (HMAC). This allows the message to be validated and authenticated.

The incident objects, in the field *incidents* are represented in the format described in chapter 5.

Resource	Path	METHOD
Incident types	/incidents/types	GET POST* DELETE*
Incident type	/incidents/types/{id}	GET POST* DELETE*
Trigger types	/incidents/types/{id}/triggers/types	GET POST* DELETE*
Trigger type	/triggers/types/{id}	GET POST* DELETE*
Notification Types	/notifications	GET POST
Notification Type	/notifications/{id}	GET POST DELETE
Notification incidents	/notifications/{id}/incidents	GET POST
Notification incident	/incidents/{id}	GET POST DELETE
Notification triggers	/incidents/{id}/triggers	GET POST
Notification trigger	/triggers/{id}	GET POST DELETE
Notification validation	/notifications/validate	POST

Table 4.1: Endpoints in the REST API.
* only available to the owner of the interface instance

Chapter 5

Exchange Format

In order to support two high level approaches to incident handling, manual and automated, without requiring organizations which only need manual handling to implement the more extensive format needed for automation, the incident format consists of different parts. The core format is what supports manual incident handling, this is the case where the incident is read and acted upon by a human. An implementation on this level is titled *Level 1* for the remainder of the thesis. In order to support automation, it is possible to embed more structured formats in the core incident format. An implementation supporting automation is titled *Level 2*. With respect to preexisting formats, as mentioned in section 3.4, their flexibility and completeness comes at a cost, and makes it very difficult to implement the two level system envisioned. It is, however, possible to use preexisting formats in addition to the format specified in this chapter, when aiming for *Level 2* implementation.

A significant amount of time was spent trying to make a unified format using IODEF, but the fast growing complexity and the findings by Floodeen et al. (2013), mentioned in section 3.4, made it clear that it was not the right approach given the principles outlined below.

The fields included in the core format are based on the common denominator of the formats discussed in section 3.2. In addition, the following principles were used to decide which other fields to include:

- Ensure uniqueness
- Provide traceability
- Provide incident handler with flexibility on which information to include – not all information is relevant for every incident
- Make it easy to get an overview of the incident before drilling into details
- Integrate with the concept of incident types

Note that the format presented here is what served as a basis for the evaluated prototype, as described in chapter 6. Based on the findings from the interviews, possible improvements are discussed in chapter 7.

5.1 Incident ID

Each incident needs to be uniquely identifiable, therefore each incident must be assigned a universally unique identifier. In order to obtain the necessary level of uniqueness, UUID version 4 (Internationl Telecommunication Union, 2012) needs to be used when generating incident IDs. It is also imperative to ensure a sufficient amount of entropy in the generation of the IDs.

5.2 Parent Incident

Each propagated incident references a parent, if one exists. This is to keep track of the entire tree of related incidents, allowing for complete audits, criminal investigations or just requests for more information. Since this is only a reference to the parent of the incident, not a complete reference of related incidents, the receiver would need to contact each provider in turn and get the necessary information to go one step further up the tree. Specialized tools could be utilized for audits and criminal investigations, allowing for automatically traversing specific parts of the tree. Maintaining the chain of custody is emphasized by ISO (2013). The parent incident is referred to by its incident ID, as well as its provider and the endpoint at which the receiver of the incident may request more information about the incident if he has sufficient access.

5.3 Incident Type

This property references the incident type the subscriber added to his notification. They are the types created by the provider, defined in section 4.1.1. Referencing is done by embedding the incident type in question, thus synchronizing the meaning of the incident type between the two parties. This also contributes to reducing the potential for confusing situations arising from the provider changing the meaning of an incident type while the subscriber still only knows the former meaning of the type. The representation is equal to that found in section 4.1.1.

5.4 Incident Language

The language in which the text of the incident is written, is to be decided by the provider and the subscriber upon initiating the contract for the service to be delivered. It is recommended that the language is English. The language used must be specified in the format *RFC 4646: Tags for Identifying Languages* described by Phillips and Davis (2006). For an incident written in, e.g., American English, the language field would include the string *en_US*.

5.5 Status

During the life of an incident, it goes through multiple stages or states, such as *Detected*, *In progress*, *Mitigated*, *Closed*. The most important information for the subscriber, is whether the incident is unresolved or resolved. If the incident is resolved, the subscriber might be able to use a different and possibly simpler approach to handling it than if the incident is still active or open. The status field is represented as a string with the value of respectively *Resolved* and *Unresolved*.

5.6 Impact

In order to help the subscriber understand the ramifications and seriousness of the incident, each incident needs to include an impact classification. This is to be considered an estimation until the incident is resolved, at which time the value is to be considered binding and final. This means that the provider is not responsible for the value being correct as long as the incident is open, but is required to provide the correct value upon resolving an incident. This is to give the provider flexibility while working on an incident and at the same time ensure honesty and accountability. The field is represented by a float value, allowing for different degrees of seriousness to be added upon need.

5.7 Summary

In order to support incident handlers at the subscriber to easily comprehend the incident, each incident must include a short summary describing the incident as accurately as possible in as few words as possible. The field is represented as a string.

5.8 Description

Each incident must also include a description. At the very least, the description field must give a general description of the incident. This field may also be used to provide any information the provider finds relevant for the subscriber, but is unable to add anywhere else in the incident report. This field also allows the incident handler to inform about any mitigating actions, recommendations for subscribers and any other extra information he might see fit to provide. The field is represented as a string.

5.9 Occurrence Time

This field describes when the incident first happened on the affected system. If incidents are linked in a parent child scheme, this value holds the time of incident occurrence at the incident root. This value will need to start out as a best guess and evolve into a final value upon resolving the incident. The time needs to include date, time and timezone in the format specified by ISO (2004).

5.10 Detection Time

Information about when the incident was first detected must be included in the incident notification. The time needs to include date, time and timezone in the format specified by ISO (2004). Combining this information with *Occurrence Time* would allow the subscriber to assess how good the provider is at detecting incidents.

5.11 Liaison

Depending on the contracts and Service Level Agreements (SLAs) the subscriber has with the provider, he could have been assigned a personal liaison for security incidents. If that is the case, contact information to this liaison is to be included here. If no personal liaison is assigned, contact information to the security incident handling desk or general support desk is to be included here. If the provider is a sensor or other internal tool feeding the incident management system with information, the operator or maintainer of such tools and systems are to be considered the liaison.

5.12 Attachments

By allowing attachments of a known type, it is possible to represent a rich amount of data in a structured and deterministic way. This will allow automation of parts of the incident handling given that the subscriber has the necessary systems in place to interpret and act upon the information in the formats and that the provider can generate the necessary formats correctly. This allows for reuse of existing formats, while not trying to represent everything that could occur in one huge format.

Examples of formats that might be attached:

- IODEF
- CybOX
- eCrime
- IDMEF
- STIX

In this way, the provider and subscriber might agree on which format to use for each type of incident, and incrementally support more and more automation or just support automation for one specific format for one specific incident type fulfilling a specific set of criteria. Kampanakis (2014) presents a brief discussion on more potential formats.

This would allow for machine interpretable representation of information such as:

- Causes
- Sources
- Mitigating actions

- Vectors
- Ports
- Event logs
- Domains
- Operating Systems
- Targets
- Processes / Services
- Changed files

5.12.1 Cause

The provider should be able to give some insight into what caused the incident. This insight could be provided by a brief description of what the direct and indirect causes of the incident were. This value is likely to be inaccurate, if existing at all, at the start of the incident handling process and evolve into a final value before the incident is resolved. This information could also be valuable for the provider in learning from the incidents and how they are handled. Furthermore, the information is valuable for the subscriber, as he will be able to learn about the providers routines for learning based on the frequency of recurring causes.

5.12.2 Source

The source of the incident could be valuable information for the subscriber in protecting his system from the perpetrator(s) or the perpetrating infected system(s), e.g., in the case of a botnet. The source(s) of an incident could be represented as a list of timestamped IP-addresses.

5.12.3 Mitigating actions

In this field, the provider could account for which actions have been taken in order to mitigate the incidents negative effects. It has several purposes: reassuring the subscriber that the incident has really been handled, giving guidance to the subscriber on one way of handling the incident, and documenting the mitigation for later use or audit.

5.12.4 Vector

The method and approach utilized in order to cause the incident. This could be everything from an inside man to a conventional DDoS attack.

5.12.5 Port

Floodeen et al. (2013) rank ports as the most important information to receive in the event of an incident, when they weight usefulness of information together with ease of obtaining the information. Is is important to note that the personnel they included in their study were technicians from military computer network defense organizations, so this might not be equally important to all organizations. Nonetheless, ports might be valuable in some situations and is representable by several of the formats already defined and in use.

5.12.6 Event Log

Event Logs were also ranked as very important to the technicians asked in the study by Floodeen et al. (2013). Such logs become even more important when the incident happens in the area between systems managed solely by the provider and systems managed solely by the subscriber. If the provider were to provide logs to the subscriber, he would need to have a system in place to extract log items related to the subscriber but not to any other customers. Logstash or Transparency Log, by Pulls et al. (2013), might be of help here. Logs could either be included as a separate attachment or represented using one of the existing formats.

5.12.7 Operating System

In some situations, the incident only concerns a certain operating system. In such cases it might be helpful to know which operating system this is, and thereby be able to prioritize available resources to most effectively mitigate the incident.

5.12.8 Target

Knowing the target might be beneficial in mitigating the incident, though this might also be information the provider is reluctant to release as it might be considered trade secrets or a security mechanism to not publish such information. It might be possible to provide more general, but still useful information, such as *RDS data center, EU-west* for Amazon.

5.12.9 Processes and Services

As with *Target*, knowing the affected processes and services might be beneficial in mitigating the incident, though not necessarily information the provider would hand out willingly. Nonetheless, the information could be included in one of the existing formats.

5.12.10 Changed files

Changed files could be represented in numerous different ways. E.g. diff files from a version control system could be provided, only a list with names of changed files or simply an archive of all changed files.

5.13 Custom Fields

By introducing extra or custom fields, the provider will be able to provide almost any information agreed upon with the subscriber through the SLA, which is too simple to justify the overhead of using one of the complex preexisting formats available, and where the subscriber does not want that particular piece of information to only be included in the description. One field could e.g. be *Next update* containing information about when the provider will provide more details on the incident at hand, or *Ports* containing a list of ports related to the incident.

5.14 Format Representation

```
{
    "id": UUID,
    "parent": {
        "id": UUID,
        "provider": STRING,
        "endpoint": URI
    },
    "type": {
        "id": UUID,
        "name": STRING,
        "description": STRING,
        "consequence": FLOAT,
    },
    "language": STRING,
    "status": STRING,
    "impact": FLOAT,
    "summary": STRING,
    "description": STRING,
    "occurrence_time": ISO 8601,
    "detection_time": ISO 8601,
    "liaison": {
        "id": UUID,
        "name": STRING,
        "email": EMAIL,
        "phone": STRING,
        "address": STRING,
        "zip": STRING,
        "city": STRING
    },
    "attachments": [
        {
            "format": STRING,
            "attachment": URI,
        },
    ],
    "custom_fields": [
        {
            "id": UUID,
            "value": STRING,
            "type": {
                "id": UUID,
                "name": STRING,
                "description": STRING,
                "type": STRING, INT, URI, JSON, etc.,
            }
        },
    ]
}
```

5.15 Notification

The subscriber registers triggers when setting up notifications. These triggers define when the subscriber is to be notified and what he wants to be notified about. The available trigger and incident types are defined by the service provider, allowing the provider to be in control of what information is made available to which subscriber. Additionally, the provider could define mandatory notifications that each subscriber must subscribe to. These could be used by the provider to fulfill legal requirements such as breach notifications.

Chapter 6

Prototype

This chapter presents the work done in prototyping the specification in the earlier chapters.

6.1 Purpose

The purpose of the prototype was created to facilitate testing of incident response tools with experienced incident handlers. The prototype will facilitate an investigation into requirements related to workflow, which information is needed for incident exchange to be effective and useful, and in which cases the incident handlers would notify the subscribers. Therefore, the goal is not to create production ready software, but rather to test the approach presented in the prior chapters. Findings from doing the implementation is used to improve the specification as each problem that surfaces can be examined both from a practical point of view as well as a theoretical point of view. Furthermore, an important task of the prototype is to show that incident exchange is technically feasible, as well as demonstrate how the incident exchange system can be integrated in the workflow of an incident handler. Moreover, having a application capable of demonstrating the system outlined in this report, makes it easier for both the reader and other stakeholders to envision how the system could be used. Therefore, the prototype is minimal and in some respects quite naive, but it is made to meet the goals and purposes mentioned above.

6.2 Technology

Due to the application being a prototype, speed of programming was stressed more than run time performance. Still it was taken into consideration that it should be possible to refactor the application into a production-ready application. The choice of implementing a dummy incident tracker called *IncidentTracker* came after examining the possibility of integrating the system with RTIR from Best Practical Solutions LLC (2015) by creating a plugin. As RTIR is written in Perl and the plugin API is sparsely documented, it was decided to use more familiar and developer friendly a programming language, as well as a framework with more extensive documentation. Therefore, Python was chosen as the programming language, and Django as the underlying framework.

6.2.1 Python

Python is developed by Python Software Foundation (2015) and volunteers. The language is dynamic and interpreted, as well as having automatic memory management. Idiomatic code written in Python is highly readable and usually easily understandable. Its expressive and simple nature made it possible to get started with building the *IncidentTracker* quite easily.

6.2.2 Django

Django is developed by Django Software Foundation (2015) and individual contributors. Django is a featureful web framework with extensive documentation and an active user community. By default, the framework includes features such as database abstraction, model system, view system, template systems, forms support, security measures, internationalization and localization support, geographic framework, and more. In addition, the community offers many packages that can be used with any project at DjangoPackages.com. At the time of writing, there are 2738 packages, of which 653 are compatible with Python 3.

In order to easily create a robust REST API, Django REST Framework was used to create the API and integrate it with the backend. Among the features offered, one can find near automatic API generation from defined models in Django, OAuth, and extensive documentation of high quality. The framework makes it easy to create most simple APIs, but needed some extra work - compared to manual API creation - in order to accept the notification payload defined in the specification.

6.2.3 Template

In order to be able to focus more on prototyping functionality, rather than primarily the user interface, the *IncidentTracker* is built upon the template AdminLTE by Almsaeed (2015). It is a general purpose management template as can be seen in figure 6.1, built using Bootstrap 3 (Twitter, 2015). The template is licensed under the MIT license, allowing use and distribution in both open source and commercial projects. The license is not contagious like, e.g., the GNU General Public License (GPL) (Open Source Initiative, 2015), which requires anything that use GPL licensed solutions to also be licensed under the GPL.

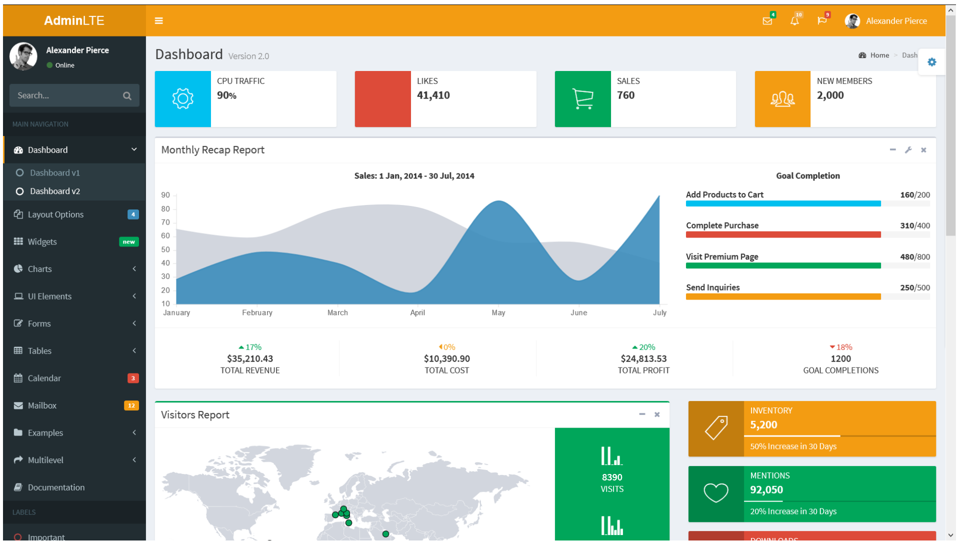


Figure 6.1: The template *IncidentTracker* is visually based upon

6.3 Implementation

The implementation relies heavily on the database for both permanent and intermediary storage. Figure 6.2 shows the main database. In addition, the prototype relies on a small database handling the queue for sending notifications. A better implementation of this problem would be to use an asynchronous queuing system, such as Celery by Ask Solem and contributors (2011), and utilize its native form of queuing and storing queues. In order to keep the diagram readable, the relations relating to the `Provider` entity are not visualized. Entities relating to `Provider` are: `IncidentType`, `NotificationType`, `NotificationIncident`, `NotificationTrigger`, `Liaison`, and `Incident`. This approach of connecting all these entities to the provider is not a requirement of the solution, but rather a simple way for the prototype to keep track of from where local and received information originates.

The user interface (UI) of the prototype does not cover the entire specification, only what is needed to demonstrate its capabilities and facilitate the interviews. What is lacking is subscribing to a provider from within an incident handlers own instance of the prototype. It is, however, possible to add subscribers from a provider, which is the same process as adding a subscription to a provider. Therefore,g this did not impact the interviews.

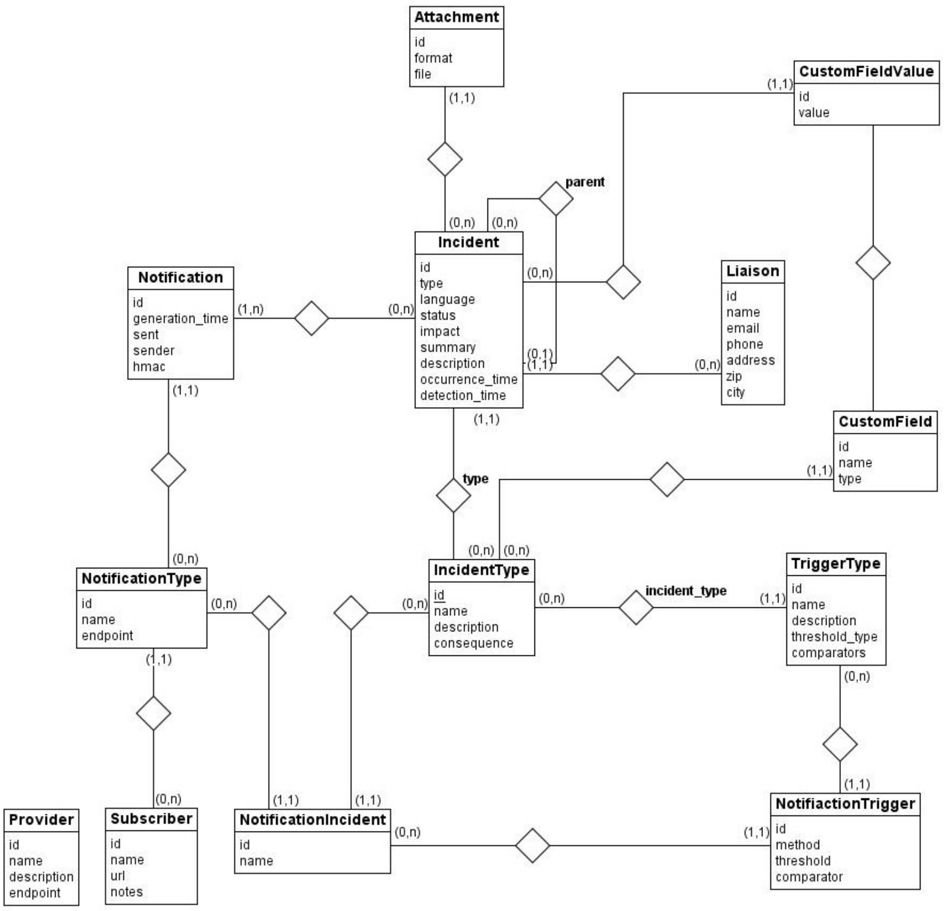


Figure 6.2: Diagram of the internal database of the IncidentTracker

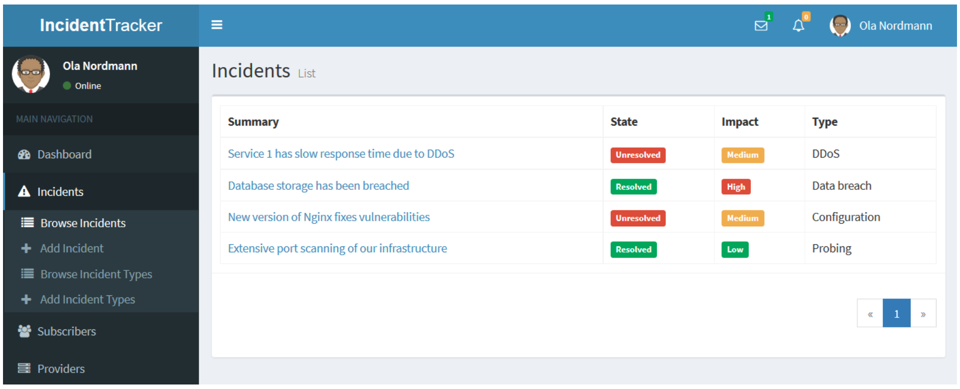
6.4 IncidentTracker

In this section, the incident management tool *IncidentTracker* is presented. This is done by means of showing screenshots and explaining how the tool works.

6.4.1 Incidents

The Incident list is the center of *IncidentTracker*, as this is where the incident handler gets an overview of the incidents in need of attention as well as his starting point for actually handling the incidents. Figure 6.3 shows how such an incident list could look and how it is implemented in *IncidentTracker* at the moment. The shell of the application is a topbar and a sidebar, both are available on all subpages. The sidebar gives the user quick access to all the features of the tool, while the toolbar in the header gives the user access to messages, alerts and profile management.

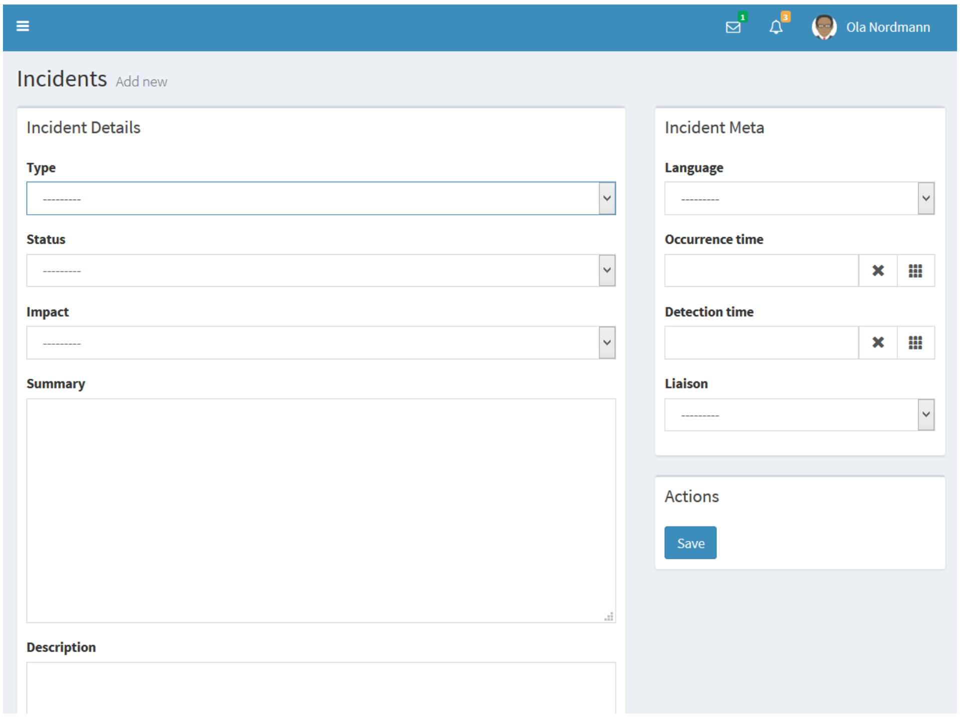
The light gray area is where new content appears on each page. The Incident list in figure 6.3 displays vital information about incidents, in order to help the incident handler prioritize which incidents to handle first. Both state and impact is color coded, in order for the incident handler to easily get an overview of which incidents are resolved and which are not, as well as the degree of impact of each incident. A further improvement could be to only display unresolved incidents by default, but allowing the handler to change filters in order to display those that have already been solved as well. The impact of the incident is a high level method for prioritizing the order in which to handle incidents, even though relying solely on that value means fully trusting the judgment of another incident handler - potentially at another organization with another infrastructure and different threat situation.



Summary	State	Impact	Type
Service 1 has slow response time due to DDoS	Unresolved	Medium	DDoS
Database storage has been breached	Resolved	High	Data breach
New version of Nginx fixes vulnerabilities	Unresolved	Medium	Configuration
Extensive port scanning of our infrastructure	Resolved	Low	Probing

Figure 6.3: The Incident list provides some basic information about incidents, as well as indicating whether each incident's status

Adding an incident is shown in figure 6.4. Adding incidents is done through a simple form, where the the fields are grouped and placed in their order of significance. First, the handler needs to select the incident type, as the custom fields are decided by which incident type is chosen. Thereafter, the status of the incident is added. In most cases this is likely to have the value of unresolved upon entering the application. Going on, the handler estimates the impact of the incident. In the simplistic version implemented here, only three values are used: low, medium, and high. The specification does, however, represent impact as float values. Having provided this basic information, the handler is ready to create a short summary of the incident, before entering all the details necessary into the description field. At the bottom of the page, he is also able to enter information into custom fields associated with the incident type. The right hand side of the screen, holds meta information about the incident: when it was detected, when it occurred, the language in which the report is written and the assigned provider liaison between the provider and the customer.


Ola Nordmann

Incidents Add new

Incident Details

Type

Status

Impact

Summary

Description

Incident Meta

Language

Occurrence time

Detection time

Liaison

Actions

Save

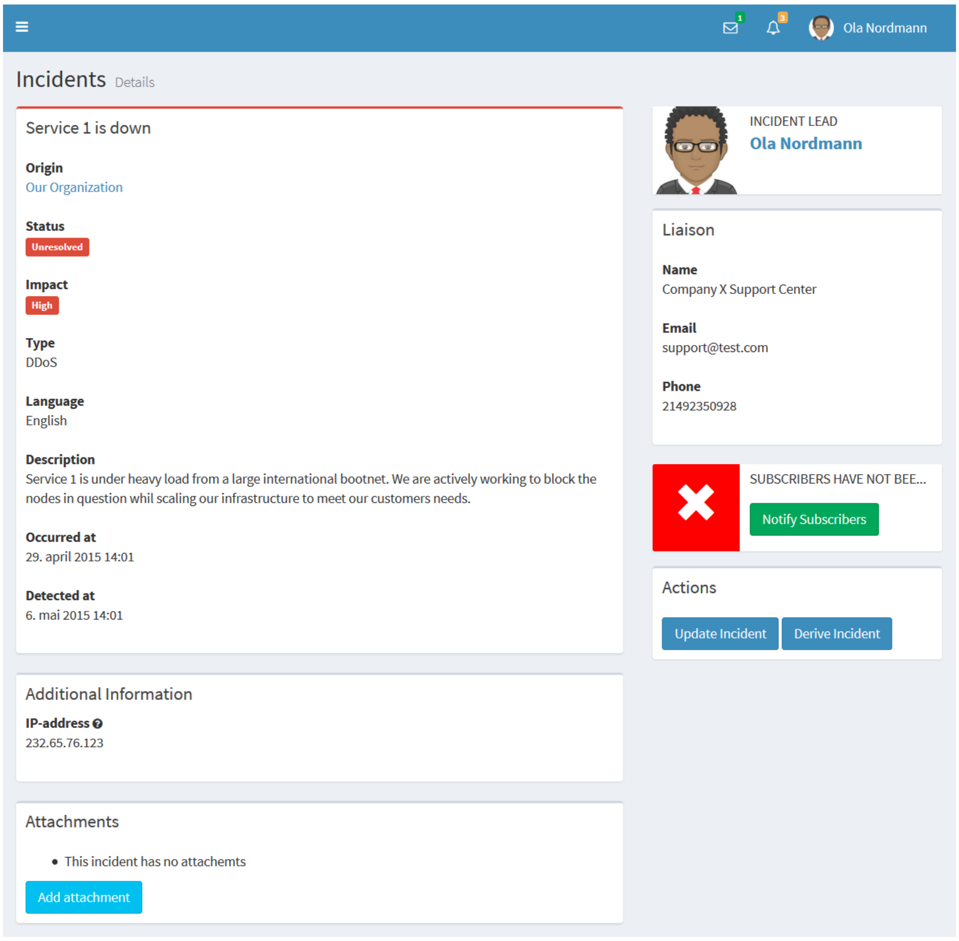
Figure 6.4: Adding a new Incident is done by filling out a simple, customizable form

Figure 6.5 shows the detail view of an incident, in which the handler is also able to add and manage attachments. The two column layout holds multiple boxes of information or actions. The top left is the largest, and holds the information the incident handler is most likely looking for: information about the incident itself. The bottom left box holds attachments as well as allows the handler to add new attachments. Attachments are of predefined types in order to ease their handling. Above the attachments, custom fields and their values are shown, if the incident type has any custom fields associated.

The top right box presents information about who has the lead on the incident in question. This is very important in order to avoid situations where different people believe the others are responsible and the incident is never handled. By designating a specific lead for the incident, and giving this information a prominent place in the incident tracker, all parties involved can be sure that incidents are handled and by whom. This does not mean that the lead needs to work on the incident alone, but rather that he/she is in charge of the incident and its activities.

The middle right box presents information about the incidents liaison - the person to contact if more information is necessary, to provide more information or any other matter. In this case it is a support center, but it might just as well have been a specific person. E.g. for large customers it could have been their designated contact in the provider's incident management team.

The bottom right box holds the actions available to the handler. He is able to update the information in the incident and notify the subscribers if the incident is created by his organization. If the incident is received, the incident handler needs to derive it - create a new incident based on the received one - before being able to notify subscribers. When a handler presses the button *Notify Subscribers*, all subscribers that subscribe to a notification involving the incident type and which fulfills the defined triggers, will be notified.


Ola Nordmann

Incidents Details

Service 1 is down

Origin
Our Organization

Status
Unresolved

Impact
High

Type
DDoS

Language
English

Description
Service 1 is under heavy load from a large international bootnet. We are actively working to block the nodes in question whil scaling our infrastructure to meet our customers needs.

Occurred at
29. april 2015 14:01

Detected at
6. mai 2015 14:01

Additional Information

IP-address
232.65.76.123

Attachments

- This incident has no attachemts

Add attachment

INCIDENT LEAD
Ola Nordmann

Liaison

Name
Company X Support Center

Email
support@test.com

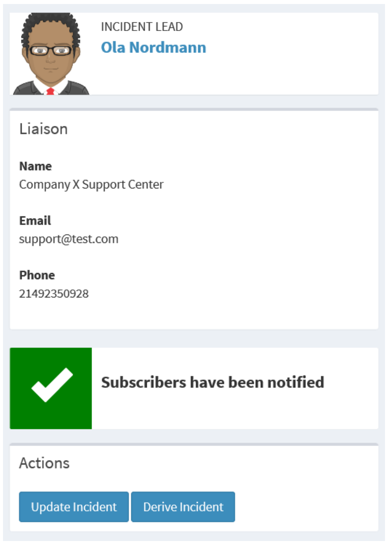
Phone
21492350928

SUBSCRIBERS HAVE NOT BEE...
Notify Subscribers

Actions
Update Incident Derive Incident

Figure 6.5: By opening an Incident, the handler might examine all related information. The image shows a received incident

Figure 6.6 shows the right sidebar for an incident that is created by the organization where the incident handler belongs. At the bottom, the incident handler has the possibility of updating the incident as well as deriving the local incident. Above this box, an indicator shows that all subscribers have been notified about this incident. If the content of the incident is changed, the indicator changes as well. Figure 6.5 shows how the indicator looks when subscribers have not been notified at all, or the incident has been changed since last notification. The incident handler is now presented with a button to notify the subscribers. After subscribers are notified, the indicator goes back to the state shown in figure 6.6.


INCIDENT LEAD
Ola Nordmann

Liaison

Name
Company X Support Center

Email
support@test.com

Phone
21492350928

✓ Subscribers have been notified

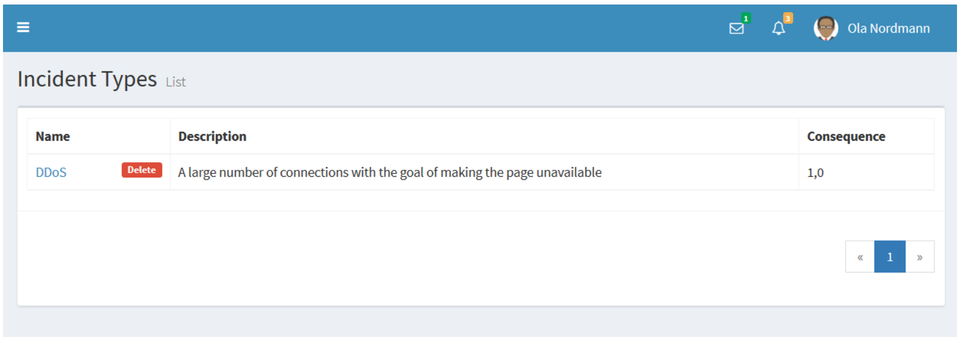
Actions

Figure 6.6: Shows how the right side bar of the Incident Detail view looks, when the incident originates from the organization in question and the incident handler is allowed to send notifications without first deriving a new incident. The status indicates that notifications have been sent.

6.4.2 Incident Types

Figure 6.7 shows the list of incidents added to the system by the incident handlers. This is the way for the incident handlers and the provider to decide which kind of information a subscriber can receive. The naive implementation in this prototype allows all subscribers access to all incident types, but a more sophisticated implementation of the backend could limit access based on criteria defined for each subscriber, distinguish between internal and external incident types, and many more possibilities.

The list gives the incident handler a simple overview of the types of situations and incidents the subscriber could sign up to be notified about. As in the other lists, emphasis is put on making it easy to understand the information with little effort. Therefore, the name, a short description and the provider's estimated consequence is listed by default.

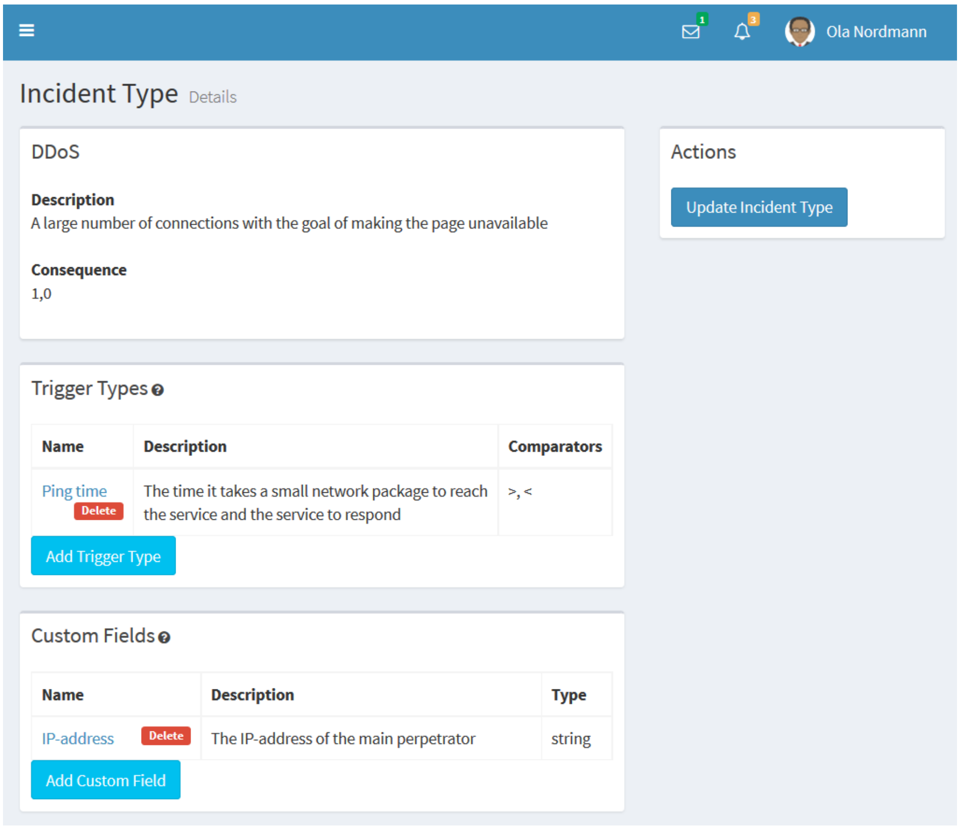


Incident Types List

Name	Description	Consequence
DDoS	A large number of connections with the goal of making the page unavailable	1,0

Figure 6.7: The Incident Type list gives an overview of all Incident Types in the system

Figure 6.8 shows how the incident handler can get an overview of the incident type as well as the connected notification triggers that a subscriber might choose to activate. The top left box gives details about the incident type, while the middle left box lists the associated trigger types. Bottom left, the incident handler has access to decide which custom fields should be available for this particular incident type. From here, the incident handlers are able to modify the incident type and add, edit and remove trigger types as well as custom fields.



Incident Type Details

DDoS

Description
A large number of connections with the goal of making the page unavailable

Consequence
1,0

Actions

Trigger Types

Name	Description	Comparators
Ping time Delete	The time it takes a small network package to reach the service and the service to respond	>, <

Custom Fields

Name	Description	Type
IP-address Delete	The IP-address of the main perpetrator	string

Figure 6.8: By opening an Incident Type, the handler might examine all related information

6.4.3 Subscribers

Figure 6.9 shows the list of subscribers for the incident handlers to manage. In the prototype, this is not connected to a customer database or system, which it would likely be in reality, and as a result the displayed information is a bit sparse. More information about the subscriber, the services or amount of services he uses could be interesting information to add here.

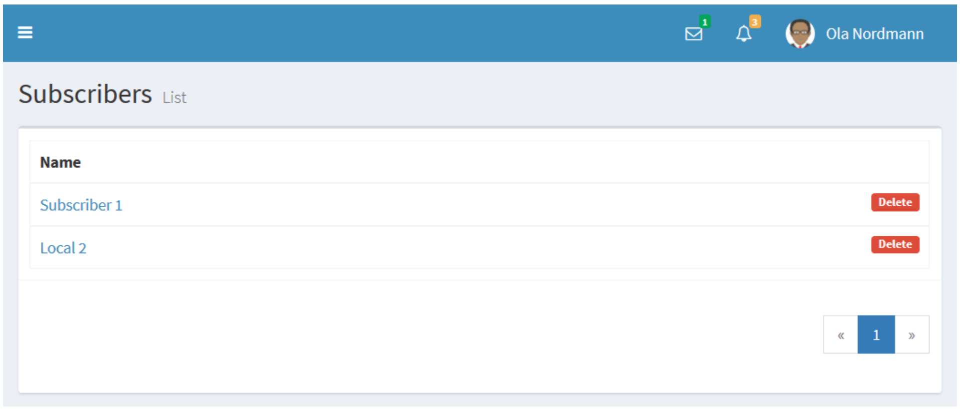
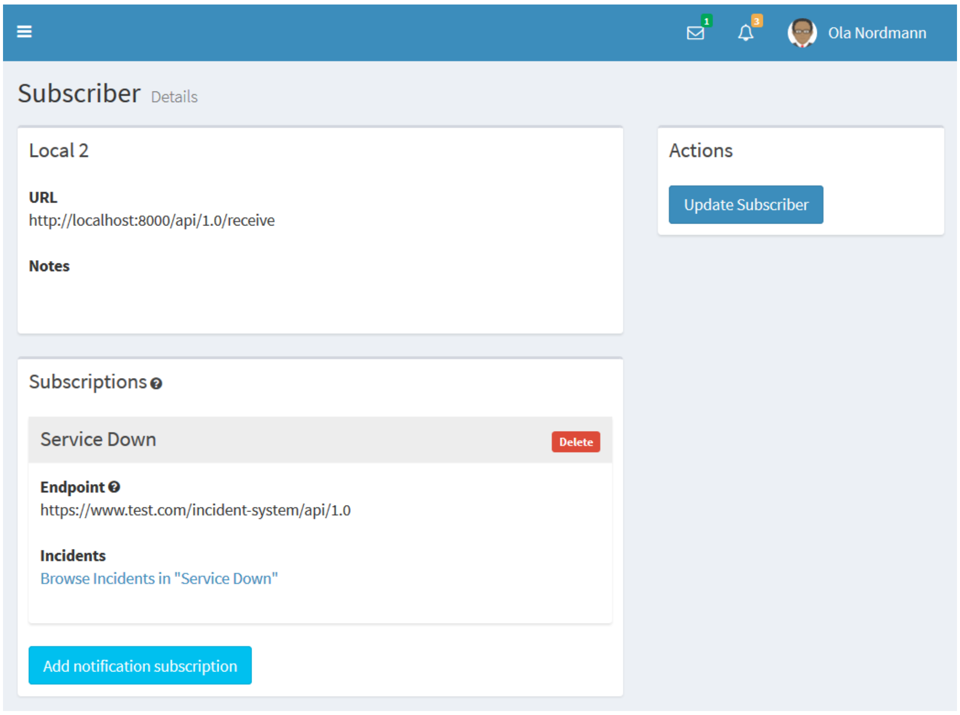


Figure 6.9: The Subscriber list provides a simple overview of the active subscribers

Figure 6.10 shows the detail view of a subscriber. As with the other screens, this is also divided into boxes, each with its own purpose. The top left box presents information about the subscriber, such as its name, main URL and some notes about the subscriber for internal use. A more sophisticated backend would likely include more information and more advanced manners of categorizing and cataloging them. New subscribers are added through a simple form, just like most other content in the *IncidentTracker*.

The bottom left box, in figure 6.10, presents a list of notification subscriptions. This is done using a card view, due to the width of the content not fitting into a table on either desktop computers nor hand held devices. For each notification subscription card, the notification name, its endpoint, and a link to its incidents are displayed. A better user experience (UX) could be achieved by inlining the associated incidents in each card and utilize AJAX to add, modify or remove elements.



Subscriber Details

Local 2

URL
http://localhost:8000/api/1.0/receive

Notes

Subscriptions

Service Down

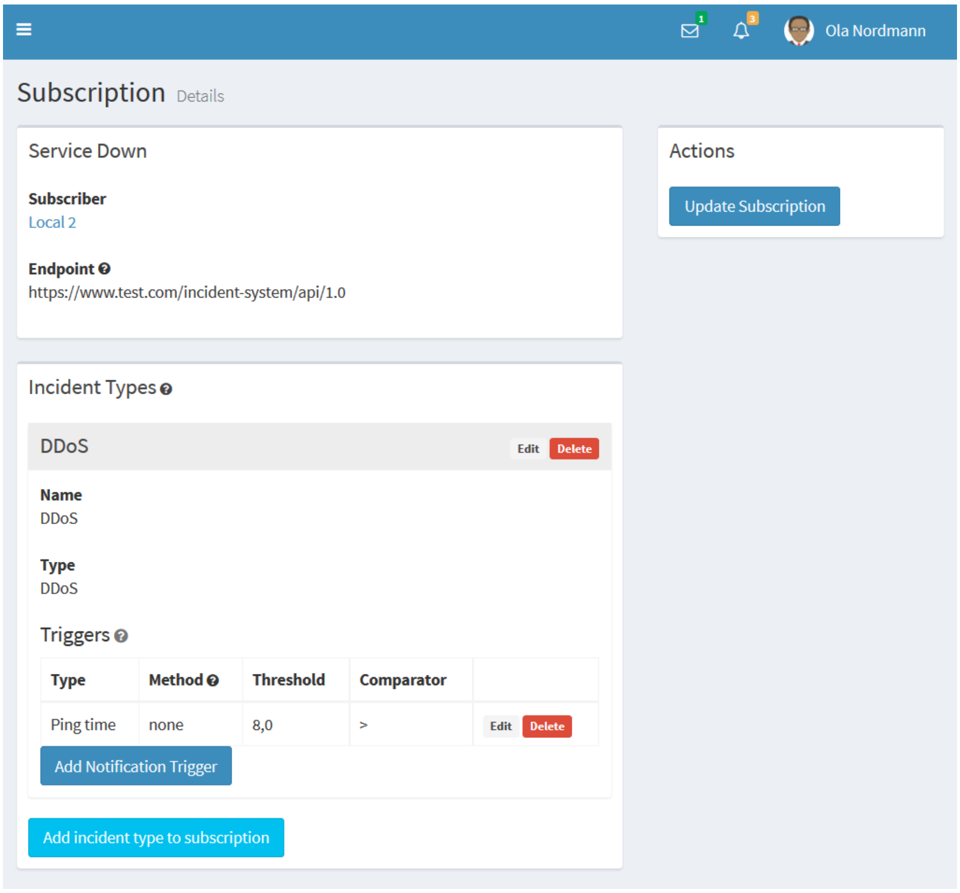
Endpoint
https://www.test.com/incident-system/api/1.0

Incidents
Browse Incidents in "Service Down"

Actions

Figure 6.10: By opening a Subscriber, the handler can get information about the subscriber as well as any active subscription

Figure 6.11 shows the information included in a notification subscription. As mentioned above and as can be seen in figure 6.2, the subscription is linked to the subscriber. In figure 6.2, a subscription is called a *NotificationType*. In the upper left box, information about the subscription is presented: name, linked subscriber and endpoint for where to send the notification if triggers are fulfilled. In the bottom left box, incident types are listed using the same card view as in figure 6.10. Incident types can be added, modified, and removed. For each incident type, the incident handler might add triggers from a list of predefined possible trigger types associated with each incident type. The triggers are displayed inline in the incident type card, using a table to display the information. It is also possible to edit the trigger from the list. A better UX could be accomplished by utilizing AJAX here as well.



Subscription Details

Service Down

Subscriber
Local 2

Endpoint
https://www.test.com/incident-system/api/1.0

Incident Types

DDoS

Name
DDoS

Type
DDoS

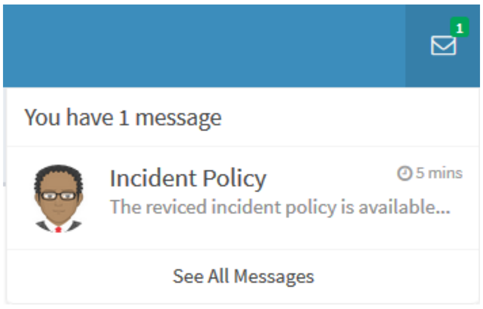
Triggers

Type	Method	Threshold	Comparator	
Ping time	none	8,0	>	Edit Delete

Figure 6.11: Each subscription can be examined and new triggers added

6.4.4 Organization and Cooperation

IncidentTracker could be used not only to list and exchange security incident information, but it is also possible to use the tracker for internal cooperation. Figure 6.12 shows how internal messages could fit into the UI. It is also possible to extend the exchange protocol to make it possible for trusted parties to communicate using trusted channels instead of normal email.

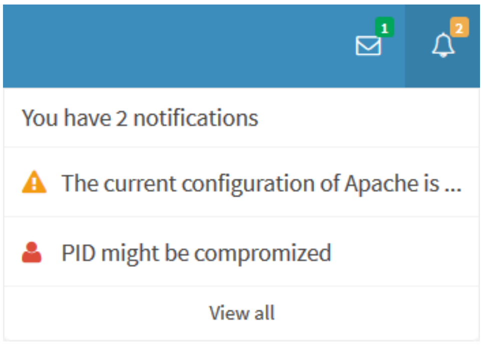

You have 1 message

Incident Policy 5 mins
The reviced incident policy is available...

See All Messages

Figure 6.12: The user could be notified of new internal messages through the interface

Figure 6.13 shows how incident handlers could be notified about new incidents received from other providers, information from sensors, potential problems with configurations, etc. Many of these examples require automation in form of an inspection engine or information being fed from sensors and providers in the format understandable for the system. Drawing from methods used in customer support, one could also define a threshold for how fast incidents of different severities and of different types must be handled, and notify the incident handler responsible if those thresholds are approaching or are not honored. When clicking on the alert, the incident handler is immediately taken to more information and is able to update the incident when it is resolved.


You have 2 notifications

The current configuration of Apache is ...

PID might be compromized

View all

Figure 6.13: The handler could be notified about issues or incidents through the interface

Given the complexity of today's information systems, it is not realistic that every incident handler should be able to handle any incident in the best possible way. Therefore, figure 6.14 includes a link to view the incident handler's competence. In this way, it would be possible to assign the incidents to the incident handler best suited to resolve the issue. In a more sophisticated implementation, suggestions for suited incident handlers could be provided by the system by inspecting the incidents.

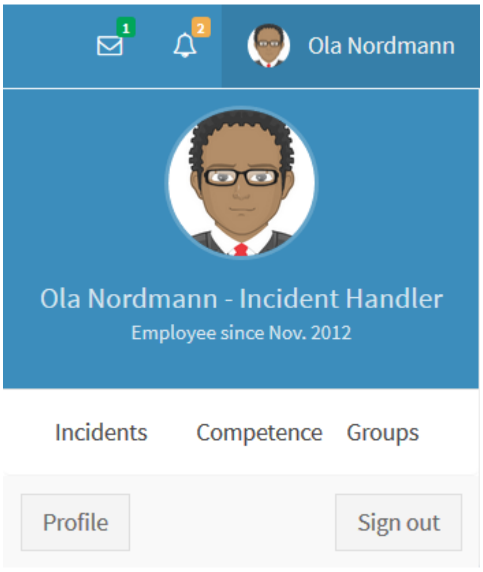


Figure 6.14: The User Profile could include information relevant in delegating incidents

6.4.5 Further Improvements

As this is a simple prototype, quite some improvements are needed for the tool to be really useful and pleasant to use. Some of these needed improvements are already mentioned above, but in this section they will be outlined and structured together with other needed improvements.

Note that the improvements presented here is what was found before the interviews in chapter 7. Based on the findings in the interviews, some further needed changes to the prototype are expected to be found.

User Interface

Due to the UI being quite static, it contains quite deep nesting, particularly when adding subscribers and subscriptions and therefore also when adding subscriptions at providers. By utilizing JavaScript in some areas, separate pages could be merged and the mental overhead on the user could be reduced. It would, however, be important to take care not to overload the user with information on each page.

Furthermore, there is also a need to visually distinguish between incidents created by this provider and incidents received from other providers, and potentially also the relation between different incidents. One way of achieving this would be to display a network structure for as much information as is available. Some incidents would only have the parent from another provider, while others could have a whole trail of locally derived incidents. If merging of incidents is implemented, such a trail would be useful in order to see the development of the incidents. It would be equally useful to be able to see all incidents derived from a parent incident in order to better understand the ramifications of that particular incident.

When deriving an incident from a parent incident, the incident handler should be presented with a screen where all the information from the parent incident is shown as uneditable and with input forms as displayed in figure 6.4. The information from the parent incident should be paired with the correct input field, and the incident handler should be able to check a box for each parent field to be included into the derived incident. The correct information would then be copied into the corresponding input field in the derived incident.

Upon browsing lists in *IncidentTracker*, the incident handler should be able to search and filter on any and all fields. This would allow for complex queries to be created, and as such making it easier to find specific incidents or make other incident handlers aware of a specific subset of incidents by sharing a link or a query. Furthermore, it should be possible for each incident handler to decide which fields they would like to be present in each list.

Automation

It would be interesting if a later prototype could demonstrate automated handling of a small set of incidents with well defined rules. A simple start would be to define rules for which incident handler should be assigned as lead incident handler on an incident. This could, e.g., be accomplished by taking into account the competence of each incident handler, number of related incidents handled, number of active incidents assigned, severity of the incident in question, workdays, etc.

Another automation that could be created is to automatically lock all accounts and require users to create new passwords if the provider's database system is compromised. Such automation is highly specific to the infrastructure and the application in question, and is probably difficult to entirely generalize for use by different systems. A possible approach would be to generalize the rule engine reasoning over received incidents, and provide a plugin or hook mechanism for specializations to connect to. Such an engine would need to provide access to the entire incident as well, in order for plugins to work around any potential limitation in the engine.

Management

Secure internal messages with easy integration with issues, making it easy to reference and connect messages to incidents, would be a nice improvement for traceability of activities conducted in relation to an incident. This could be achieved by integrating an existing messaging or general communication tool, or by building a new that is tightly integrated with *IncidentTracker*.

Alerts currently allow the incident handlers to be notified about incoming incidents, but also changes in incidents, changes in responsibility, and any other matter relevant for the handling of incidents should be implemented. Alerts are tracked per user, so the system knows who has seen the alerts. A much needed improvement here is to relate alerts so that a second incident handler does not have to spend time looking at an alert another incident handler has already handled. A simple solution would be to include the current status of the incident in the alert message. One could also relate incident handlers to incident types, so only relevant incident handlers were notified.

User profiles with a competence directory would allow the incident handlers to easily find the most capable person to handle the incident in question. Especially where the CSIRT relies on a matrix or virtual team organization to pull in the necessary personnel on a case to case basis, this would be useful. This could be achieved by integrating an existing competence and profile system, or building one as part of *IncidentTracker*.

Merging incidents might sometimes be necessary, as an organization might, e.g., receive multiple incidents that all basically have the same solution and consequence. Instead of having multiple parallel active incidents, with multiple incident handlers doing the very same thing, such incidents could be merged into one incident and assigned to one incident handler.

6.4.6 Run locally

In order to run *IncidentTracker* locally for testing, follow these steps:

- Install Python 3
- Install pip (if not included with your Python install)
- Navigate to source folder with your preferred command line client
- Install dependencies: Run *pip install -r requirements.txt*
- Create database: Run *python manage.py migrate*
- Create user: Run *python manage.py createsuperuser* and follow the instructions
- Start server: Run *python manage.py runserver*
- Start another server: Run *python manage.py runserver 0.0.0.0:8800*

You'll now have one or two servers, depending on whether you started the last one or not. The servers are accessible at `localhost:8000/manage/dashboard` and `localhost:8800/manage/dashboard`, respectively. The API is available at `localhost:8000/api/1.0/...`

6.4.7 Limitations

IncidentTracker does not implement the entire specification, due to time limitations and the priorities listed at the beginning of this chapter. The system has not received any UI for subscribing to another provider from within the system, nor has HMAC signing and notification validation been implemented. Securing the channel between two instances of *IncidentTracker* is currently not done, and logging any and all activity is not done either. Triggers are implemented, but due to their need for close integration with the underlying infrastructure in each case, these are not honored when sending notifications. Neither are the notifications sent in an asynchronous manner.

This all comes down to the prototype not being complete, rather than the specification being lacking.

6.5 Impact on Specification

During creation of the prototype, some changes were made to the specification due to problems that became apparent during implementation.

Initially, the notification payload included references, by URI, to types, parent, etc. This was changed to fully embed any related information. Custom fields were made more generic and the custom field type embedded next to the value, rather than creating a different flat representation of both value and type. The first versions of the current incident representation format included attachments as embedded content, encoded with Base64. This was changed to the format including URIs to fetch attachments when needed. Some cycles in the interface payloads was removed.

Chapter 7

Interview Results and Analysis

This chapter presents the results from the focused interviews, catalyzed by the prototype and defined scenarios, as well as the analysis of the answers. The interviews were conducted at the participants venue, and he was given the scenarios defined in appendix D.

The results are organized by subject; incident format, incident exchange, and workflow. For each subject, the answers and reflections of the participants are presented as purely as possible, and the author's analysis is placed in separate corresponding sections.

7.1 Incident Format

Participant A pointed out that the incident reports presented as part of the interview, were mostly complete, though logs were missed as his reflex behavior was to investigate the incident. After being reminded that the incident originated from another provider, he found the message format acceptable, but still missed more detailed information about timezone, time for next update and expected amount of time to pass before the incident is solved. *Participant B* also found the format mostly complete, but would want to have a way of expressing recommended actions to guide the receiver in how to face the incident.

7.1.1 Results

Next update

From other, similar situations, *participant A* is used to status updates being provided with a specified interval. He suggested the format to include a field stating the time for the next status update. This would avoid the incident handler having to wonder if the provider is doing anything about the received incident, without having to hamper the work of the provider by making them have to attend to update requests from customers rather than actually handling the incident.

When creating an incident in the system, *participant B* ended the incident description with the following note: "We are sorry for the inconvenience this might have caused you and will provide more information as soon as we know more.” He also said there could be more severe situation where regular updates were necessary, even if they just said “Nothing more to report at this time.”

Expected time of correction

Closely related to the next update field, was *participant A*s desire for information about an estimate for when the incident would be solved.

Channel

Participant B expressed a desire to know from which reporting channel an incident originates. E.g. was it reported by phone, email, etc. It is also interesting to know who reported the incident to the organization from which the incident originates.

Impact

Impact was said to be of significant importance, though understanding the value might be a more involved task. *Participant A* wondered whether the value applied to the incident’s impact on the provider itself, or if it was the provider’s estimate for the impact on the customer.

Participant B expressed the difficulty of defining an incident’s impact, in the case of incident 1, without a through investigation. He also pointed out that the impact would be very different if the affected provider was his only provider and the problem affected all services, rather than being one among many providers and serving a less important role.

Time

Participant A pointed out the need for exact timestamps in relation to any information that could change as time passes, IP-addresses being one such example. This was brought up while handling incident 1, which includes an IP-address as a custom field.

Accurate time is very important, as pointed out by one of the participants. He was particularly concerned that information about which time zone the provided times originated from, given that the prototype interface he was presented with only printed date and time.

Participant B did not understand the point of having two separate time fields. Detection time is easy to identify and provide, but occurrence time might prove more difficult to identify. Occurrence time might prove useful when identified though.

Attachments

Participant A said logs were the single most important information source for handling incidents, and would therefore like to receive logs whenever possible in cases where he or his team were expected to do anything about the incident. He did, however, not believe automation in retrieving logs were the way to go, as automation could not judge which logs were important. He believed that a human would need to fetch the relevant logs. *Participant B* stated that for simpler, automated attacks it is possible to detect the attack and fetch the relevant logs automatically. For more sophisticated, manual attacks he was more doubtful that automation would be able to find the relevant information due to attackers being diligent in cleaning up after an attack. Even unsuccessful attacks would be relevant information if they differed in some way from the usual background noise on the internet. Incident 4 was used as an example of usual internet background noise he would not like to receive, and which made the provider sending the incident appear incompetent with regard to deciding which information is relevant.

7.1.2 Analysis

Next update

Such a field for next update was considered in early versions of the incident format, but discarded because of being perceived as non-vital or unnecessary information. If the updates are non-updates, e.g. just resending the original incident, this would probably not be helpful for the receiving incident handlers, other than potentially indicate that there is activity related to the incident at the provider. If update times are automatically enforced, a situation could occur where the provider is not handling an incident, while the customer is under the impression that he is. Therefore, it was decided to not include information about the next update in the main format, but rather encourage the use of custom fields for this purpose in a mutual agreement between two parties. The interviewees did, however, make a valid case for the field to be included, and it should therefore be examined further if it should be included as a standard field.

Time for solution

This field was not considered before, but is still easy to include if it proves important and is also representable by the current custom fields. The expected time of a solution could allow receiving incident handlers to better plan and allocate the necessary time to perform local activities after the provider has completed his. A potential challenge would be the difficulty of estimating how long it will take to solve an incident. This is closely related to the decision of when to notify customers and subscribers. If a provider has a policy of notifying subscribers instantly when a new incident is detected, experience could be used to guess when the incident could be solved, but this would probably not be an accurate estimate. If subscribers are notified after the investigation phase is conducted, the estimate would probably be more accurate. Nevertheless, an estimate of whether a solution would take minutes, hours, days or weeks, would probably be valuable to the subscriber and allow him to take more justified decisions on how to proceed.

Reporting channel

Participant B expressed the desire to know from which reporting channel an incident originated. He would like to know if it came by email, reported by phone or through any other channel. This is partly supported by the current solution, as a notification includes information about the sender and the format includes information about the any ascending incidents. In order to represent the actual channel used, and the person or entity initially reporting the incident, this would have to be done using custom fields or simply written in the description. There are, however, the question about what added value this information brings. If organization A receives a notification from organization B, A should not need to know who reported the information to B. A should simply act as if the incident originated from B. If A needs more information about the incident, he would ask B which in turn would ask the initial reporter if need be. This way, one can avoid short circuiting the cloud supply chain and ensure that the provider is able to control the information flow.

Impact

One participant asked about who the impact field applies to. Since the provider usually does not have intimate knowledge of e.g. which information the customer stores in the provided database, the impact value is the provider's estimate on how the incident impacts their own services. If the receiving incident handler decides to derive the incident and notify their customers about the problem, he would have to adapt the impact value to take into consideration the impact on the provider, the importance of the services, the content, and any other information that could affect the impact estimate. This is in line with the reasoning of *participant B* as well, who stressed the importance of investigating and having intimate knowledge of the system use in order to decide on a proper impact value.

Time

Accurate time is of crucial importance. The particular problem about timestamped IP-addresses could be easily solved by adding another custom field for the specific time at which the IP-address was identified. Another solution would be to attach a more complete and complex format, like IODEF or STIX, to represent this information. Finally, the information could be represented by a JavaScript Object Notation (JSON) custom field containing both the IP-address and the timestamp.

The format represents time accurately down to seconds, according to ISO 8601, and also supports time zones as requested by *participant A*. The reason for the confusion was the prototype converting the received time to the incident handlers local time. In order to reduce confusion and ambiguity, tools should provide the user with an indication of which time zone the time belongs to. If local time is chosen to be the time zone in which time should be presented, it should be easy to view the time in its original time zone and vice versa.

The reason for having two different fields representing time, detection and occurrence, was to increase awareness surrounding for how long the incident has been in effect. If only one of the fields were included, this would have to be the occurrence time field, which is the most difficult time to pinpoint. The combination of the detection and occurrence time also gives the subscriber a means to measure how good the provider is at detecting incidents. The shorter the interval between occurrence and detection, the better. Understanding that occurrence time is difficult to pinpoint, and needs through investigation in order to identify, the occurrence time field is to be considered an estimation until the incident is closed at which time it should be exact.

Attachments

The solution allows for exchange of logs that are found either automatically or by human incident handlers at the provider. In very simple cases it might be possible to fetch the correct logs by means of automation, but in more complex scenarios an actual incident handler would need to choose the right logs. A more complicated problem with regard to exchanging logs, is filtering the logs to include only the information the subscriber is allowed to access. It might be possible to use Transparency Logs by Pulls et al. (2013) to only fetch data the subscriber is allowed to see.

7.2 Incident Exchange

Participant A stated that their team is currently exchanging some incident information by means of email and phone. Some emails are automatically created based on information in their incident management system. *Participant B* said that solutions like the dashboard are already common, but the API and the incident format are what CERTs could start using. The important thing is that it is compatible with incident management systems already in use. The concept is a good alternative to the more common email approach.

7.2.1 Results

Notify

Upon examining incident 2, *participant A* stated the need to contact the sender, as the incident has been marked as resolved without indicating which actions were taken. This does not build confidence in the provider's proper handling of incidents. Furthermore, he finds the information too vague. In his opinion, the notification was sent too early, since the provider does not know exactly what has happened and to whom it has happened. This raises the important issue of when to notify customers. His view was that a notification must be concrete, with regard to what has happened and to whom, otherwise it would only scare customers and cause more work for incident handlers. *Participant B* also stresses the importance of not notifying before knowing for sure that customers have been affected. In contrast to *participant A*, *participant B* finds that incident 2 increases his trust in the provider as it shows that the provider has systems for detecting problems, has a policy for sharing information and has thought about this sort of incidents beforehand. If everything went well and no information belonging to his organization was compromised, he would spin the situation into being a good exercise to improve awareness of which information is stored in external services. Like *participant A*, *participant B* would only notify all customers if the incident reached the media. Otherwise he would only notify those actually affected by the incident.

When asked about notifying customers, the *participant A* stresses the importance of notifying only those who have actually been affected. This means that he would need to gain access to information about who had data on a particular system at a particular time. He would not notify customers before fully investigating the incident. If the entire system and all of its content was compromised, he would consider doing a press release about the incident. On a direct question from the interviewer on the usefulness of the solution presented in this thesis, he acknowledged that the solution could be useful if the recipients had the same system and they themselves could notify end users.

Upon being asked about the threshold for pressing the "Notify subscribers" button, *participant A* characterized this a high threshold for incident 2. He described the threshold for pressing the button as lower for less severe incidents, like incident 1. *Participant B* would like to see who receives the notification before actually notifying, and that trust must be in place before any information could be handed out.

In relation to incident 2, *participant B* pointed out that there were two types of incident sharing that were relevant. First notification of customers, advising them how to cope with the situation. Secondly, if the incident has occurred as a result of a specific vulnerability, this is information relevant for others in the industry and should therefore be shared. He would notify the national CERT which in turn would assess the information and has the means to issue a general warning to the entire industry. A simple button for notifying the national CERT would be useful.

With regard to incident 4, *participant A* pointed out that he would not like to receive this kind of incidents, as they were part of normal operations and he assumed that the provider handled them correctly.

In order for a system supporting exchange of incident information to be effective and be adopted, it needs to be integrated with internal systems. *Participant A* used an example of some problem being related to an IP-address, where the system would need to know how to consult their internal database of IP-addresses and users to know whom to notify.

When asked if notification is an activity conducted in the final phase of an incident, it is said to vary on a case to case basis. Incidents are different and many different situations could occur, making it difficult to give an exact answer to when incident notification is or should be done. It is also important to take care not to notify too often, as that might cause important information to either be lost in lots of information or in the devaluation of notifications from a provider. The organization of one of the participants seldom sends notifications, but does so for incidents that are important. He also points out that their organization would not like to receive too much information from other organizations either. *Participant A* is not interested in receiving information about incident they do not find important or they are unable to handle. He thinks it would be acceptable to receive information about incidents that relates to them. Thus, it is a balancing act to figure out how much and how often to notify customers.

As mentioned in the section about the format, *participant A* requested information about when the next update, concerning the incident, would be received. His experience from operations makes him believe this information to be useful as it makes the incident handler confident that the incident is actually being handled by the sending party. The customer is, in some cases, likely to request more information from their provider. If this provider receives information about the incident, they will be better placed to serve their own customers.

Reply

After having sent incident 1 to his customers, *participant B* would reply to the provider and inform him that the information has been passed on. He could also ask if there are more information to be shared. He would also have used this functionality to request more information about how the relevant information was obtained, in the example of incident 1 he would ask about the IP-address. He is used to all communication being integrated into the incident management tool, allowing external communication to be done inline. He points out that this might become a problem if the provider receives several thousand replies, and therefore suggests to include a comment section instead. Comments could be seen by all receivers of the incident or just the provider. The provider could then decide which comments to reply to, and which questions they decide to answer.

Internal

When handling incident 3, *participant A* commented that he would have contacted the person responsible for the tool in question and asked him to rectify the issue. The way this is handled at the moment, is by sending an email or walking the halls in order to notify the right people. The internal communication surrounding incidents is an area in which he wishes they had better tooling in place. *Participant B* also stressed the importance of communicating this incident to the system administrator as soon as possible, and that the task of upgrading the outdated software should be of the highest importance. If the system administrator has access to this system, he should be made aware of the incident. Otherwise the content of the incident should be copied into an email for the system administrator to receive.

Approval

Participant A points out the need to have information sharing approved by upper management. In the case of incident 5, he would need to get approval from the chief executive before handing out any information.

Honesty

Participant A stresses the importance of being honest about what has happened, and not try to hide the incident. In relation to incident 5, he said it is not normal behavior to notify customers or partners of incidents. He does, however, see the industry moving towards becoming more open about incidents and breaches in their systems.

7.2.2 Analysis

Notify

The lack of information in incident 2 was pointed out. This underlines the fact that the solution presented in this thesis does not solve all non-technical problems. If an incident handler or an organization provides to little information, this might be a problem with the culture in that particular organization. A possible solution could be to provide more specific and mandatory fields in the base format, but this would be very difficult to make general enough to not be a hindrance in other situations. If to little information is a persistent problem, this should be handled at contract level between two organizations. Such contracts could e.g. specify which information should be included in each incident notification, and custom fields or attachments could be used to communicate this information.

With regard to notifying customers, it is important to keep in mind that there is a distinction between notifying the incident handlers at another organization and notifying end users. It is possible this was not communicated clearly enough to *participant A*, as he was concerned about receiving many requests from end users for further information - taking resources away from actually handling the incident.

The problem of notifying end users was considered briefly when designing the interface and the incident format. The reasoning was that the last link in the Cloud Delivery Chain, before the end user, had the best - an quite possibly the only - overview of which of their customers are affected by different incidents. Thus, CSPs should send incident reports to their subscribing customers, which in turn would use the received information, together with intimate knowledge of their own systems, to notify any direct end users and provide relevant information to their subscribing CSPs. This is equal to the procedure *participant A* describes for notifying their direct customers.

The difference in threshold for notifying subscribers, depending on the type and the severity of the incident, is another non-technical problem. While this solution does not fully handle this problem, the need to create incident types is likely to help build awareness that such severe incidents could happen. Ideally, the organization should also create accompanying procedures or rules for notification and who should be involved in the different incident types. This way, many considerations surrounding legal problems and public relations could be made without the added stress of an active incident, while at the same time allow incident handlers to move faster in the face of an actual incident. It might be unrealistic to expect every eventuality to considered beforehand, but if most incident types could be handled by following a predefined script, it is likely to be easier to devote the required attention to those incidents which can not be handled in that way.

Both the interface and the prototype supports allowing the subscriber to choose which kind of incidents he wants to receive, and he could thus simply choose not to subscribe to incidents their organization does not find useful. This does, however, require that the provider has created suitable incident types he might subscribe to. This way, both participants could have avoided receiving incident 4, and combined with triggers, this would allow the subscriber to decide how much information he wants to receive and thus help with balancing the amount of incidents received. The provider could then notify for all incidents, and the subscriber would be responsible for setting triggers and subscriptions that are relevant for him.

Integration with existing systems was said to be a critical factor for such a solution to be adopted in the industry. This was one of the core consideration made when designing the solution. The solution separates the problem into three distinct parts, where the format and the interface is what is specified in this thesis, and the implementation of a backend is left to the implementer. Thus, organizations could integrate the solution into their existing systems and still be able to communicate with other instances implementing the same interface and format. If, at a later time, an extendable interface for automation was created as well, it would make it easier for organizations to change incident management tools, as the specializations for their infrastructure would still work with a new system.

Replies

Participant B pointed out that he would like to be able to reply to the sender to inform him that the information had been shared and to request more information. He acknowledged the potential problem of overloading the sender with replies and questions, and therefore suggested a comment functionality available to all recipients of the incident. This seems like a noteworthy idea that should be examined further, and might improve collaboration around handling incidents. This would, however, likely increase the importance of solving the underlying problem of trust.

Internal

Both participants commented on the need for internal communication. This is an area where the interface and the format probably is of less use, but the prototype presents some ideas that might be useful. The database of who is responsible for what and has which knowledge, combined with the possibility to assign incidents as well as message other users, could be helpful in the case of notifying the correct person. This would, however, require that all employees were users of the incident management system or that the incident management system was closely integrated with the company's existing intranet or communication system.

Approval

The need for approval before sharing incident information was brought up by *participant A*. As mentioned before, when discussing notifications, the fact that providers would have to list all incident types the customer could subscribe to, gives them the opportunity to also consider sharing of such information. By creating clear rules for when information could be shared instantly, when it could not be shared at all and when it would need to be signed off by an executive, it would be less pressure on deciding in the face of an incident.

Honesty

Participant underlined the importance of being honest about what has happened. This further strengthens the impression of the need for bidirectional trust, for incident information exchange to work. If a provider is not honest when describing an incident, the information might not be useful to the receiving organization or in worst case be harmful. If it is discovered that an organization is dishonest in its use of received information, the sending provider might not be willing to share any more information with this particular organization. Honesty and trust is, thus, intertwined requirements for such incident information exchange to work. To some degree, it is expected that this trust and honesty can be established by means of contracts.

7.3 Workflow

While the prototype supplied to the participant was only meant to facilitate reflections surrounding incident representation and exchange of incident information, the participants had some crucial reflections about workflow that is worth recording for further work in the field.

7.3.1 Results

Local vs Received Incident

Participant A pointed out a problem related to local vs received incidents. The user interface did not distinguish them clearly, and he therefore wondered who the status field applied to. If it was resolved, was that for his organization or from the perspective of the sending organization. Furthermore, in his view, the local incident would be resolved after forwarding information to customers, at least in the case of incident 1, but this would not be the case for the provider.

Participant B pointed out the need to be able to send a derived incident with another incident type than that of the received incident. In the example of incident 1, he would receive an incident about DDoS, but to his customers it would only be a reduction in quality of service.

Traffic Light Protocol

When receiving information, *participant A* said it was normally encoded using the Traffic Light Protocol (TLP). The TLP indicates with whom the receiving party is allowed to share the received information. E.g. only the receiving person is allowed to view the information, the CERT, security people, etc.

Participant B pointed out, in the context of incident 1, that in the absence of TLP indications on the incident information, he would have to contact the sender to obtain permission to share specific information like the IP-address of the main perpetrator. He might send a general message to his customers about the problem, then obtain permission from the sender, and finally update the information he provides to his customers. If the entire incident, or parts of the incident, were marked with TLP, he would not have to do this extra work, but could act according to the TLP marking.

Organization

Participant A pointed out the need for a notification system to build on, or be integrated with, a incident management system. This close connection is needed in order to avoid doing extra work, like adding the same information twice and updating incidents in two different places. They have some email notifications integrated in their current incident management system, allowing easy notification about simple incidents. Recipients are subscribed to different email lists, allowing the organization to send notifications to the relevant personnel. When creating local incidents, derived from received ones, *participant A* pointed out that these should only be visible from the parent incident.

Participant B expressed the importance of powerful search and filtering options, as well as the possibility of tagging incidents for easy retrieval later. In addition to organizing incidents in a parent-child system, they use tags to create sibling incidents. A main ticket could hold hundreds of children, and in such situations graphing the relationships are useful.

Queue

Participant A described how their current tool handle incidents and that this was preferable. Incidents are added to a queue, and when an incident has been handled, it is removed from the queue. This makes for less information the incident handler has to browse in order to find what is important at that time. It is also important to know the internal state of the incident, like who is working on it, what has been done, which information is given to the customer, etc. The workflow of such a tool is crucial, as usually an entire team is working on an incident. Updates to an incident, from the provider, must be added to the original incident in order to avoid extra work.

Integration

Participant B pointed out that he would like the system to integrate against e.g. email, so that notifications could be sent to email lists as well as other instances of this system. He would also like to be able to receive emails directly into the system. This would provide the proper traceability for information as incident handling is mainly communication.

Automation

Participant B expressed that their current solution relies on scripts and automation in order to organize large amounts of received information. In the example of incident 1, they would prefer to receive a list of all involved IP-addresses, organize them in geographical order and send the relevant sections of the list to system operators, Internet Service Providers (ISPs) or local authorities.

7.3.2 Analysis

Local vs Received Incident

Participant A pointed out the need to distinguish between local and received incidents, which is not done clearly enough in the user interface of the prototype. Handling this particular problem, is left to be done by those integrating the solution in existing tooling. If the prototype should be developed further to enhance this aspect, it could be done by having separate lists for local and received incidents, having different background colors on the rows, having an icon indicating that the incident was received, or having a cell stating the origin of the incident. The preferred approach would be to indicate the origin, perhaps with a small icon in addition to the name of the origin, as well as allowing the incident handler to filter the list to show only local or only received incidents.

*Participant B*s need to change the type of the incident came as a surprise, but was perfectly logical and reasonable once explained. The solution proposed in this thesis supports this in every part; interface, format and prototype.

Traffic Light Protocol

The TLP was mentioned by both participants. Implementing the traffic light protocol in the solution would consist of two elements: including the traffic color in the incident format and requiring organizations to adopt the protocol. The first element is simple, though complicated by the second element. The Traffic Light protocol differs slightly between the US and the EU, which results in the solution having to support both or requiring the organizations to adopt only one of the protocols. If the solution should support both protocols, a prominent problem would be that the traffic light colors might no longer be as useful because of the ambiguous meaning.

Organization of Incidents

With regard to being integrated with the incident management system, as mentioned before, this was an important consideration when designing the solution. When it comes to derived incidents only being shown from their parent incident, this is a decision that would need to be made by the implementers of the solution. There might, however, be challenges with only allowing derived incidents to be viewed from their parent incident. If an organization receives a very broad incident and chooses to derive multiple very different incidents, it could fast become difficult to navigate and always have the necessary overview of the situation. A possible enhancement could be to provide filtering for derived incidents, as well as being able to expand parent incidents in the incident list. A parent incident could even indicate how many derived incidents it has as well as how many of those are solved.

The importance of search and filtering was also brought up i section 7.3.1. This has been acknowledged during the development of the prototype, but not prioritized as the concept was the goal, not the workflow. The importance of such functionality has, nonetheless, been pointed out in section 6.4.5. Including tagging is a valid idea, and one integrators of the interface and format should consider for their implementations as this is a issue that only concerns the backend. This is also the case for *participant B*s wish, in section 7.3.1, to have email and other communication utilities integrated in the backend.

Queue

Participant A presented his view on how an incident management tool should work. The queuing approach is well tested and widely applied in commercial customer support software, and it is possible the workflow should mimic such solutions. If an existing tool and its workflow is the most desired modus operandi, the solution presented in this thesis could be integrated there. The desire for using an existing tool might not necessarily come as a result of that tool being objectively better, but an element of resistance to change, as described by Watson (1971), might be involved. The incident management tool chosen, should record any changes in a history for an incident. This makes it easier for other incident handlers to know what has been done before, as well as make it easier to audit the incident handling.

7.4 Summary

This chapter has presented the views of experienced incident handlers on the proposed interface and format, as well as how this would fit into their workflow. The format was found to be mostly complete, but fields such as *next update*, *expected time of correction*, and *channel* were requested. With regard to incident exchange, the participants pointed out that it would be different thresholds for notifying customers depending on the severity of the incident. In some cases, upper management would have to approve the message. Honesty was said to be important when informing customers about what has happened. One participant identified the need to reply to the sender in order to request more information, and suggested a shared comment section in order not overwhelm the sender with requests. With regard to workflow, the prototype did not fit completely with the participants' current workflow, but they said the solution would be useful if integrated with their current tooling.

Chapter 8

Discussion

8.1 Incident Representation

Cichonski et al. (2012) state that each team must choose their own list of required data elements, based on factors like team model, team structure and how the team defines an incident. In order for the concept outlined in this document to work, it is necessary to agree on a definition of an incident and its data elements. This would allow for easier development and deployment of new systems as well as increase interoperability, as one would not have to conform to multiple different definitions and data sets, but rely on one base format. If an organization needs a different representation internally, this will still be possible as long as it is feasible to translate this representation into the common representation. E.g., if one internally wants to represent time and date as two separate fields, this is possible as these can be combined into one time and date field. Similarly, an organization could represent a system by its nickname, and automatically substitute this for a systematic representation before sending a notification.

8.1.1 One or Multiple Formats

In order to ensure that every party of the cloud supply chain is able to understand the information sent, the project conducted by Frøystad (2014) identified the IODEF as a suitable common format. However, there are significant potential problems, including the complexity of the format, making it difficult to understand for humans, and the amount of extensions that would need to be created in order for the format to be fully usable for automatic handling of incidents. Another potential problem is the amount of data the format would have to represent. As can be seen in table 3.1, the number of data points required becomes quite large when considering just a few incident reporting guidelines or requirements. Combining this information with the findings of Floodeen et al. (2013), that incident handlers require all information of a system or a standard and do not enter only the necessary information based on situation, raises the problem of incident handlers wasting precious time. This could result in this solution reducing productivity, which would reduce the chances of such a system being adopted by providers.

Schneier (2014) claims that "*Incidents aren't standardized; they're all different.*" This raises the question of whether it is actually possible to represent every incident in one format, or if it is a better approach to allow multiple formats and thus allow for specialization. By using only one format that is able to represent everything, one could, theoretically, know that all conforming implementations would be able to understand any incident received and be able to handle it automatically if so desired. On the other hand, the format would be very complex, as it needs to include mechanisms to represent all possible relevant information for any incident imaginable. This would lead to increased costs in implementing the solution, as well as increasing the cost for adding new types of incidents due to the high degree of coupling.

At the other end of the extreme scale, the option can be found to only transfer text messages between parties in the cloud supply chain. This would reduce the solution to be a secure "email" system, and thus in accordance with how incident information is mostly exchanged today (Frøystad, 2014). The ability to provide free text would allow the parties to exchange any information required, but they would have to agree on a special formatting for the text if planning to support automatic handling of incidents. This would make it easier to implement the solution, and thus reduce initial adoption costs. The flexibility of being able to input any information in any formatting, makes it easy to include new types of information as well as only include what is necessary for the situation at hand. Notable drawbacks of this approach include fragmentation in message formatting, leading to potential difficulties for humans in interpreting or encoding incident information. The lack of a common way of formatting incidents would also lead to overhead in exchanging information with different parties, as the very same incident information could potentially need to be encoded in n different formats for n different recipients. Based on these considerations, it is likely that such an approach would cause more difficulties than it solves.

A middle ground could be to have a small base format, with the ability to represent the most common information in a simple way and providing a structured way of attaching other incident formats such as IODEF, eCrime, STIX or CybOX. In addition, the format could support *custom fields*, which would allow the provider and the subscriber to agree upon extra information to be included in the base format without altering its base structure. This would allow for two levels of implementation of the solution discussed in this document, as well as support incremental development of the highest level. *Level 1* would only implement the base format, and thus only be suitable for incident handling where there are humans acting on the incident reports. Some degree of automation could be supported in basic situations if *custom fields* were used intelligently. *Level 2* would implement different attachment formats, and thus be able to support more automation of incident handling. This allows for reuse of existing incident formats, specialization of the formats, incremental implementation of formats as needed, flexibility to support newer formats, and the possibility to also exchange evidence in the case of a forensic investigation on behalf of another party in the cloud supply chain.

The middle ground appears to offer the most beneficial approach, especially since it allows multiple levels of implementation which will allow businesses and organizations to start with a simple solution that can grow with their needs. The matter of forward compatibility is also an important factor in preferring this approach.

8.1.2 Propagate Incidents

Two approaches to representing propagated incidents were considered. The first approach embeds the parent incident in an unchanged manner, allowing all recipients of a derived incident - an incident created based on information from another incident - to access all of the information received by an earlier link in the supply chain. At first glance, this seems like a valid approach and one that could result in better cooperation, faster response time and better incident handling as a result of access to more and better information. There are problems with the approach, though. If all information could be passed on to entities further down in the chain by a receiver, this would result in loss of control for the entity sending the incident as it would not be know who receives the incident information. Given the potentially sensitive nature of information included in or related to security incidents, there is a real possibility this would result in the system not being used or only superficial information being shared.

In order to allow the incident sender to be in control of his incidents, a better approach is to only reference the parent incident when propagating down to a subscriber of a subscriber. In this way, only relevant information would propagate directly, through the actions of an entity, while there would still be a hard link to follow in order to establish exactly what happen during the incident handling. This could, e.g., be useful when auditing a provider or during criminal investigations. Put simply, only those with a direct connection to a provider will be able to access the incident information sent by this provider. Subscribers of this provider may propagate incidents to their own subscribers, but not include the incident they received, only a reference to the incident notification. This way the original sender of the incident is still in control of who accesses the information, and at the same time it is possible to maintain a complete chain of custody and hopefully protect its integrity. Figure 8.1 shows the different types of actors that could provide incident information or receive incident information.

8.2 Interface Considerations

This section discusses the choices and challenges faced with when creating the interface and the current proposal for exchanging incident information between parties in a cloud delivery chain.

8.2.1 Probability of incident

The solution described earlier in this document, initially included a property to indicate the probability of an incident occurring. This was meant to assist customers with less experience with security - a pointer to where they would need to focus some of their efforts. After some consideration, it was dropped as it could be a huge Pandoras box. Imagine a court procedure where the probability estimate from the provider is upheld as an absolute truth, and the provider is found to be liable for a something which is really caused by incompetence or carelessness on the customer's part. Another related and relevant aspect is that the European Union's new Data Protection Regulative leaves no room for incompetent customers, as the demands for risk assessment and procedures are the same for small

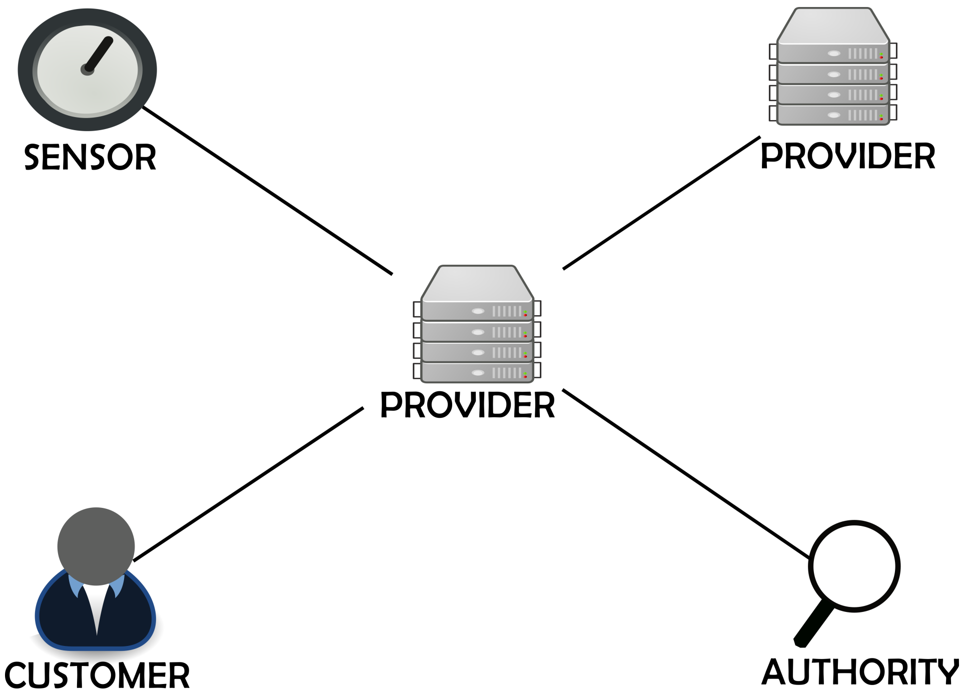


Figure 8.1: The different types of relationships that an organization might have with regard to security incident information exchange

companies with only one employee and large multinational companies. The fines for not conforming to the regulation are substantial enough to motivate both small and large companies. The fine is whichever is more of the following: 2 % of total global turnover or € 1 000 000 (Tañà, 2013). A possibility could be for the probability to be fetched from world wide statistics, identical for all providers. This would, however, not justify the property to be included in the API, but rather calls for a table of general information to which the customer could be pointed should he need to consult probabilities.

Another possibility could be to generate the probability property based on architecture, competence, operations, experience, etc. of the individual data centers, but this could affect the company's marketing. Due to affecting marketing, it is considered unlikely that such calculation of probabilities would contribute to increased security or better quality information and decisions. It is far more likely that the probability values would be incorrect, as they are only to be considered estimates and advice, and there would be few or no repercussions for lying about the probabilities. The reason a provider would lie about the probabilities could be to keep customers, look good in marketing and comparisons or simply to not admit that certain parts of his system has serious problems. Another problem with this approach would be to come up with a scoring system, agree on it and figure out a way of validating the system and the resulting scores.

Due to the above issues and reflections, and being convinced that the inclusion of probabilities would not be of any real assistance for neither the customer nor the provider, the probability property was not included as part of an incident type.

8.2.2 Notification of withheld information

Initially the draft included a clause of a notification being sent to the the customer if the provider did not notify him about an incident within a given amount of time. The idea was to provide the customer with necessary information as soon as possible, even though the provider would not be ready to hand out details at that moment in time. A prominent problem with that approach was that it is the provider who would be responsible for notifying the customer that he is not willing to share some incident information at that given time. This is likely to be unrealistic, as the provider is withholding information for a reason. Still, it is important to find mechanisms to solve these situations. The solution is, however, more likely to be found in SLAs than in technical mechanisms likely to be disabled or not implemented by the provider. The provider could make a trigger type for “withholding incident information” available to customers having clauses about this in their contracts or SLAs. The provider would nonetheless have to log any and every incident that occurs, even if he does not notify the customer. Thus he could notify those who do not have the relevant clauses in their contracts when the incident is resolved.

8.2.3 Threshold

Section 4.1.2 presents the concept and payload of *Trigger Type*s. In a prior revision of the interface, *Trigger Type*s also included an option to choose the type of threshold it operated on. The rationale was to not limit users in the way they could trigger incident notifications, and therefore allow for multiple types of numbers, strings and even regular expressions. The current version of the specification, requires the *threshold* to be of type *FLOAT*. One of the reasons for this change, is that thresholds should be easily identifiable and computable, and therefore also easily understood. Numbers are more intuitive and easy to compare than strings and regular expressions. Furthermore, the trigger types that can be achieved with strings and regular expressions might not strictly be triggers, but rather filters. It was therefore decided to only allow thresholds of type *FLOAT*, and leave filtering to be implemented in the backend logic of each implementation where such functionality is desired.

8.3 Notification

Different approaches could be taken with regard to when to notify the subscriber. One approach is to leave it all to be decided by the mutual agreement between subscriber and provider, and let it be managed by triggers and incident types. This would leave it up to the provider and the subscriber to make sure that they exchange the necessary information to fulfill the relevant laws and provide sufficient data for information exchange to be of use.

Another approach could be to notify about everything, but flag notifications that fits the defined triggers and incidents types. This would include notifications with a core message like: “We have been breached, but your data was not compromised.”. While this could theoretically result in everyone having the necessary information, and being able to act upon it, it might just as well cause the subscriber to be flooded with information he does not need. It might also cause performance issues at the provider, as it would be a quite large amount of data to send.

A hybrid, combining the two extremes, would allow the subscriber to mostly receive information he has asked for by defining triggers while still allowing the provider to fulfill his legal obligations to notify about certain specific cases, such as breaches in systems containing personal information. The specifics of when and in which cases the provider should override the defined preferences and triggers defined by the subscriber, is likely to vary from jurisdiction to jurisdiction and will thus need local adjustments. This will not, however, affect the interface or the exchange format, only the back-end.

Due to the substantial fines defined by the Data Protection Regulation, service providers are given an incentive to ensure accurate and timely notification about breaches relating to personal information. Figure 8.2 shows an example of the relationship between service providers and services used, and thus provides an example of the amount of unneeded or unwanted incident information a subscriber could receive if the defined triggers are not taken into account. Therefore, it is expected to be better to use the hybrid approach, where the subscriber defines what he wants to be notified about and the provider additionally pushes all information required by law to the subscriber by using mandatory notifications that could be defined for each subscriber, thus being able to comply with different laws.

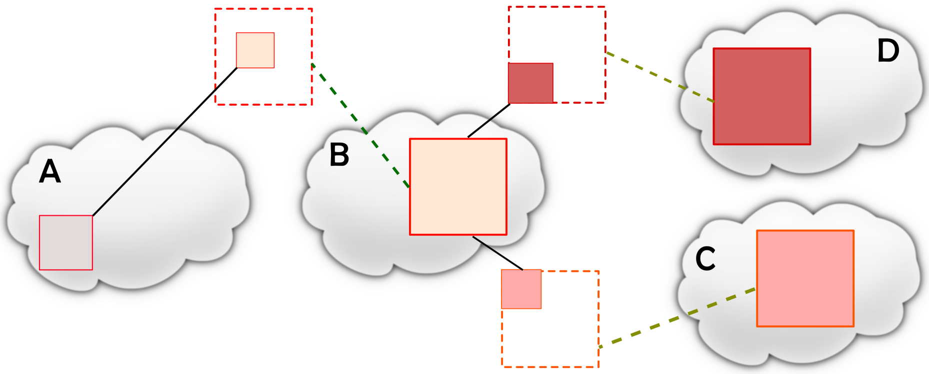


Figure 8.2: Data flow in supply chain. Each cloud represents a service provider. Each colored square represents the services used by the subscriber. The colored square inside each stippled square, represents the data or parts of the services actually used by the subscriber. The figure is based on a sketch by Martin Gilje Jaatun

8.3.1 Language

Given the globalization of computing, the provider and the subscriber could have very different native languages. This requires some consideration into how to handle the language barrier. There are three feasible options, each with apparent strengths and weaknesses. The first option is to leave it to the provider to decide in which language to provide incident notifications. This would lower the barrier for the provider, as he would often be working in his native language and would be able to express himself easily and efficiently. The most apparent weakness is that the provider's native language is not necessarily understandable by the subscriber, which would either have to hire an incident handler who understands the provider's language, rely on mechanical translation or just discard the notifications. Furthermore it would be much harder to audit a cloud supply chain if the auditor would have to rely on translators or involve multiple auditors.

The second option goes to the other extreme, standardizing on one language. At the moment, English is the predominant language in the technology industry, though this is likely to change as non-western countries rise in the technology world. A solution to the different language challenge could be to require all incident information to be written in English. The most prominent strength of this approach is that all information is recorded in the same language, theoretically giving the provider and subscriber equal terms with regard to language. Another strength is that this is likely to be accepted by the dominant providers of today, as these are mostly US based, e.g., Amazon, Microsoft and Google. Another positive aspect of this approach is that it would align the language used for security incident notifications with the recommended language used in other security critical communication, such as aviation and maritime. Important drawbacks include varying proficiency in English, English not being first nor second language in large parts of the world and the possibility of some countries opposing the idea of using English as a standardized language. It is however important to note that much of todays technology is English centered.

The final possibility is to leave the matter of which language to use to be decided by the provider and the subscriber when negotiating SLAs and contracts for the purchase. This would theoretically result in an agreement that suits both parties, but this is not always likely to be the case. The large providers, such as Amazon, Microsoft and Google, are likely to offer a small set of languages, typically English and Chinese, and demand that anyone using their services needs to agree to their use of those particular languages. The difference in size, turnover and negotiating position makes this less of an agreement and more of an requirement left to be decided by the provider. Another problem with this approach could be duplication of effort if provider A provides incident information in English to subscriber B, in Chinese to subscriber C and in French to subscriber D. This would potentially require a large amount of time being spent on translating incident notifications into other languages, introducing delays and potentially new problems due to inaccurate translation. This duplication of effort grows exponentially as the next link in the chain needs to provide multiple languages to their subscribers.

Seen from a western perspective, it is preferable to standardize all incident notifications to use the English language. However, due to English not being fully adopted all over the world, it would probably harm the adoption and use of such a notification system. Therefore, the incident notification language must be decided when initiating a contract between provider and subscriber. This will, most likely, lead to English being a prominent language in areas like the US and Europe, as well as other countries which aim to sell services to areas where English is a common first, second or third language.

8.3.2 Impact

The current impact field, as described in section 5.6, is represented by a float, which allows the incident handler to use a scale of impact severity. While this might be useful, it begs the question of how the scale is to be understood. Depending on the experience with incident handling and how services are used, different organizations would probably place the same incident at different values along the impact scale. Depending on how the scale is received among seasoned incident handlers, it should be considered to include the possibility of adding a description to the impact field for each incident. Depending on the implementation of the backend, this could either be predefined descriptions of a predefined impact scale or a text field where the incident handler could write about how this impacts the organization.

8.3.3 Attachments

As mentioned in section 8.1.1 and 5.12, attachments could increase the flexibility of incident information exchange while still maintaining a simple common base format. This will also contribute to reducing the coupling of information in incident reporting compared to having one large common format to represent everything. A potential problem is that the participants in the cloud supply chain would need to know how to generate and interpret the different formats attached. One way of handling this situation is to allow the parties to generate and attach any format they want, and just suppose the receiver is able to interpret the format based on file endings or content of the file. This will allow for a high degree of flexibility and forward compatibility, but at the same time introducing ambiguity in how attachments are to be interpreted and possibly cause problems by effectively requiring a large amount of logic in every cloud supply chain participant in order to support all necessary formats.

Another approach is to treat the attachments only as attachments that can be downloaded and uploaded, generated and interpreted with other existing tools. This is likely to be the way attachments are treated in the case of *Level 1* support for this solution. In order to support automation, this solution would be very similar to the one above, as the back-end would need to decide which file format is sent and to which tool to send it.

Finally, the option of having a defined set of incident formats, agreed upon between the provider and subscriber through SLAs, provides some interesting values. The interpretation of format is explicit, making it easier to distinguish between how to interpret different formats, many of which ends with *.xml* or *.zip*, as each attachment is accompanied with a value stating the type of attachment. Additionally, it gives flexibility to the provider and the subscriber, as they can identify the formats most suitable for their use case and apply those. It is likely that existing and standardized formats will be used in most situations, because of the investment required to develop a new format. Another possibility is that smaller and more specialized formats surfaces, where each format represents one and only one type of incidents. This option provides the flexibility of the first option, support for the second and at the same time mitigating the ambiguity problems of the first option. It is therefore considered to be the better approach for handling attachment of incident information.

Initially, attachments were planned to be embedded in the incident format, rather than be exchanged as a separate step. The reasoning behind this choice was twofold:

1. The entire information exchange could be done in one operation, making it easier to ensure the integrity of the notification by the simple fact that all or nothing is received
2. All load on the provider's servers could be controlled by the sender. If a large provider like Amazon sent a notification with a somewhat large attachment, it could be beneficial to do some load balancing in when different subscribers received the attachment.

The potential for large attachments made this approach need reconsideration. While many, perhaps even most, attachments are not very large and would be easy to exchange as embedded formats, there might be attachments of multiple GB or even TB. Given that not all receivers need all attachments, it is likely that by embedding, and thus increasing the file size during transfer, the attachments, the load on both sender and receiver is unnecessarily high. This gives the following possibilities:

1. Embed attachments - do not consider problems with large files
2. Send attachments as type and link - subscriber fetches the attachments immediately
3. Send attachments as type and link - subscriber fetches attachments when needed

By using option 3, load is reduced on both provider and subscriber, while at the same time reducing the size of the payload when sending notifications. This would require the provider to have strong access control on files and attachments, but this should already be the case. A potential challenge is waiting period for the incident handler before opening files, as they would need to be transferred from the provider, when needing them. The alternative would be even worse, though, as that would mean the incident handler would not be notified about the incident before all attachments were downloaded or uploaded, thus delaying the response.

8.3.4 Custom Fields

As mentioned in section 5.13, custom fields could allow providers and subscribers to agree on extra information to include in the base format without changing its structure. This is an easy way of exchanging a few extra values that the subscriber wants, but without the overhead of a large incident representation format. There are multiple ways custom fields could be implemented, including allowing any values to be included in any format, allowing only a predefined set of basic data values, and providing data building blocks that allow representation of anything in a structured way.

Allowing any values in any format would provided a high degree of flexibility. It would, however, also introduce problems such as *custom fields* being abused when *attachments* should be used instead, and possibly also introduce problems with having to have interpretation rules for each cloud supply chain relation as the format used to encode custom information would differ.

By allowing only basic data types, such as *String*, *Integer*, and *Boolean*, use of *custom fields* where *attachments* should have been used would become less likely, but still possible through abusing the *String* value field. Explicitly stating the type of the value in the field would make it easier for the subscriber to interpret the value. A potential problem with this approach is the reduced flexibility, though it might be argued that *attachments* should be used for anything more complex than including simple values.

Using building blocks is an extension of the basic data types. By allowing JSON to be included as a data type, it is possible to incrementally create blocks usable for representing incident information. Such blocks could be port lists, IP-address, URLs, etc. This would allow organizations not implementing attachment formats to formally handle information in a predefined manner.

While the value proposition of using building blocks is interesting, this will all be representable by attachment formats and should therefore not have high priority, but could be added at a later time. Therefore, custom fields will be able to represent basic data types, and JSON since that is needed to be able to implement building blocks later. The result is that any data can be represented, but not in any format, and preferably with basic data types, allowing for easy interpretation.

Custom fields are related to incident types, thus incident types can be viewed as a form of template for incidents. This reduces the mental burden of having to browse through large amounts of unrelated fields, but only the fields needed for the incident type in question are made available to the incident handler.

8.4 Adoption

In order for this solution to be useful, it needs to be adopted by businesses and CSPs. Given how most businesses strive to improve their financial results, it is likely that for the system to be adopted, one of the following criteria must be fulfilled:

- Use of the solution results in reduced costs or increased revenue - directly or indirectly

  > ...companies need to know what they'll get back from the USG [US Government]. The commercial sector looks for 'return on investment,' and this is one area where they will seek a clear response. - Shawn Henry (SANS Institute, 2015)

- Actors are required by law to use a system similar to this solution

8.4.1 Reduced costs

A *Level 1* implementation, that is exchange of security incident information without any automation in incident handling, is unlikely to result in significantly reduced costs, if reduced costs at all. However, the solution has been designed with implementation cost in mind, so the cost of adopting the solution should be quite low. The CSP could integrate the interface with their existing incident management tool, and use an incident format adapter or translator to convert between their local format and the format used by the solution outlined in this document. If the solution was separated into microservices, one could further decrease the implementation cost by offering them as open source.

The incremental nature of the solution allows implementer to gradually introduce more formats and also automation. As the implementation progresses into a *Level 2* implementation, with an increasing amount of automation, the reduced costs are expected to become noticeable. Metzger et al. (2011) claim that more than 85 % of abuse cases can be partly or fully automated, which in turn would free up resources allowing for reduced costs or for the incident handlers and the security team to focus more energy on improving security and handling the more difficult cases where full automation is not desired. Some incidents might require inspection and decisions to be made by a human before any information could be passed on to subscribers.

8.4.2 Increased revenue

This solution is unlikely to contribute directly to a higher revenue stream, but might contribute indirectly. If a CSP, or any other organization adopting this solution, is diligent in sharing information about incidents, this could contribute to building an image of trustworthiness and professionalism. Such an image could in turn result in more customers, and thus increased revenue. This would, however, require the organization to be careful to explain incidents and their process in an understandable manner, so the customer is reassured rather than unnecessarily alarmed. CSPs could even offer incident management dashboards for customers, so the customer would not need to administer their own instance of such a system. If the organization fails to appear trustworthy, it is likely to lose customers which in turn would affect the organization's revenue. It is therefore important to have competent incident handlers operating the system and any automation put in place, to ensure quality in both incident handling and communication.

8.4.3 Laws

Laws are powerful incentives for changing behaviors in entire industries within a country. When large unions, like the United States or the European Union, introduces quite similar laws, this affects the entire western world and also, to some degree, the rest of the world (Greenleaf, 2012).

Laws are not only an incentive, but sometimes also a hindrance or at least an obstacle. The difference between laws covering personally identifiable information (PII), could complicate information exchange. The organization wishing to send incident information, needs to make sure that no PII is included. It has been claimed that information disclosure has the biggest potential contributor to CSIRT liability (Brown et al., 2003, p. 57). To mitigate the fear of being sued for sharing incident information, the US introduced a bill to protect companies that shares information with the government from liability (Osborne, 2015).

8.4.4 Trust

In this document, it has been assumed that trust can be moved from a level where it exists between two humans, to a level where it exists between two organizations. This is not necessarily realistic. Traditionally, incident information has mostly been shared directly between people with a direct trust relationship, i.e., who know each other personally. Cloud providers need to trust each other on an organizational level, where one can agree to share information that relates to the services that the subscriber is using in the provider's infrastructure. As Henry states below, without trust, there will be reluctance towards sharing.

> The concerns of US companies, on the heels of Snowden and other revelations, are completely understandable. Phyllis's assessment is absolutely correct. Until companies can trust the USG [US Government], they will be reluctant to share. - Shawn Henry (SANS Institute, 2015)

Many other problems can be defined as sub-problems of trust. Legal worries about sharing incident information comes from the fear of legal action as a result of information sharing, but this can be traced back to not trusting how the recipient uses the received information. Public relations worries related to public perception of the company being damaged as a result of sharing information can also be traced back to lack of trust in how the recipient uses the information received. While this solution does not automatically make service providers trust each other, it could make a way for communication to happen in a flexible fashion between distinct and already known organizations, through a secure channel and in an environment controlled by the sender of incident information. It is expected that most non-technical problems, such as trust and who is allowed to forward what information to whom, can be solved by adding terms to the respective SLAs and possibly adopt and enforce the TLP (US-CERT, 2015a)(European Union Agency for Network and Information Security (ENISA), 2015). Use of sanctions for breach of contracts and trust might also assist in making it easier for organizations to share incident information, as they would know that any misbehavior by the other party would affect their bottom line.

8.4.5 Security

A serious challenge of using this approach, as well as others like it, is the fact that it requires to be connected to the internet. Information stored includes how the organization's and their providers' systems have been compromised as well as how such cases were mitigated. If this information was to be accessed by malicious actors, large amounts of potentially harmful information would be in the wrong hands. Such information could potentially be used to take control over an entire cloud supply chain, if a malicious actor gains access to one link and is able to use information shared by the next link to take control over that one as well. This makes it imperative that such a solution is serious about security at every link, otherwise it is unlikely that actors would share any helpful details about an incident.

Another challenge is if a malicious actor gains access to one instance of the solution and sabotages the chain by spreading false incident notifications. Provided that such false incident notifications are carefully crafted, it might be possible for the malicious actor to cause harm to the CSPs reputation and, in some cases, even facilitate intrusion into entities that rely upon the compromised actor. This shows both the importance of good security, authentication and authorization routines, as well as the importance for competent incident handlers or advanced automation, capable of assessing the information received from other organizations.

8.5 Validation of Approach

Cusick and Ma (2010) describe a solution similar to the one proposed here, but for internal use in the organization. Users might subscribe to be notified when events are created or changed, which has lead to improved communication around the incidents. The authors state that this feature alone made it worth their while to create a new process and implement new tooling. The solution proposed in this document takes this one step further, and allows other entities to be notified just as easily as the human subscribers. It is not known how simple it would be to integrate this solution with the Sharepoint based solution by Cusick and Ma, but theoretically this is possible without changing their working system. Cusick and Ma had a different experience from that described by Floodeen et al. (2013) who found that incident handlers entered any information requested by the system. Cusick and Ma found that engineers entered the minimum amount of information required and often just closed the ticket when it was handled without any information about steps taken to solve the incident. This problem will not be solved by the solution proposed in this document; if anything it will become more apparent and pressing. Depending on the relationship between two parties exchanging incident information, it might be alarming for one party to never receive any information other than “we have an incident - it was solved”. This might lead the consuming organization to wonder if the incident has really been handled at all, and thus affect the trust between the two organizations. The solution to the problem is likely to be a combination of cultural and a question of resources. If the CSIRT has a culture for not storing information on how an incident was solved, or only stores such information in a group internal database, this is behavior that would need to be examined and perhaps acted upon. It could also be that the CSIRT has more work to do than resources allow, and providing information on how the incidents were solved are therefore not prioritized. This leads to large amounts of knowledge being confined to the actual incident handlers, which is likely to cause added costs when an incident handler leaves the organization and is replaced by a new incident handler without this knowledge. It is therefore likely that organizations will save money and improve performance by requiring incident handlers to make some notes on the steps taken to handle an incident.

Ahmad et al. (2012) describe a case where a large institution, consisting of several physical locations, need to collaborate on incident response. Risk assessments by the security department are stored in their local database, while incidents should be stored in a company wide database. Incident reports are received to a help desk solution, and thereafter entered into the internal incident management tool. There are several ways in which the solution proposed in this document could assist this organization. First and foremost, the interface could allow to simplify the transfer of incidents from the help desk to the incident management tool by either fully automating the process or making it as simple as for the help desk operator to click a button. There might also be a possibility for the solution to be utilized to empower communication between the local risk database and the company wide incident database, but the assistance of the solution is expected to be limited here due to risks having a wider application area than the actual incidents considered in the current solution.

Hove et al. (2014) do not reveal many details surrounding the procedures of each of the examined companies, but it is clear that organization A receives notifications from their supplier and organization B receives an updated list of occurred incidents each month.

It is likely that the solution presented here could be of assistance for organization A in receiving incidents from their provider. In the case of organization B, the solution would probably not be of use given that they only want to receive incident information once per month. It could, however, be used, so that organization B receives incident information the during a month and only examines the list at a specific time. Thus, the solution could be used for the use case, but does not provide any added benefits.

Metzger et al. (2011) describe a system that notifies subscribers about malware and other unwanted programs running on their systems, and is able to handle 85 % of all incidents in an automated manner - fully or partly. While the solution specified in this document does not provide any assistance on the detection part, it might be of assistance in notifying subscribers who could then also be offered more fine grained control over which notifications they wish to receive. The proposed solution could also be used for sensors to enter incidents into the central incident database. The solution would not, however, replace email and phone as reporting channels for all reporters. For cooperating organizations, the solution could replace such means of communication, but for single individual humans a simpler way of reporting would be needed, such as a web interface or email and phone as described in the article.

Cichonski et al. (2012, p. 9) state that *The incident response team should discuss information sharing with the organizations public affairs office, legal department, and management before an incident occurs to establish policies and procedures regarding information sharing.* The solution proposed in this document facilitates such decisions to be taken before incidents occur, as subscribers are able to subscribe to incident types made available to them by the CSP. Thus the CSP needs to have decided beforehand which incident types each subscriber is allowed to subscribe to. In addition, each organization is free to decide how to implement the backend and can thus require a man in the loop, allowing for a second screening of incidents before they are sent to subscribers.

> Even the smallest organizations need to be able to share incident information with peers and partners in order to deal with many incidents effectively. Organizations should perform such information sharing throughout the incident response life cycle and not wait until an incident has been fully resolved before sharing details of it with others. - Cichonski et al. (2012, p. 48)

The above citation, makes it clear that all organizations need to share and receive incident information in order to effectively handle many incidents. An important consideration taken when creating the proposed solution of this document, was that it should be feasible for even small organizations to implement, utilize and receive value. The two level implementation might not be perfect, but it allows for smaller organizations to reap a subset of the potential benefits, while larger organizations implementing full automation reaps the full set of potential benefits.

> Organizations should attempt to automate as much of the information sharing process as possible to make cross-organizational coordination efficient and cost effective. In reality, it will not be possible to fully automate the sharing of all incident information, nor will it be desirable due to security and trust considerations. Organizations should attempt to achieve a balance of automated

information sharing overlaid with human-centric processes for managing the information flow. - Cichonski et al. (2012, p. 48)

The quote above states the importance of automating as much of the incident information sharing process as possible. The solution proposed in this document was created to thrive in this cross section, as this would all be left for the specific implementation of the backend. If all incident information exchange was automatable, a fully automatic backend could be applied. If no incident information exchange could be automated, a full manual backend could be applied. If some information exchange could be automated, a hybrid backend could be applied. There is no limitation in the solution itself on how the backend would handle incident information exchange.

Cichonski et al. (2012, p. 48, 50) state the need to identify the types of information to be shared and store these in a formal data dictionary. They also point out the importance of identifying which partners should be allowed to receive different kinds of information. While the solution proposed in this document does not provided insight into the relationship between data entities, the incident types are a way of cataloging the possible incidents that could occur and that could be shared, thus serving as a dictionary of possible incidents. If an internal network of sensors and detectors using the proposed solution exists, each of these would be a provider in the main instance and the organization would thus also have insight into the internal entities and their relationships. The use of incident types and making each of these available to only the correct subset of potential subscribers would allow the CSP to control which organizations receive which information.

Doelitzscher et al. (2012) present a system for interconnected agents to detect and communicate occurring incidents. The solution proposed in this document could be used for the communication between the agents, allowing an aggregating node to receive information from multiple sensor agents, process the received information and pass it on to the higher link in the chain. It is not, however, clear that this would provide any added benefits over the approach already taken by the authors. One potential benefit could be that if the proposed solution, or some form of it, hypothetically becomes widely adopted and standardized, it would be easier to plug in new sensors and any other actors into the interconnected web of actors.

As mentioned, both in the problem description and in the introduction, this thesis is based on a call for research by Grobauer and Schreck (2010). The solution proposed in this thesis seeks to solve some of the problems examined in their paper. This solution seeks to give the subscriber access to CSP controlled event sources and vulnerability information, while still allowing the CSP to be in control of information sharing. This is done by providing an interface to access and receive the relevant data allowed by the provider. The CSP is provided with the means to inform the subscriber about the relevant data sources, and the traceability makes it easier to gather evidence if one has the right to do so. Like Grobauer and Schreck (2010), it is acknowledged in this thesis that contracts and SLAs needs to be used in order to introduce and enforce such a system.

8.5.1 Interviews from prestudy

The prestudy by Frøystad (2014) included interviews with subject matter experts, providing the basis upon which this part of the validation is based.

Subject 1 claims that the organization has little or no dialog with other participants in the delivery chain, even though there is some form of cooperation with the developers of the underlying platforms. They also employ another company which is tasked with validating the organizations work in relation to security. When breaches are detected at a customer, communication channels are opened. It is important to be able to trace incidents and thus be able to prosecute any perpetrator. All decisions on how to handle incidents are made during contract negotiations, and the contract is a strict guideline. It is important for the organization to have a human make the decision on whether to notify customers about incidents or not.

The solution proposed in this document takes all of these concerns and restrictions into consideration. The organization might receive information about the underlying platform components by subscribing to the instance run by the developers or producers. The company tasked with auditing or validating their work could subscribe to the organization's handling of incidents, and thus be able to observe the process more closely. There might, however, be legal challenges with allowing another company to have full access, so some screening of information and access would probably be needed. The requirement to be able to trace incidents and also any perpetrator made traceability an important element of the incident exchange format, making it possible for someone with the right access and authority to trace an incident back to its origin and receive the needed information to e.g. take legal actions. The solution works between two entities, and thus supports, and relies on, the concept of contract negotiation and mutual agreement. Since the solution does not make any implications on the type of backend to be used, it fully supports the organization's need to have a human make the decision on whether to notify or not.

Subject 2 states that it is important to be able to choose which incidents one receives, as one does not necessarily want all available information. He envisions a future where cloud service brokers handle legal challenges and handle most incidents by relying on each other. The solution proposed in this document supports both sentiments. It is possible to only receive selected incidents by only subscribing to selected incident types as well as defining the relevant triggers. The solution could serve as a means for cloud service brokers to exchange security incident information. The solution will, however, not be able to assist in handling the legal matters, as these would need to be solved at another level before initiating incident exchange.

Subject 3 describes the security incident information exchange to be too poor, as a result of security requirements and demands not being a central part of the contract negotiations. They do exchange incidents with other CERTs by means of structured email and to some degree with the police by means of fax. He claims CERT to CERT trust is established as a CERT maintains a good track record over some years. A area that could be improved is automation, which he says would require a structured format for incidents. He is concerned about the impact on established processes and points out that it would have to be easy to share information.

The solution presented in this document aims to improve the communication between parties, with regard to security incident information, and recognizes that while a technical solution facilitates such exchange and communication, the exchange needs to be initiated by the contracts and SLAs between two parties. The solution might provide a more structured way of sending incident information than email or fax, while at the same time guarantee delivery and provide easy validation of sender and content. The two level implementation and the custom fields, allow for the solution to be introduced in their existing tool in a transparent way. Automation could then gradually be added where needed, without necessarily requiring a paradigm shift within the CERT. Exchanging incident information could be as simple as pressing a button.

8.6 Comparison to other solutions

There are other solutions solving similar or related problems to the one proposed solved in this document. In this section, some existing systems and solutions are compared to the one proposed here.

8.6.1 Trusted Automated eXchange of Indicator Information

Trusted Automated eXchange of Indicator Information (TAXII), as described by Connolly et al. (2014), is quite similar to the solution proposed in this document with regard to how it can be integrated with existing systems. Like this solution, TAXII only provides specifications, definitions, protocols and a exchange format - specifically STIX. Its goal is to enable sharing of cyber threats in a secure and automated manner. While there are similarities, there are also differences. One such difference is the scope of the problem the solution aims to solve. TAXII, through its connection with STIX has a scope extending to all threat information, while the solution presented in this document, only focuses on actual security incidents. TAXII and this solution both support a number of different compositions for information exchange, but TAXII has perhaps better support for the hub version. It seems possible to implement this solution as an extension to TAXII, but that would defy one of its designing goals of simplicity. TAXII is a large and complex collection of specifications and protocols, while the solution proposed in this document is intentionally kept simple to allow for easier adoption. That being said, both TAXII and STIX have large, competent organizations backing it and there is a real possibility that the systems and specifications needs to be this complex in order to support their defined use cases.

8.6.2 Soltra Edge

Soltra Solutions LLC (2015) creates Soltra Edge, a system utilizing TAXII and STIX to share threats and cyber intelligence with the goal of increasing resilience of critical sector entities. The approach is to create trusted hubs and environments in which companies can share cyber threat intelligence. A company or an organization could be invited to join a circle of trust or create one themselves. The system is a complete software platform that can be installed and configured at the user's premises.

The main differences between Soltra Edge and the solution proposed in this document, can be summarized in scope and intrusion. Soltra Edge, facilitating sharing of any and all types of threats, has a much larger scope than this solution, which only focuses on actual incidents between actors. Soltra Edge appears to try to solve the entire problem of information sharing and threats in general, while this solution focuses on one small aspect.

The level of intrusion on existing processes and systems is also a significant difference between the two solutions. Soltra Edge seems to be a replacement, or an additional system, to any preexisting systems in an organization, which makes it necessary to adjust processes, change used systems and provide new training. That being said, this would not be a problem for organizations new to incident response and threat handling, as those would probably benefit from Soltra Edge being a complete platform. The solution in this document has very little impact on processes and preexisting software, since it is an interface to be connected to any software system and most any preexisting processes. Processes might still need some adjustments in order to handle the new possibility of sharing information in a more simplified manner. The circles of trust are another example of difference in intrusion. While it appears that circles of trust can be established between only two participants within Soltra Edge, it is worth mentioning the potential overhead of establishing contracts and trust between multiple participants at the same time. The solution in this document works around this by requiring sharing to be one-to-one, and thus also contracts be on a one-to-one level. It is, however, still possible to create hubs or circles of trust by establishing a trusted broker with which all participants share information or one could simply establish connections between all participants. It is also possible to integrate this solution into Soltra Edge if so should be desirable.

8.6.3 Malware Information Sharing Platform

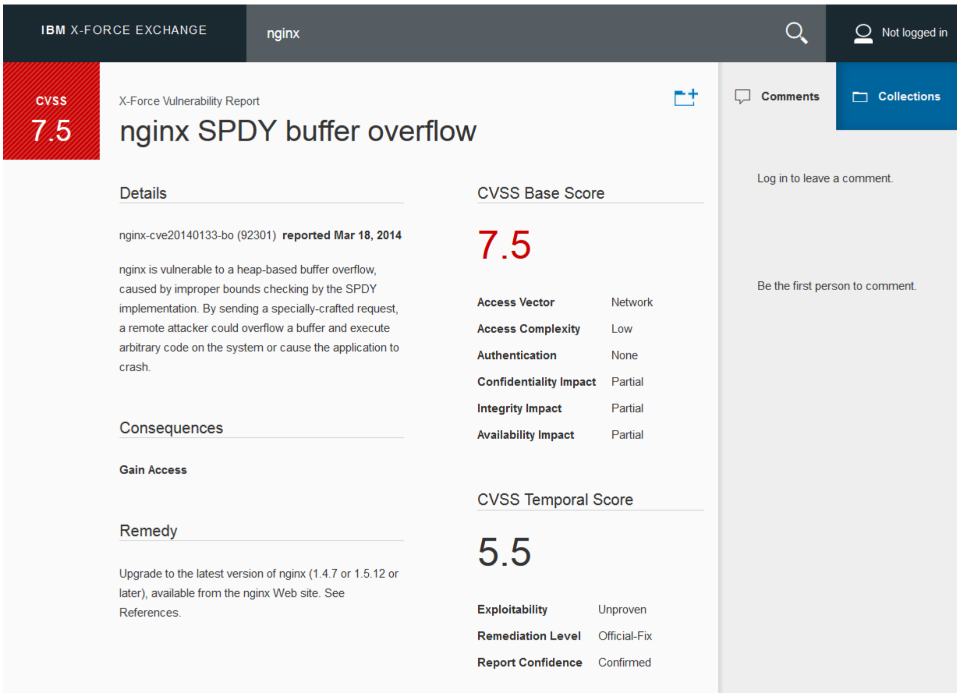
Like Soltra Edge, Malware Information Sharing Platform (MISP) is a complete platform built by MISP (2015). The platform is meant to facilitate sharing, storing and correlation of information related to malware and targeted attacks. Like Soltra Edge, it relies on groups of trusted partners to share information within. Once again both scope and intrusion differs between MISP and the solution proposed in this document. MISP replaces or becomes an additional software platform to maintain and relate to, while this solution integrates into existing processes and software. MISP focuses on malware and targeted attacks, while this solution is indifferent to the type of attack and handles all types of incidents. MISP could be used as a backend for the interface in this solution if desirable.

8.6.4 Real-time Inter-network Defense

Moriarty (2012) describes the Real-time Inter-network Defense (RID), a system for real time information broadcasting from a provider, primarily to a consortium of providers. It is, however, possible to use RID between only two participants. There are many similarities between RID and the solution proposed in this document. Both rely on SLA and contracts in order to establish the needed trust, both rely on one-to-one connections, and both focus on exchanging actual security incident information. There are noticeable differences as well, like RID preferring consortia while this solution prefers one-to-one agreements. In a RID relationship, the other party receives everything even though they might not be interested or need the information. The solution presented in this document allows the subscriber to decide which kind of information he wishes to receive. RID uses the more complicated format IODEF, this solution uses a simpler base format while still allowing for more complicated and complete formats like IODEF and STIX to be attached.

8.6.5 X-Force Exchange

IBM News Room (2015) describes IBM X-Force Exchange, which is a solution meant to make researchers and industry collaborate on threat intelligence. Participants might query the tool for information about threats or vulnerabilities as can be seen in figure 8.3, and IBM states that support for doing so using TAXII and STIX is planned. From the information made available and experimenting a bit with the tool, is seems like IBM X-Force Exchange is complementary to the solution described in this document, not a replacement. Exchange of actual, ongoing incidents are unlikely to happen as openly as what is planned for IBM X-Force Exchange, but rather kept between a lower number of trusted providers and subscribers. The IBM X-Force Exchange could, however, be valuable for incident handlers in identifying similar attacks or vulnerabilities and examine how such problems have been solved in the past.



X-Force Vulnerability Report

nginx SPDY buffer overflow

Details

nginx-cve20140133-bo (92301) **reported Mar 18, 2014**

nginx is vulnerable to a heap-based buffer overflow, caused by improper bounds checking by the SPDY implementation. By sending a specially-crafted request, a remote attacker could overflow a buffer and execute arbitrary code on the system or cause the application to crash.

Consequences

Gain Access

Remedy

Upgrade to the latest version of nginx (1.4.7 or 1.5.12 or later), available from the nginx Web site. See References.

CVSS Base Score

7.5

Access Vector	Network
Access Complexity	Low
Authentication	None
Confidentiality Impact	Partial
Integrity Impact	Partial
Availability Impact	Partial

CVSS Temporal Score

5.5

Exploitability	Unproven
Remediation Level	Official-Fix
Report Confidence	Confirmed

Log in to leave a comment.

Be the first person to comment.

Figure 8.3: The IBM X-Force Tool

8.6.6 Fully Integrated Defense Operation

Fry et al. (2015) describe Fully Integrated Defense Operation (FIDO), a system for automated incident response developed by Netflix. FIDO uses commercially available products as detectors, interprets their events, and tries to develop this information further before acting on it or notifying the security team. FIDO analyzes the target of the attack to see if it is up to date, which type of software is running, etc. By combining the information received in the events, with the analysis of the target and information from external databases, FIDO is able to correlate and score events. The solution presented in this thesis is not a replacement for FIDO, but could, based on the available information, be a useful complementary solution in order for other providers to act as detectors in the FIDO ecosystem. Likewise, FIDO seems to be a good starting point for examining how to introduce automation into incident handling in general, and this solution in particular.

8.7 Validity of Results

Given the low number of participants, it is not possible to generalize the answers and claim that they represents a normal incident handler. Thus, the study has low external validity, but the internal validity is made high by creating similar environments for the participants and following the same script for the interview. The answers of the participating incident handlers, and their reflections, are nonetheless valuable.

Another potential problem is the fact that the author of this thesis was responsible for both developing the solution and interpreting the interview results. This could lead to confirmation bias and the results therefore not necessarily reflect the actual views of the interviewees. In order to counter this threat, emphasis has been placed on trying to include most of what was said by the interviewees in their own words. This has not been possible to a full extent, as the answers have been organized by subject, transferred to the third person, and translated into English. The reader should nonetheless be able to identify any potential misconceptions in the analysis and use of the interview results.

8.8 Further work

While this thesis contribute a suggestion on how to exchange security incident information both in the cloud and between organizations in general, there are still more work to be done before such a solution could be widely adopted. Here, the work is separated into Non-technical, Interface, Incident Format and Prototype.

8.8.1 Non-technical

There is a need for examination of how SLAs should be created and what they should contain in order to facilitate sharing of security incident information. There should also be done work on how establish trust between providers, or work around the problem of transferring trust from a personal to an organizational level.

More work needs to be done on the legal aspects of security incident sharing; identifying, measuring, and mitigating the potential legal problems an organization might be exposed to by sharing security incidents. The possibility of using canned legal assessments for predefined incident types should also be examined.

The impact on an organization's public image as a result of sharing incident information should be subject for examination. Does incident sharing impact the public's view of the organization? If so, how? How does this impact the organization's bottom line.

Another aspect to examine is whether actively sharing incident information leads to a better threat and security awareness, and if so, does this contribute to improved security?

Furthermore, there should be done examinations on how to best inform the end user about incidents related to services he use and particularly his personal information.

8.8.2 Interface

Security is an important concern when creating and adopting solutions such as the one described in this thesis. Further work on securing the channel between two organizations is needed. Is Secure Socket Layer (SSL) a good enough solution? Is more transfer security needed? Mutual authentication of organizations as well as exchange of secret key to generate hmacs with is needed.

Another aspect that should be examined, is the consequence of a link being compromised. Would this affect the security of other instances in the network? If so, how could that be mitigated? Should the interface include some indication on whether it has been compromised or not? How would such an indication work and how would one know that the indicator has not been compromised?

The idea from *participant B* in section 7.2.1, about making a comment section available to all recipients of an incident, should be examined further. The examination should focus on how this would be represented in the interface, how it would impact the format, how it would impact adoption, the trust needed for such comments to work and whether shared comments improves incident handling for both providers and subscribers.

Which elements from the interface defined in this thesis could be useful for inclusion in TAXII? If any, how should the integration be done. Which elements from TAXII could be useful for inclusion in the interface defined here? Is this interface suitable or adaptable for the use cases TAXII is created for? If not, what are the significant differences and are they equally significant for all organizations?

8.8.3 Incident Format

The format needs further testing with more incident handlers in different situations. Existing incidents should also be represented with the new format, in order to observe any shortcoming or missing critical data fields.

The use of TLP should be examined. Particularly how it could be integrated in the format, and how to solve the problem of different TLP protocols. A potential inclusion of TLP raises other questions such as: How should the protocol be enforced? How to detect that the receiver does not comply with the protocol? Should it be possible to grade different information in the same incident with different lights of the TLP? If so, how?

The need and feasibility of a *Next update* field should be examined. Would this provided added value? Should it be an exact time or a unit of time? Should the subscriber be updated even if there is nothing new to tell? Does this provide the subscriber with needed information?

Should the format include a way of representing history? If so, how should this be represented? Should the history be included when deriving incidents? How is the history events created? - as a difference between last and current notification or all actions conducted during the solving the incident? How would the different detail levels of history affect the way incident handlers use the incident management tool?

8.8.4 Prototype

The prototype, in its current inception, is not ready to be used by incident handlers, but allows for exploration of concepts and ideas. Some of the concepts and ideas that needs to be studied further are archiving incidents, workflow, internal notifications and communication, indication on whether an incident has been read or not, how to distinguish between local and received information, develop hierarchy of incidents and visualization of such an hierarchy.

Further research should address how automation could best be done and which tasks are suitable for automation. There should also be some work done on visualization and how this might assist the incident handlers. Furthermore, some consideration should placed on how automation and visualization could work together with the incident handler - making him more efficient - rather than replacing him.

Chapter 9

Conclusions

This thesis shows the need for exchanging security incident information and contributes toward achieving this goal by suggesting a way to o this by means of an interface and an incident format.

9.1 RQ-1: Exchange Interface

A subscription-based interface between a provider and a subscriber, as described in this thesis, would allow the provider to push security incident information to the subscriber in a timely manner, while still leaving the provider in control of which information is shared and when it is shared. The simple format makes it easier for organizations to get started with incident information sharing, while still allowing organizations with more complex needs to attach fully featured formats like IODEF and STIX.

9.2 RQ-2: Non-technical challenges

The most prominent non-technical challenge is one of trust. It is necessary for organizations to establish some form of trust on an organizational level, rather than on a strictly personal level for exchange of security incident information. It is expected that this, at least to some degree, can be accomplished by means of contracts and SLAs.

Sharing of security incident information is further complicated by privacy laws, placing restrictions on the information that might be shared. The advance of logging solutions like Logstash and Transparency Log might be central in solving this challenge.

Public relations or the common perception of a company is another challenge; the question of how information sharing affects the company's image. Sharing of security incident information in a professional way might improve the image of a company, while unprofessional sharing might harm it.

Adoption is a major challenge as the solution will only be useful when in use. This thesis has identified three main drivers for adoption of such a solution: reduced costs, increased revenue or legal requirements.

9.3 RQ-3: Current Practice

Email is currently heavily used in organizations for exchange of security incident information, sometimes integrated directly into incident management tools. The solution presented in this thesis could replace email as the channel of incident information exchange without much impact on how the incident handlers work, given that the integration into incident management tools can be done at the same level of detail as today's email integration.

While not common practice, TAXII is the most similar solution identified during this thesis. Both provide the possibility of sharing incident information and offer subscriptions – though through different models. The solution presented in this thesis has less features than TAXII, but offers ease of integration and more simplicity.

The workflow of the prototype for common incident handling did not fit well with the current practice of the involved participants. However, the participants did point out that if the underlying interface and format were integrated into their current tooling, it would be a useful solution. This is normal as people in most sectors are resistant to changing tools, and prefer to keep using the tools with which they are already familiar.

Bibliography

Ahmad, A., Hadgkiss, J., Ruighaver, A., Jul. 2012. Incident response teams Challenges in supporting the organisational security function. Computers & Security 31 (5), 643–652.
URL http://linkinghub.elsevier.com/retrieve/pii/S0167404812000624

Almsaeed, A., 2015. Almsaeed Studio.
URL https://almsaeedstudio.com/

Amazon.com, I., 2015. Amazon.com Announces First Quarter Sales up 15% to $22.72 Billion.
URL http://phx.corporate-ir.net/phoenix.zhtml?c=97664&p=irol-newsArticle&ID=2039598

Ask Solem and contributors, 2011. Celery: Distributed Task Queue.
URL http://www.celeryproject.org/

Barnum, S., 2012. Standardizing cyber threat intelligence information with the Structured Threat Information eXpression (STIX). Tech. rep., The MITRE Corporation.
URL http://stix.mitre.org/about/documents/STIX_Whitepaper_v1.0_(Draft).pdf

Best Practical Solutions LLC, 2015. RTIR: RT for Incident Response.
URL https://www.bestpractical.com/rtir/

Brown, M. W., Stikvoort, D., Kossakowski, K.-P., Killcrece, G., Ruefle, R., Zajicek, M., 2003. Handbook for Computer Security Incident Response Teams (CSIRTs) — SEI Digital Library. Tech. Rep. April, Software Engineering Institute.
URL http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=6305

Cain, P., Jevans, D., 2010. Extensions to the IODEF-Document Class for Reporting Phishing. Tech. rep., IETF.
URL http://www.rfc-editor.org/rfc/rfc5901.txt

Cichonski, P., Millar, T., Grance, T., Scarfone, K., 2012. Computer Security Incident Handling Guide: Recommendations of the National Institute of Standards and Technology, 800-61. Revision 2. Tech. rep., National Institute of Standards and Technology.

Connolly, J., Davidson, M., Schmidt, C., 2014. The Trusted Automated eXchange of Indicator Information (TAXII). Tech. rep., The Mitre Corporation.
URL `http://taxii.mitre.org/about/documents/Introduction_to_TAXII_White_Paper_May_2014.pdf`

Cusick, J. J., Ma, G., 2010. Creating an ITIL inspired Incident Management approach: Roots, response, and results. In: Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP. Ieee, pp. 142–148.
URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5486589`

Danyliw, R., Meijer, J., Demchenko, Y., 2007. The Incident Object Description Exchange Format.
URL `http://www.ietf.org/rfc/rfc5070.txt`

Debar, H., Curry, D., Feinstein, B., 2007. The Intrusion Detection Message Exchange Format (IDMEF). Tech. rep., IETF.
URL `http://www.ietf.org/rfc/rfc4765.txt`

Django Software Foundation, 2015. The Web framework for perfectionists with deadlines — Django.
URL `https://www.djangoproject.com/`

Doelitzscher, F., Reich, C., Knahl, M., Passfall, A., Clarke, N., 2012. An agent based business aware incident detection system for cloud environments. Journal of Cloud Computing 1 (1).
URL `http://link.springer.com/article/10.1186/2192-113X-1-9`

European Union Agency for Network and Information Security (ENISA), 2015. Information disclosure.
URL `https://www.enisa.europa.eu/activities/cert/support/incident-management/browsable/incident-handling-process/information-disclosure`

Floodeen, R., Haller, J., Tjaden, B., 2013. Identifying a shared mental model among incident responders. Proceedings - 7th International Conference on IT Security Incident Management and IT Forensics, IMF 2013, 15–25.

Frøystad, C., 2014. Exchange of Security Incident Information in the context of Cloud Services.

Fry, R., Evans, B., Chan, J., 2015. Introducing FIDO: Automated Security Incident Response.
URL `http://techblog.netflix.com/2015/05/introducing-fido-automated-security.html`

Greenleaf, G., 2012. The influence of European data privacy standards outside Europe: implications for globalization of Convention 108. International Data Privacy Law 2 (2), 68–92.
URL http://idpl.oxfordjournals.org/content/early/2012/04/04/idpl.ips006.abstract

Grobauer, B., Schreck, T., 2010. Towards Incident Handling in the Cloud :. In: Proceedings of the 2010 ACM Workshop on Cloud Computing Security Workshop. ACM, Chicago, Illinois, USA, pp. 77–85.
URL http://doi.acm.org/10.1145/1866835.1866850

Horne, B., 2014. On Computer Security Incident Response Teams. Security & Privacy, IEEE 12 (October 2014), 13–15.
URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6924687

Hove, C., Tarnes, M., Line, M. B., Bernsmed, K., May 2014. Information Security Incident Management: Identified Practice in Large Organizations. In: 2014 Eighth International Conference on IT Security Incident Management & IT Forensics. IEEE, pp. 27–46.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6824080

IBM News Room, 2015. IBM Opens Threat Intelligence to Combat Cyber Attacks.
URL https://www-03.ibm.com/press/us/en/pressrelease/46634.wss

Internationl Telecommunication Union, 2012. ITU-T Rec. X.667 (10/2012) Information technology - Procedures for the operation of object identifier registration authorities: Generation of universally unique identifiers and their use in object identifiers. Tech. rep.
URL http://www.itu.int/rec/T-REC-X.667-201210-I/en

ISO, 2004. ISO 8601: Data elements and interchange formats - Information interchange - Representation of dates and times.

ISO, 2013. ISO/IEC 27036-3: Information technology - Security techniques - Information security for supplier relationships - Part 3: Guidelines for information and communication technology supply chain security.

Kalloniatis, C., Mouratidis, H., Vassilis, M., Islam, S., Gritzalis, S., Kavakli, E., Jun. 2014. Towards the design of secure and privacy-oriented information systems in the cloud: Identifying the major concepts. Computer Standards & Interfaces 36 (4), 759–775.
URL http://linkinghub.elsevier.com/retrieve/pii/S0920548913001840

Kampanakis, P., Sept 2014. Security automation and threat information-sharing options. Security Privacy, IEEE 12 (5), 42–51.

Merton, R. K., Kendall, P. L., 1946. The Focused Interview. American Journal of Sociology 51 (6), 541 – 557.

Metzger, S., Hommel, W., Reiser, H., May 2011. Integrated Security Incident Management – Concepts and Real-World Experiences. In: 2011 Sixth International Conference on IT Security Incident Management and IT Forensics. Ieee, pp. 107–121.
URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5931116`

Microsoft IT, 2014. Microsoft IT Introduction - Modernizing our Datacenter.
URL `http://blogs.technet.com/b/it_pro/archive/2014/08/18/microsoft-it-introduction-modernizing-our-datacenter.aspx`

MISP, 2015. MISP - Malware Information Sharing Project.
URL `http://www.misp-project.org/`

Moriarty, K., 2012. Real-time Inter-network Defense (RID). Tech. rep., IETF.
URL `http://tools.ietf.org/html/rfc6545`

Oates, B. J., 2006. Researching information systems and computing. Sage.

Open Source Initiative, 2015. The MIT License (MIT).
URL `http://opensource.org/licenses/MIT`

Osborne, C., 2015. Threat-sharing cybersecurity bill unveiled.
URL `http://www.zdnet.com/article/threat-sharing-cybersecurity-bill-unveiled/`

Peffers, K., Tuunanen, T., Gengler, C. E., Rossi, M., Hui, W., Virtanen, V., Bragge, J., 2006. The Design Science Research Process: A Model for Producing and Presenting Information Systems Research. the Proceedings of Design Research in Information Systems and Technology DESRIST06 24, 83–106.
URL `http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:The+Design+Science+Research+Process:+A+Model+for+Producing+and+presenting+Information+Systems+Research#0`

Phillips, A., Davis, M., 2006. RFC 4646: Tags for Identifying Languages.
URL `http://www.ietf.org/rfc/rfc4646.txt`

Pulls, T., Peeters, R., Wouters, K., 2013. Distributed privacy-preserving transparency logging. In: Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society. WPES '13. ACM, New York, NY, USA, pp. 83–94.
URL `http://doi.acm.org/10.1145/2517840.2517847`

Python Software Foundation, 2015. No Title.
URL `https://www.python.org/`

SANS Institute, 2015. Volume XVII - Issue #28.
URL `http://www.sans.org/newsletters/newsbites/xvii/28`

Schneier, B., 2014. The Future of Incident Response. Security & Privacy, IEEE 12 (October), 95–96.

Soltra Solutions LLC, 2015. About — Soltra.
URL http://avalanche.fsisac.com/soltra/about

State of California Department of Justice Office of the Attorney General, 2015. DATA SECURITY BREACH REPORTING.
URL http://oag.ca.gov/ecrime/databreach/reporting

Synergy Research Group, 2015. AWS Market Share Reaches Five-Year High Despite Microsoft Growth Surge.
URL https://www.srgresearch.com/articles/aws-market-share-reaches-five-year-high-despite-microsoft-growth-surge

Tañà, L. V., 2013. EU Data Breach Notification Rule: The Key Elements.
URL https://www.privacyassociation.org/news/a/eu-data-breach-notification-rule-the-key-elements

Takahashi, T., Landfield, K., Kadobayashi, Y., 2014. An Incident Object Description Exchange Format (IODEF) Extension for Structured Cybersecurity Information.
URL http://www.rfc-editor.org/rfc/rfc7203.txt

The European Commission, 2013. Commision regulation (EU) No 611/2013 of 24 June 2013.

The European Parliament and the Council of the European Union, 2013. Inofficial Consolidadated Version After Libe Committe Vote Provided by the Rapporteur 22 October 2013 Regulation. Tech. Rep. October.
URL http://www.janalbrecht.eu/fileadmin/material/Dokumente/DPR-Regulation-inofficial-consolidated-LIBE.pdf

The Mitre Corporation, 2015. STIX - Structured Threat Information eXpression.
URL http://stix.mitre.org/

Twitter, 2015. Bootstrap - The world's most popular mobile-first and responsive front-end framework.
URL http://getbootstrap.com/

US-CERT, 2015a. Traffic Light Protocol (TLP) Matrix and Frequently Asked Questions.
URL https://www.us-cert.gov/tlp

US-CERT, 2015b. US-CERT Federal Incident Notification Guidelines.
URL https://www.us-cert.gov/sites/default/files/publications/Federal_Incident_Notification_Guidelines.pdf

Watson, G., May 01 1971. Resistance to change. The American Behavioral Scientist 14 (5), 745, last updated - 2013-02-24.
URL http://search.proquest.com/docview/1306750518?accountid=12870

Appendix A

Paper

The following paper has been submitted to:

7th IEEE International Conference on Cloud Computing Technology and Science (Cloud-Com 2015)
Vancouver, Canada, November 30 - December 3, 2015

Security Incident Information Exchange for Cloud Services

Christian Frøystad (IDI, NTNU), Erlend Andreas Gjære (SINTEF ICT), Inger Anne Tøndel (SINTEF ICT) and Martin Gilje Jaatun (SINTEF ICT)

Abstract—The complex provider landscape in cloud computing makes incident handling difficult, as Cloud Service Providers (CSPs) with end-user customers do not necessarily get sufficient information about incidents that occur at upstream CSPs. In this paper, we argue the need for commonly agreed-upon incident information exchanges between providers, and potentially end-user consumers, as a means to improve accountability of CSPs. The discussion considers several essential technical challenges and non-technical aspects related to improving the situation for incident response in cloud computing scenarios. In addition, we propose a technical implementation which can embed standard representation formats for incidents in notification messages, building over a publish-subscribe architecture and a web-based dashboard for handling the incident workflow.

Keywords—incident response, cloud computing, accountability

1 Introduction

Cloud computing offers its users a significant amount of benefits such as increased agility, reliability, easier and better scalability and elasticity, maintenance, device and location independence, and reduced cost [1]. Due to this, the popularity of cloud infrastructure is understandably on the rise among both smaller companies and multinational enterprises alike, further enabling start-up companies to innovate with rapidly growing customer bases without costly IT investments up-front.

While the benefits of cloud computing are well known, there are still drawbacks or challenges which need to be taken into account by stakeholders looking into adoption of such infrastructure. This paper focusses on *incident response*, which is the process of handling the occurrence of an incident from detection, through analysis, containment, eradication & recovery and preparation [2]. These activities have increased in complexity since the time when servers were physical machines running a single system for one organisation, possibly at their own physical premises, too. However, a set of security issues related to incident handling *in the cloud* were examined by Grobauer & Schreck [2] back in 2010, who then called for more research in several areas. In the years which have passed since this paper, there has only been published a little amount of research addressing those challenges. Most of this, however, is mainly concerned with digital forensics in the cloud, or more traditional incident response scenarios, and nothing on the perspective of dealing with provider chains. In general, we find a lack of academic attention paid to incident management between independent companies and organizations involved in cloud computing, and to giving the involved parties sufficient access to related data and event sources in a timely manner.

Sharing of incident information has typically been based on one-to-one trust relationships between Computer Security Incident Response Team (CSIRT) members in the relevant organisations. The actual exchange of incident information normally happen by means of email, phone, incident trackers, help desk systems, conference calls and face to face. With the advent of cloud computing, however, the human element is much less prominent. A cloud service can be made up of a chain of providers where none of the CSIRT members have ever communicated directly with a representative from any of the other providers. In addition, an incident in the cloud may need to involve entirely new parts of the provider chain, potentially even in a automated, real-time fashion, to minimise business disruption [3]. This sets new requirements to the way incident response needs to be managed and supported by tools which are able to communicate efficiently across rapidly changing constellations of organisations.

An essential challenge is that there is no single part of the supply chain which has access to all events and all areas to monitor. This results in different actors having access to only limited parts of the relevant information, where nobody can immediately see the full picture. In a survey conducted by Torres [4], *little visibility* into system/endpoint configurations/vulnerabilities as such was considered one of the top hindrances to effective incident response in their organisation. The cloud actors therefore need to be able to communicate efficiently in order to provide each other with information to ease detection or assist responding to an incident. It has also been claimed that attackers are better at handling information sharing than those protecting services and systems [5]. This adds to the importance of providing good tools and solutions to incident handlers.

In this paper we provide an overview of technical challenges (section 2) and non-technical aspects (section 3), which directly impact incident response abilities for cloud computing scenarios. Moreover, in section 4, we propose a notification message format which has room for both standards-based and customised contents, and a demonstrate it in use through a prototype of our *IncidentTracker* system. The prototype features a scalable architecture of *IncidentTracker* instances, each implementing a common web interface specification for communication between providers, as well as a simple web-based dashboard for managing the workflow related to notification messages. An overall discussion of the approach is provided in section 5, while section 6 concludes the paper.

2 Technical Considerations

As a basis for our proposed specification, we have investigated technical aspects of how incident information should dealt with in an efficient manner. The following provides an overview of how these aspects relate to cloud computing scenarios in particular.

2.1 One Format to Rule Them All?

In order to ensure that every party of the cloud supply chain is able to understand the information sent, we believe it is necessary to agree on a definition of an incident and its data elements. This would allow for easier development and deployment of new systems as well as increase interoperability, as one would not have to conform to multiple different definitions and data sets, but can rely on one base format. At the same time, if an organization needs a different representation internally, this should still be possible as long as it is feasible to translate this representation into the common representation.

Schneier [6] claims that *"Incidents aren't standardized; they're all different."* NIST [7] further state that each CSIRT team must choose their own list of required data elements, based on factors like team model, team structure and how the team defines an incident. This raises the question of whether it is actually possible to represent every incident in one format, or if it is a better approach to allow multiple formats and thus allow for specialization. By using only one format that is able to represent everything, one could, theoretically, know that all conforming implementations would be able to understand any incident received and be able to handle it automatically, if desired. On the other hand, the format would be very complex, as it needs to include mechanisms to represent all possible relevant information for any incident imaginable. This would lead to increased costs in implementing the solution, as well as increasing the cost for adding new types of incidents due to the high degree of coupling.

We have found the Incident Object Description Exchange Format (IODEF) to be a comprehensive format for expressing incidents. While IODEF in practice can support any property included in other relevant formats, including Structured Threat Information eXpression (STIX), eCrime, and Cyber Observable eXpression (CybOX), it also brings along significant challenges. The format is severely complex, making it difficult to understand for humans, and the amount of extensions that would need to be created in order for the format to be fully usable for automatic handling of incidents, is large. Another potential problem is the amount of data the format would have to represent for each incident. Floodeen et al. [8] has found that incident handlers in practice enters all information required by a system or a standard, rather than only the necessary information based on situation. This raises the problem of incident handlers wasting precious time and reducing productivity, which in turn would reduce the chances of such a system being adopted by providers.

At the other end of the scale, the option can be to only transfer unstructured text messages between parties in the cloud supply chain. This would reduce the solution to be a secure "email" system, and thus in accordance with how incident information is mostly exchanged today [9]. The ability to provide free text would allow the parties to exchange any information required, but they would have to agree on a special formatting for the text if planning to support automatic handling of incidents. This would make it easier to implement the solution, and thus reduce initial adoption costs. The flexibility of being able to input any information in any formatting, makes it easy to include new types of information as well as only include what is necessary for the situation at hand. Notable drawbacks of this approach include fragmentation in message formatting, leading to potential difficulties for humans in interpreting or encoding incident information.

A middle ground could be to have a small base format, with the ability to represent the most common information in a simple way and providing a structured way of attaching other incident formats such as those mentioned above. This allows for reuse of existing incident formats, specialization of the formats, incremental implementation of formats as needed, flexibility to support newer formats, and the possibility to also exchange evidence in the case of a forensic investigation on behalf of another party in the cloud supply chain. This allows multiple levels of implementation which will allow businesses and organizations to start with a simple solution that can grow with their needs. The matter of forward compatibility is also an important factor in preferring this approach.

2.2 Notification

At some point, the CSP experiencing an incident needs to notify its consumers. One approach is to decided it all *a-priori* by the mutual agreement between provider and customer, who each make sure that they exchange the necessary information to fulfil the relevant laws and provide sufficient data for information exchange to be of use.

Another approach could be to notify about everything, but flag notifications which fits the defined triggers and incidents types. This would include notifications with a core message like: "We have been breached, but your data was not compromised.". While this could theoretically result in everyone having the necessary information, and being able to act upon it, it might just as well cause the subscriber to be flooded with information he does not need. It might also cause performance issues at the provider, as it would be a quite large amount of data to send.

A hybrid, combining the two, would allow the subscriber to mostly receive information he has asked for by defining triggers, while still allowing the provider to fulfil his legal obligations to notify about certain specific cases, such as breaches in systems containing personal information. The specifics of when and in which cases the provider should override the defined preferences and triggers defined by the subscriber, is likely to vary from jurisdiction to jurisdiction and will thus need local adjustments. This will not, however, affect the interface or the exchange format, only the back-end system.

Figure 1 shows an example of the relationship between service providers and services used, and thus provides an example of the amount of unneeded or unwanted incident information a subscriber could receive if the defined triggers are not taken into account. Therefore, it is expected to be better to use the hybrid approach, where the subscriber defines what he wants to be notified about and the provider additionally pushes all information required by law to the subscriber by using mandatory notifications that could be defined for each subscriber, thus being able to comply with different laws.

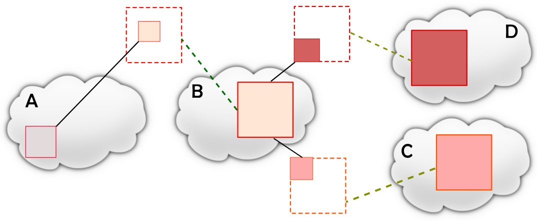


Fig. 1. Data flow in supply chain. Each cloud represents a service provider. Each colored square represents the services used by the subscriber. The colored square inside each stippled square, represents the data or parts of the services actually used by the subscriber.

2.3 Propagation of Incidents

Incidents which affect one CSP could in turn affect other CSPs, and hence be propagated. Two approaches to representing propagated incidents have been considered. The first approach embeds the parent incident in an unchanged manner, allowing all recipients of a derived incident – an incident created based on information from another incident – to access all of the information received by an earlier link in the supply chain. At first glance, this seems like a valid approach and one that could result in better cooperation, faster response time and better incident handling as a result of access to more and better information. However, if all information could be passed on to entities further down in the chain by a receiver, this would result in loss of control for the entity sending the incident as it would not be know who receives the incident information. Given the potentially sensitive nature of information included in or related to security incidents, there is a real possibility this would result in the system not being used or only superficial information being shared.

To allow the incident sender to be in control of his incidents, a better approach is to only reference the parent incident when propagating down to a subscriber of a subscriber. In this way, only relevant information would propagate directly, through the actions of an entity, while there would still be a hard link to follow in order to establish exactly what happen during the incident handling. This could, e.g., be useful when auditing a provider or during criminal investigations. Put simply, only those with a direct connection to a provider will be able to access the incident information sent by this provider. Subscribers of this provider may propagate incidents to their own subscribers, but not include the incident they received, only a reference to the incident notification. The original sender of the incident would hence maintain control of who accesses the information, and at the same time it is possible to secure a complete chain of custody and protect its integrity.

3 Non-technical Aspects

In order for this solution to be useful, it needs to be adopted by businesses and CSPs. Given how most businesses strive to improve their financial results, it is likely that for the system to be adopted, one of the following criteria must be fulfilled:

- Use of the solution results in reduced costs or increased revenue – directly or indirectly

  > ...companies need to know what they'll get back from the USG [US Government]. The commercial sector looks for 'return on investment,' and this is one area where they will seek a clear response. - Shawn Henry [10]

- Actors are required by law to use a system similar to this solution

3.1 Trust

It has been assumed that trust can be moved from a level where it exists between two humans, to a level where it exists between two organizations. This is not necessarily realistic. Traditionally, incident information has mostly been shared directly between people with a direct trust relationship, i.e., who know each other personally. Cloud providers need to trust each other on an organizational level, where one can agree to share information that relates to the services that the subscriber is using in the provider's infrastructure. As Henry states below, without trust, there will be reluctance towards sharing:

> The concerns of US companies, on the heels of Snowden and other revelations, are completely understandable. [...] Until companies can trust the USG [US Government], they will be reluctant to share. - Shawn Henry [10]

Many other problems can be defined as sub-problems of trust. Legal worries about sharing incident information comes from the fear of legal action as a result of information sharing, but this can be traced back to not trusting how the recipient uses the received information. Public relations worries related to public perception of the company being damaged as a result of sharing information can also be traced back to lack of trust in how the recipient uses the information received. While this solution does not automatically make service providers trust each other, it could make a way for communication to happen in a flexible fashion between distinct and already known organizations, through a secure channel and in an environment controlled by the sender of incident information. It is expected that most non-technical problems, such as trust and who is allowed to forward what information to whom, can be solved by adding terms to the respective Service Level Agreements (SLAs) and possibly adopt and enforce the Traffic Light Protocol (TLP) [11][12]. Use of sanctions for breach of contracts and trust might also assist in making it easier for organizations to share incident information, as they would know that any misbehavior by the other party would affect their bottom line.

3.2 Legislation

Laws are powerful incentives for changing behaviors in entire industries within a country. When large unions, like the United States or the European Union, introduces quite similar laws, this affects the entire western world and also, to some degree, the rest of the world [13]. Due to the substantial fines mandated by the Data Protection Regulation, service providers are given an incentive to ensure accurate and timely notification about breaches relating to personal information.

Laws are not only an incentive, but sometimes also a hindrance or at least an obstacle. The difference between laws covering personally identifiable information (PII), could complicate information exchange. The organization wishing to send incident information, needs to make sure that no PII is included. It has been claimed that information disclosure has the biggest potential contributor to CSIRT liability [14, p. 57]. To mitigate the fear of being sued for sharing incident information, the US introduced a bill to protect companies that shares information with the government from liability [15].

3.3 Financial

A *Level 1* implementation, that is exchange of security incident information without any automation in incident handling, is unlikely to result in significantly reduced costs, if any at all. However, the solution has been designed with implementation cost in mind, so the cost of adopting the solution should be quite low. The incremental nature of the solution allows implementer to gradually introduce more formats and also automation. As the implementation progresses into a *Level 2* implementation, with an increasing amount of automation, the reduced costs are expected to become noticeable. Metzger et al. [16] claim that more than 85 % of abuse cases can be partly or fully automated, which in turn would free up resources allowing for reduced costs.

This solution is unlikely to contribute directly to a higher revenue stream, but might contribute indirectly. If a CSP, or any other organization adopting this solution, is diligent in sharing information about incidents, this could contribute to building an image of trustworthiness and professionalism. Such an image could in turn result in more customers, and thus increased revenue.

4 *IncidentTracker* SPECIFICATION

Service providers need practical ways to handle many types of incidents, occurring in both the services they provide and in relation to services they rely on. In many cases, affected customers may also need to be notified, including both personal and corporate end-users, as well as other service providers using the service as part of their own service offering. The exchange of notifications, as well as the handling of notification messages, can be supported by using standardised channels for exchange of incident information. A standardised incident exchange format allows for increased automation of incident response tasks, yet it does not require automation to be implemented anywhere. Only the interface provisioned by notification publishers need to be defined, while the implementation of the underlying functions can be up to the implementers, and can vary between different actors.

4.1 Notification Message Format

For the content of the incident notifications messages, we propose a core format which supports the scenario manual incident handling as well, i.e. the case where the incident is read and acted upon by a human. Essential here is the capability to keeping track of e.g. where an incident has occurred and how it may be correlated with other incidents, to make the notification system well suited for complex networks of services. In order to support automation, i.e. when a service could adapt itself during runtime to mitigate a threat [3], we also make it possible to attach more structured and standardised formats to the message. A goal with our work is to keep the complexity to a minimum so that even small organisations/start-ups should be able to implement and utilise cloud incident response tools. While some organisations only need manual handling, we retain this possibility to support more extensive formats needed for automation, at the same time facilitating specialisation and forward compatibility.

The fields included in the core format are based on a review of the RFC 5070 standard for an IODEF [17], the Federal Incident Notification Guidelines from the US Computer Emergency Readiness Team [18], the EU Commission Regulation No 611/2013 [19], STIX [20], and a shared mental model for incident response teams described by Flodeen et al. [8], here presented in the JSON [21] notation with data type indications replacing the actual data:

```
{
    "id": UUID,
    "parent": {
        "id": UUID,
        "provider": STRING,
        "endpoint": URI
    },
    "type": {
        "id": UUID,
        "name": STRING,
        "description": STRING,
        "consequence": FLOAT,
    },
    "language": STRING,
    "status": STRING,
    "impact": FLOAT,
    "summary": STRING,
    "description": STRING,
    "occurrence_time": ISO 8601,
    "detection_time": ISO 8601,
    "liaison": {
        "id": UUID,
        "name": STRING,
        "email": EMAIL,
        "phone": STRING,
        "address": STRING,
        "zip": STRING,
        "city": STRING
    },
    "attachments": [
        {
            "format": STRING,
            "attachment": URI,
        },
    ],
    "custom_fields": [
        {
            "id": UUID,
            "value": STRING,
            "type": {
                "id": UUID,
                "name": STRING,
                "description": STRING,
                "type": STRING, INT, URI, JSON
                    ↪ , etc.,
            }
        },
    ]
}
```

The parent field in the format provides traceability to propagated incidents. Two approaches to representing propagated incidents were considered. The first approach embeds the parent incident in an unchanged manner, allowing all recipients of a descending incident to access all of the information received by an earlier link in the supply chain. At first glance, this seems like a valid approach and one that could result in better cooperation, faster response time and better incident handling as a result of access to more and better information. There are problems with the approach, though. If all information could be passed on to entities further down in the chain by a receiver, this would result in loss of control for the entity sending the incident as it would not be know who receives the incident information. Given the potentially sensitive nature of information included in or related to security incidents, there is a real possibility this would result in the system not being used or only superficial information being shared.

In order to allow the incident sender to be in control of his incidents, a better approach is to only reference the parent incident when propagating down to a subscriber of a subscriber. In this way, only relevant information would propagate directly, through the actions of an entity, while there would still be a hard link to follow in order to establish exactly what happen during the incident handling. This could, e.g., be useful when auditing a provider or during criminal investigations. Put simply, only those with a direct connection to a provider will be able to access the incident information sent by this provider. Subscribers of this provider may propagate incidents to their own subscribers, but not include the incident they received, only a reference to the incident notification. This way the original sender of the incident is still in control of who accesses the information, and at the same time it is possible to maintain a complete chain of custody and hopefully protect its integrity.

4.2 Custom Fields and Attachments

Our preliminary approach uses a small base format, with the ability to represent the most common information in a simple way, while providing a structured way of attaching other incident formats such as IODEF and STIX. In addition, the format supports *custom fields*, which allow the provider and the subscriber to agree upon extra information to be included in the base format without altering its base structure. This allows for two levels of implementation, as well as support incremental development of the highest level. *Level 1* would only implement the base format, and thus only be suitable for incident handling where there are humans acting on the incident reports. Some degree of automation could be supported in *Level 1* implementations if *custom fields* were used intelligently. *Level 2* would implement different attachment formats, and thus be able to support more automation of incident handling. This allows for reuse of existing incident formats, specialization of the formats, incremental implementation of formats as needed, flexibility to support newer formats for incidents not yet known, and the possibility to also exchange evidence in the case of a forensic investigation on behalf of another party in the cloud supply chain.

The option of having a defined set of incident formats, agreed upon between the provider and subscriber through SLAs, provides some interesting values. The interpretation of format is explicit, making it easier to distinguish between how to interpret different formats, as each attachment is accompanied with a value stating the type of attachment. Additionally, it gives flexibility to the provider and the subscriber, as they can identify the formats most suitable for their use case and apply those. It is likely that existing and standardized formats will be used in most situations, because of the investment required to develop a new format. Another possibility is that smaller and more specialized formats surfaces, where each format represents one and only one type of incidents. This option provides the flexibility of the first option, support for the second and at the same time mitigating the ambiguity problems of the first option.

Custom fields allow providers and subscribers to agree on extra information to include in the base format without changing its structure. This is an easy way of exchanging a few extra values that the subscriber wants, but without the overhead of a large incident representation format. There are multiple ways custom fields could be implemented, including allowing any values to be included in any format, allowing only a predefined set of basic data values, and providing data building blocks that allow representation of anything in a structured way.

Allowing any values in any format would provided a high degree of flexibility. It would, however, also introduce problems such as *custom fields* being abused when *attachments* should be used instead, and possibly also introduce problems with having to have interpretation rules for each cloud supply chain relation as the format used to encode custom information would differ.

By allowing only basic data types, such as *String*, *Integer*, and *Boolean*, use of *custom fields* where *attachments* should have been used would become less likely, but still possible through abusing the *String* value field. Explicitly stating the type of the value in the field would make it easier for the subscriber to interpret the value. A potential problem with this approach is the reduced flexibility, though it might be argued that *attachments* should be used for anything more complex than including simple values.

4.3 Architecture and Workflow

The customers of a cloud provider may have varying preferences when it comes to which systems, services and incident types they are interested in notifications for, as well as which thresholds for severity these should operate in relation to. Supporting such individual preferences can be accommodated by publishers e.g. through offering customers a subscription mechanism. The individual provider will obviously need to have the final say as to what information their customers are allowed to receive, regardless of preferences. To avoid data leakage and enforce the principle of least privilege, access to the Application Programming Interface (API) should only be provided through a secure channel, i.e. over HTTPS. Both senders and receivers should be authenticated.

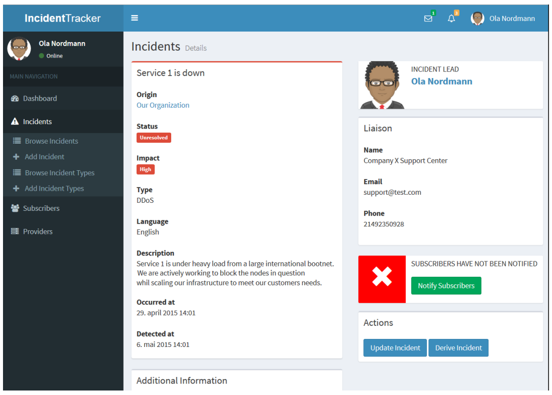
We propose an API, in our case implemented through Representational State Transfer (REST), which acts as an *adapter pattern* [22] so the system is allowed to function on every platform as long as they implement the API and provide the mechanisms needed to support REST. By reducing coupling to a minimum increases flexibility, modifiability and portability, and makes it easier to adopt the solution also for established systems and solutions. A list of all endpoints along with their HTTPS method can be found in Table 1.

TABLE 1
Complete REST interface for solution

Resource	URI	HTTPS METHOD
Incident types	/incidents/types	GET POST* DELETE*
Incident type	/incidents/types/{id}	GET POST* DELETE*
Trigger types	/incidents/types/{id}/triggers/types	GET POST* DELETE*
Trigger type	/triggers/types/{id}	GET POST* DELETE*
Notification Types	/notifications	GET POST
Notification Type	/notifications/{id}	GET POST DELETE
Notification incidents	/notifications/{id}/incidents	GET POST
Notification incident	/incidents/{id}	GET POST DELETE
Notification triggers	/incidents/{id}/triggers	GET POST
Notification trigger	/triggers/{id}	GET POST DELETE
Notification validation	/notifications/validate	POST

4.4 Dashboard Prototype

Fig. 2 shows a screen-shot from a working prototype built around the concept and specification outlined in this paper. The prototyped user interface presents a minimal, custom incident format which includes basic



Incidents Details

Service 1 is down

Origin
Our Organization

Status
Unresolved

Impact
High

Type
DDoS

Language
English

Description
Service 1 is under heavy load from a large international bootnet.
We are actively working to block the nodes in question
whil scaling our infrastructure to meet our customers needs.

Occurred at
29. april 2015 14:01

Detected at
6. mai 2015 14:01

Additional Information

INCIDENT LEAD
Ola Nordmann

Liaison

Name
Company X Support Center

Email
support@test.com

Phone
21492350928

SUBSCRIBERS HAVE NOT BEEN NOTIFIED

Actions

Fig. 2. Incident Details from the prototype built upon the proposed solution

incident information and is meant to be easy to understand for human incident handlers. As a subscriber, the dashboard allows you to browse through the received incidents notifications. As a provider, the prototype has functionality for defining possible incident types which others can subscribe to, as well as composing new incident notifications and updates related to such. For those who are both a subscriber and a provider, the interface contains functionality for deriving a new message from a received notification, which shall retain links to the origin of the notification.

A link to the repository containing the prototype source code will be made available after the review.

5 Discussion

Cusick and Ma [23] describe a solution similar to the one proposed here, but for internal use in the organization. Users might subscribe to be notified when events are created or changed, which has lead to improved communication around the incidents. The authors state that this feature alone made it worth their while to create a new process and implement new tooling. Our proposal takes this one step further, and allows other entities to be notified just as easily as the human subscribers. It is not known how simple it would be to integrate this solution with the Sharepoint based solution by Cusick and Ma, but theoretically this is possible without changing their working system. Cusick and Ma had a different experience from that described by Flooden et al. [8], who found that incident handlers entered any information requested by the system. Cusick and Ma found that engineers entered the minimum amount of information required and often just closed the ticket when it was handled without any information about steps taken to solve the incident. This problem will not be solved by the solution proposed here; if anything it will become more apparent and pressing. Depending on the relationship between two parties exchanging incident information, it might be alarming for one party to never receive any information other than "we have an incident – it was solved". This might lead the consuming organization to wonder if the incident has really been handled at all, and thus affect the trust between the two organizations. The solution to the problem is likely to be a combination of cultural and a question of resources.

Metzger et al. [16] describe a system which notifies subscribers about malware and other unwanted programs running on their systems, and is able to handle 85 % of all incidents in an automated manner – fully or partially. While our proposal does not provide any assistance on the detection part, it might be of assistance in notifying subscribers who could then also be offered more fine grained control over which notifications they wish to receive. The proposed solution could also be used for sensors submitting incidents to the central incident database. It would not, however, replace email and phone as reporting channels for all reporters. For collaborating organizations, the solution could replace such means of communication, but for single individual humans a simpler way of reporting would be needed, such as a web interface or email and phone as described in the article.

NIST [7] state that *The incident response team should discuss information sharing with the organizations public affairs office, legal department, and management before an incident occurs to establish policies and procedures regarding information sharing*. Our proposed solution facilitates such decisions to be taken before incidents occur, as subscribers are able to subscribe to incident types made available to them by the CSP. Thus, the CSP needs to have decided beforehand which incident types each subscriber is allowed to subscribe to. In addition, each organization is free to decide how to implement the backend and can thus require a man in the loop, allowing for a second screening of incidents before they are sent to subscribers.

Doelitzscher et al. [24] present a system for interconnected agents to detect and communicate occurring incidents. Here, our proposed solution could be used for the communication between the agents, allowing an aggregating node to receive information from multiple sensor agents, process the received information and pass it on to the higher link in the chain. It is not, however, clear that this would provide any added benefits over the approach already taken by the authors. One potential benefit could be that if the proposed solution, or some form of it, hypothetically becomes widely adopted and standardized, it would be easier to plug in new sensors and any other actors into the interconnected web of actors.

6 Conclusion

Incident information exchange is, at present, largely a manual exercise between two individuals who trust each other, via e.g. email, phone, or help desk systems. The same methods are prominent in both in-house and cloud chains – if ever incident information is exchanged at all. There are however legal requirements and contractual obligations which in many cases cements the need to exchange incident information in an effective manner.

The solution proposed in this paper is a first step in the direction of more effortless incident information exchange by means of formalistic and deterministic channels. Here we facilitate a direct connection between two parties in the supply chain, allowing for automated and/or manual handling of exchanged incident information. The proposal also allows for easy integration with existing systems and solutions, as well as use of internationally standardised message formats for possibly handling adaptation of services automatically during runtime.

Acknowledgement

(To be added after review)

References

[1] C. Kalloniatis, H. Mouratidis, M. Vassilis, S. Islam, S. Gritzalis, and E. Kavakli, "Towards the design of secure and privacy-oriented information systems in the cloud: Identifying the major concepts," *Computer Standards & Interfaces*, vol. 36, no. 4, pp. 759–775, Jun. 2014. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0920548913001840

[2] B. Grobauer and T. Schreck, "Towards Incident Handling in the Cloud :," in *Proceedings of the 2010 ACM Workshop on Cloud Computing Security Workshop*. Chicago, Illinois, USA: ACM, 2010, pp. 77–85. [Online]. Available: http://doi.acm.org/10.1145/1866835.1866850

[3] E. A. Gjære, H. Per, and T. Vilarinho, "Notification Support Infrastructure for Self-Adapting Composite Services," in *DEPEND 2014, The Seventh International Conference on Dependability*, no. c, Lisbon, Portugal, 2014, pp. 17–24. [Online]. Available: http://www.thinkmind.org/index.php?view=article\&articleid=depend_2014_1_40_50023

[4] A. Torres, "Incident Response : How to Fight Back A SANS Survey," pp. 1 – 25, 2014. [Online]. Available: http://www.mcafee.com/hk/resources/white-papers/wp-sans-incident-response-fight-back.pdf

[5] B. Horne, "On Computer Security Incident Response Teams," *Security & Privacy, IEEE*, vol. 12, no. October 2014, pp. 13–15, 2014. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6924687

[6] B. Schneier, "The Future of Incident Response," *Security & Privacy, IEEE*, vol. 12, no. October, pp. 95–96, 2014.

[7] P. Cichonski, T. Millar, T. Grance, and K. Scarfone, "Computer Security Incident Handling Guide: Recommendations of the National Institute of Standards and Technology, 800-61. Revision 2," National Institute of Standards and Technology, Tech. Rep., 2012.

[8] R. Floodeen, J. Haller, and B. Tjaden, "Identifying a shared mental model among incident responders," *Proceedings - 7th International Conference on IT Security Incident Management and IT Forensics, IMF 2013*, pp. 15–25, 2013.

[9] C. Frøystad, "Exchange of Security Incident Information in the context of Cloud Services," pp. 1 – 64, 2014.

[10] SANS Institute, "Volume XVII - Issue #28," 2015. [Online]. Available: http://www.sans.org/newsletters/newsbites/xvii/28

[11] US-CERT, "Traffic Light Protocol (TLP) Matrix and Frequently Asked Questions," 2015. [Online]. Available: https://www.us-cert.gov/tlp

[12] European Union Agency for Network and Information Security (ENISA), "Information disclosure," 2015. [Online]. Available: https://www.enisa.europa.eu/activities/cert/support/incident-management/browsable/incident-handling-process/information-disclosure

[13] G. Greenleaf, "The influence of European data privacy standards outside Europe: implications for globalization of Convention 108," *International Data Privacy Law*, vol. 2, no. 2, pp. 68–92, 2012. [Online]. Available: http://idpl.oxfordjournals.org/content/early/2012/04/04/idpl.ips006.abstract

[14] M. W. Brown, D. Stikvoort, K.-P. Kossakowski, G. Killcrece, R. Ruefle, and M. Zajicek, "Handbook for Computer Security Incident Response Teams (CSIRTs) — SEI Digital Library," Software Engineering Institute, Tech. Rep. April, 2003. [Online]. Available: http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=6305

[15] C. Osborne, "Threat-sharing cybersecurity bill unveiled," 2015. [Online]. Available: http://www.zdnet.com/article/threat-sharing-cybersecurity-bill-unveiled/

[16] S. Metzger, W. Hommel, and H. Reiser, "Integrated Security Incident Management – Concepts and Real-World Experiences," in *2011 Sixth International Conference on IT Security Incident Management and IT Forensics*. Ieee, May 2011, pp. 107–121. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5931116

[17] R. Danyliw, J. Meijer, and Y. Demchenko, "The Incident Object Description Exchange Format," pp. 1 – 91, 2007. [Online]. Available: http://www.ietf.org/rfc/rfc5070.txt

[18] US-CERT, "Federal incident notification guidelines," Tech. Rep., 2014. [Online]. Available: https://www.us-cert.gov/incident-notification-guidelines

[19] EuropeanUnion, "Commission regulation (eu) no 611/2013 of 24 june 2013 on the measures applicable to the notification of personal data breaches under directive 2002/58/ec of the european parliament and of the council on privacy and electronic communications," Tech. Rep., 2013. [Online]. Available: http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2013.173.01.0002.01.ENG

[20] S. Barnum, "Standardizing cyber threat intelligence information with the Structured Threat Information eXpression (STIX)," The MITRE Corporation, Tech. Rep., 2012. [Online]. Available: http://stix.mitre.org/about/documents/STIX_Whitepaper_v1.0_(Draft).pdf

[21] ECMA International, *ECMA-404 The JSON Data Interchange Standard*, Std., October 2013.

[22] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Pearson Education, 1994.

[23] J. J. Cusick and G. Ma, "Creating an ITIL inspired Incident Management approach: Roots, response, and results," in *Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP*. Ieee, 2010, pp. 142–148. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5486589

[24] F. Doelitzscher, C. Reich, M. Knahl, A. Passfall, and N. Clarke, "An agent based business aware incident detection system for cloud environments," *Journal of Cloud Computing*, vol. 1, no. 1, 2012. [Online]. Available: http://link.springer.com/article/10.1186/2192-113X-1-9

Appendix B

A4Cloud

The results from this thesis are used by the Cloud Accountability Project (A4Cloud) which focuses on Accountability For Cloud and Other Future Internet Services. Specifically, the interface and format is used by the Incident Management Tool (IMT) and the interaction between this tool and other parts of the toolchain, like A-PPL-E. More information about A4Cloud can be found on the project website: `http://www.a4cloud.eu/`

The project code is FP7-ICT-2011-8-317550-A4CLOUD

Appendix C

Interview Guide

This appendix contains the interview guide used when interviewing experienced incident handlers about the solution by use of the prototype. The guide was mostly created by Erlend Andreas Gjære from SINTEF.

Vi er . . . og arbeider med . . . Nå skal du få mulighet til å gjøre deg litt bedre kjent med et nytt system som kan hjelpe ulike aktører å kommunisere med hverandre om (potensielle) sikkerhetshendelser, IncidentTracker.

I denne testen skal vi vurdere . . . meldingsformat, API, dashboard. Det er meldinger og API som kan integreres med eksisterende systemer for hendelseshåndtering, og dashboardet du ser er kun et eksempel på noe som kunne vært tilbudt som ”standard”. Vi skal derfor ikke vurdere brukervennligheten av dashboardet som sådan, men vi vil likevel bruke prototypen til å teste konseptet, arbeidsflyt, informasjonsbehov, og behov for å formidle informasjon til andre.

Vi tester ikke deg. Men vi har utarbeidet et scenario, eller en serie med hendelser, som du kanskje vil kjenne deg igjen i. Måten vi gjør dette på er at du selv skal få gjøre en vurdering, som du ville gjort i din rolle ved ordinær håndtering av hendelser, med hva du ville gjort med de ulike hendelsene og her inngår også en faglig vurdering som vil lede til litt variert bruk av verktøyet. For vi ønsker at du skal prøver å bruke verktøyet/dashboardet til å oppnå det du ønsker.

Dersom du føler behov for utfyllende informasjon, kan du spørre oss det er ikke ”juks” i denne typen eksperiment. Du har selvsagt full anledning til å avbryte dersom du blir ukomfortabel eller ikke vil gå videre med en bestemt oppgave.

Tenke høyt underveis er bra!

Beskrivelse av prototypen og dens begrensninger. Vi har en oversiktsside (Dashboard), liste over hendelser (Incidents) her er det også mulig å se hvilke typer hendelser du som tjenesteleverandør tilbyr dine kunder/brukere under ”Browse Incident types”. Så har du kundene/brukerne dine under ”Subscribers”, her ligger NTNU som eksempel. Og til slutt er dine ”Providers”, altså de av deres underleverandører som også tilbyr IncidentTracker-grensesnittet.

Er det noe du lurer på?

C.1 Scenario

Simulert overvåkningssituasjon. Testperson kan klikke litt rundt. Sett stemningen: ”Hva gjør du når du kommer på jobb?”

Følgende spørsmål stilles når en ny hendelse dukker opp:

- Kan du si noe om hva som har hendt?
- Hvordan vurderer du innholdet i meldingen?
  - Er det tilstrekkelig til at du får en forståelse av situasjonen?
  - Er det for mye informasjon? Hva er i så fall overflødig?
  - Er det for lite informasjon? Hva ville du visst mer om?
  - Hva om meldingen kun inneholdt type og ingen beskrivelse?
    (Ville du åpnet vedlegg?)
  - Hvordan tolker du status?
  - Hordan tolker du Impact/konsekvensen?
  - Hvilke tidspunkter er relevante?
- Hvordan ville du håndtert situasjonen videre herfra?
  - Ville du sendt ut en melding til dine kunder/brukere?
  - Ville du kontaktet avsenderen? Ville dette vært ulikt avhengig av om du har møtt vedkommende?
  - Ville du ha arkivert meldingen, angitt den som ”lest”?
  - Ville du ha varslet noen av dine kollegaer? Hvordan?
  - Hvordan ville dashboardet passet inn i arbeidsflyten videre?
  - Hva forventer du skjer videre fra leverandøren sitt hold?
- Hvordan var det å opprette en sak?
- Hvordan ville du avsluttet en sak?

C.2 Fokusert intervju

- Hvordan varsles du normalt om hendelser? Hvordan formidles disse videre?
  Hva er vanskelig med hendelseshåndtering på tvers av organisasjoner?
  Hvor fort har dere bruk for å bli varslet? Er en e-post godt nok, er et dashboard noe man kan være innlogget i og sjekke gjennom dagen?

- Hvor mange hendelser er det egentlig som skjer? Hvilke blir varslet videre og hvilke blir ikke? Hvorfor/hvorfor ikke?
  Burde flere bli varslet videre? Er dette en ”risiko”?

- Opplevde du verktøyet som en støtte, eller var det en hindring kontra verktøy du har tilgang på i dag?
  Hva er annerledes?

- Hvordan er arbeidsflyten når du håndterer hendelser? Hvordan passer f.eks. dette verktøyet inn i dagens arbeidsflyt?

- Hva er riktig mengde informasjon for en hendelse? Spesifikt Hva ville du ikke hatt bruk for (se ulike hendelsestyper).

- Hva tror du skal til for at noen tar i bruk et grensesnitt/meldingsformat som dette? Jf. avtaleverk, lovverk (nye krav i EU-direktiv etc.), andre incentiver?
  Hvordan tror du behovet er for en slik løsning? Standardisering?

- Kjenner dere andre utviklingsprosjekter innen dette temaet, bruk av STIX/TAXII, osv.?

- Hvordan vil du vurdere realismen av hendelsene du var gjennom i den lille øvelsen vår?

- Noe du vil legge til? Ideer, forslag?

Appendix D

Cases for interview

This appendix contains the cases used when conducting the interviews. Incident 1 and 2 were already present in the prototype when the test participant started using it, while incident 3 and 4 arrived during the test. Incident 5 was a case where the incident handler should enter incident information into the system based on a received report.

Incidents one through four were created by the thesis author, while incident five was created by Erlend Andreas Gjære from SINTEF.

D.1 Incident 1

From	External provider
Type	DDoS
Status	Unresolved
Impact	Medium
Summary	Service 1 has slow response time due to DDoS
Description	Since this morning, we have seen a large number of connections, stressing our infrastructure. You might experience some reduced quality of service until we have been able to fully stop the attack, but our infrastructure should scale enough for our services to work though slow. We have identified the main perpetrator, and are working with ISPs and the police to put an end to this network. In the meantime, be advised about the IP address of the perpetrator.
Custom Fields	Perpetrator IP: 211.5.63.163
Attachments	
Language	English
Occurrence Time	19.05.2015 09:15:30
Detection Time	19.05.2015 09:16:10
Liaison	John Doe

D.2 Incident 2

From	External provider
Type	Data Breach
Status	Resolved
Impact	High
Summary	Database storage has been breached
Description	During the weekend, our database provider has reported a breach on one of their servers. We are not yet certain which, if any, of our services have been affected.
Custom Fields	
Attachments	
Language	English
Occurrence Time	16.05.2015 22:47:30
Detection Time	18.05.2015 08:34:55
Liaison	Jane Doe

D.3 Incident 3

From	Internal sensor
Type	Configuration
Status	Unresolved
Impact	Medium
Summary	New version of Nginx fixes vulnerabilities
Description	The installed version of Nginx (1.5.10) is vulnerable to CVE-2014-0088 nginx could allow a remote attacker to execute arbitrary code on the system, caused by an error in the SPDY implementation. By sending a specially-crafted request, a remote attacker could exploit this vulnerability to corrupt worker process memory and execute arbitrary code on the system or cause the application to crash. Please update your Nginx installation to the latest version
Custom Fields	CVE: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0088 X-Force: https://exchange.xforce.ibmcloud.com/vulnerabilities/92303
Attachments	
Language	English
Occurrence Time	04.03.2014 15:30:12
Detection Time	19.05.2015 10:30:30
Liaison	John Doe

D.4 Incident 4

From	External provider
Type	Probing
Status	Resolved
Impact	Low
Summary	Extensive port scanning of our infrastructure
Description	During the last days, we have seen a large number of Chines IP-addresses scanning our infrastructure for open ports. As a precaution, we have compared their scanning list to the one we run regularly to ensure only the correct ports are open. We can confirm that only intended ports are open, and these are secured.
Custom Fields	
Attachments	Scanned ports Scanning IP-addresses
Language	English
Occurrence Time	15.05.2015 02:47:10
Detection Time	15.05.2015 02:47:30
Liaison	John Doe

D.5 Incident 5

Reported internally by email.

> Hei!
>
> Jeg var på flyplassen i morges og da jeg hadde fylt på en kaffekopp etter maten, var plutselig PCen min borte fra bordet der jeg satt!
> Jeg spurte rundt, men det var ingen som hadde lagt merke til noen mistenkelige personer.
>
> ... dessverre så hadde jeg ikke logget av, jeg skulle jo bare en snartur bort til kaffemaskinen..!
>
> Hva gjør vi?! Det er ganske mye sensitivt på den laptopen, egentlig...
> Passord til systemer og sånt, hvis man vet hvor man skal lete.
> (Har meldt saken til politiet)
>
> Med vennlig hilsen
> John Døving
> IT-sjef
>
> Sendt fra min iPhone