NTNU
Norwegian University of
Science and Technology

Security Issues with Content Management Systems (CMSs) on the Cloud

Thomas Østdahl

Master of Science in Communication Technology
Submission date: June 2011
Supervisor: Danilo Gligoroski, ITEM

Norwegian University of Science and Technology
Department of Telematics

Problem Description

Joomla! is a widely used open source Content Management System (CMS) platform. Basically, it is a software which binds the content of a website to a template that describes the design and presentation. Many CMS actors offer cloud hosting, which is economically beneficial to customers because of flexibility and scaling. Hence, cloud security becomes an important issue for both service providers and their customers.

The candidate should investigate the security issues related to cloud computing, and then study how open-source CMSs manage these. By looking into the core of the Joomla! architecture, the candidate will get an insight in how the web content is protected. If vulnerabilities are found, they should be analyzed and discussed in detail. Finally, the candidate should investigate whether there exists cloud-specific security weaknesses regarding CMSs.

Assignment given:	15.01.2011
Supervisor:	Danilo Gligoroski

Abstract

Although cloud computing is the major hype nowadays, it is actually a relatively "old" concept which can be dated back to the 1950s. Then, AT&T was developing a centralized infrastructure and storage space, where their customers could connect to using advanced telephones. Cloud computing works in a similar fashion, where customers subscribe to centralized service models. The models are separated in three main categories; Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS). The cloud is a multi-tenant environment, i.e., several customers are able to use the same service simultaneously. Moreover, the cloud is highly scalable, which means that resources can be allocated on demand. Cloud computing follows a pay-per-use payment model. Customers could reduce their operational and maintenance costs significantly, because they subscribe to a Cloud Service Provider (CSP) which is responsible for these tasks. Moreover, an organization is no longer dependent on costly upfront investments.

Most of the industry-leading technology organizations (e.g., Amazon, Apple, Google, Microsoft) have their own cloud services. Thus, the barrier to adopt the cloud for customers has lowered. Organizations have hasted to move their services to the cloud, without questioning the cloud's maturity. Even though cloud security has been a priority from the beginning, numerous attacks have been reported. The CSP's data-centers provide both physical and infrastructure security. However, traditional security threats to IT systems, is still applicable to cloud applications. Furthermore, new cloud-specific security risks emerge. Confidentiality, integrity and availability of data are always of importance, however, becomes challenging in the cloud due to its dynamic environment. Ensuring integrity of data, without knowing the whole data set, is one of several challenges. Moreover, due to the increasing incidents of Denial-of-Service (DoS) attacks, availability of data has become problematic. Although the cloud is able to scale well with such attacks, disruptions of services still occur. The scalability of clouds could also potentially

be a threat, if malicious users are able to create bot-nets of multiple clouds.

The Internet is a hostile environment, likewise is the cloud. However, this does not stop people from rapidly adopting it. Organizations have hasted to offer their services on the cloud, to benefit from its advantages. Content Management Systems (CMSs) are examples of such services. They are widely popular, and used to create professional websites without requiring technical skills. CMSs provides a user-friendly platform to manage the contents (e.g., text, pictures, music), then customize it with templates and extensions. Open source CMSs benefit from their communities of developers, which contribute to keep their systems up to date and safe, with the current technologies. Since many CMS have non-technical users, they tend to be attractive targets for adversaries. Especially, third-party extensions have been considered a major threat. The "core" of the systems are often secure. However, web application vulnerabilities apply to these systems.

Joomla! is a widely popular open source CMS, due to its simplicity and remarkable community. The latest version (Joomla! 1.6) has made it suitable for both new unexperienced users, as well as professional users. Joomla! can be used as a PaaS, to benefit from the cloud's advantages. Anyhow, Joomla! is an attractive target, due to its non-technical audience. It is considered secure by default. However, with simple open source tools, it is possible to obtain valuable information about the system, e.g., server spesifications, OS, CMS version. Moreover, the security of the back-end have potential for improvements. Since many customers use weak passwords, and the back-end is always located in the same folder, adversaries could brute-force their way through. Many CMSs have static files and resources, which finger-printers utilize to determine the system specifications. Furthermore, poorly coded extensions are gateways for attackers. If an exploit is found in a component, adversaries could automate attacks against websites with this vulnerable component installed. Hence, it is critical for users to always keep up to date.

The emerging future would introduce numerous new ways of cloud usage. Many companies invest in enormous data-centers, which is the size of small villages. More and more services would move to the cloud, and software licenses would start to be excess. The threats to the cloud are not significantly increased in numbers, compared to traditional IT systems. However, the multi-tenancy could be exploited by malicious users. Moreover, distributed attacks originating from several clouds, could force CSPs to evolve cloud security.

Acknowledgements

This Master's thesis represents the final semester's work in the Master of Communication Technology program at the Norwegian University of Science and Technology (NTNU) in Trondheim. It was carried out in cooperation with the Department of Telematics (ITEM), under the Faculty of Information Technology, Mathematics and Electrical Engineering (IME).

I would like to thank my supervisor Danilo Gligoroski, for his guidance and help on my thesis. I would also like to thank the Joomla! community, which have rapidly answered all my questions regarding CMS security. Furthermore, I would like to thank my classmates for their friendship and valuable input during our time at NTNU. Finally, I would like to thank my family, and my girlfriend, for supporting me throughout my studies.

Trondheim, 09.06.2011
Thomas Østdahl

Contents

List of Tables

List of Figures

Abbrevations

ACL	Access Control List
AES	Advanced Encryption Standard
AMI	Amazon Machine Image
AWS	Amazon Web Services
ASP	Application Service/Infrastructure Provider
CaaS	Communication-as-a-Service
CCaaS	Compute Capacity-as-a-Service
CMS	Content Management System
CSP	Cloud Service Provider
CSRF	Cross-Site Request Forgery
DBaaS	DataBase-as-a-Service
DDoS	Distributed Denial-of-Service
DES	Data Encryption Standard
DoS	Denial-of-Service
DTaaS	Desktop-as-a-Service
EC2	Elastic Compute Cloud

EDoS	Economical Denial-of-Sustainability
IaaS	Infrastructure-as-a-Service
IPS	Intrusion Prevention System
ISP	Internet Service Provider
MaaS	Monitoring-as-a-Service
MAC	Message Authentication Code
NDA	Non-Disclosure Agreemen
OWASP	Open Web Application Security Project
PaaS	Platform-as-a-Service
POR	Proof of Retrievability
QoS	Quality of Service
SaaS	Software-as-a-Service
SCP	Secure Copy
SDLC	Software Development Life Cycle
SECaaS	SECurity-as-a-Service
SEO	Search Engine Optimazation
SLA	Service Level Agreement
SQS	Simple Queue Service
SSH	Secure Shell
STaaS	Storage-as-a-Service
UXD	User Experience Design

VM	Virtual Machine
XSS	Cross-Site Scripting

1 Introduction

"The Big Switch: Rewiring the world, from Edison to Google" [9], a book written by Nicholas Carr, points out the similarities between the rise of cloud computing in the information age, to electrification in the industrial age. Before the public electrical grid, organizations had to provide their own power. However, after the electrical network was available, organizations could just plug into the grid. Electricity became a utility. Many similarities can be seen with cloud computing. Computing resources have the potential to become a utility, in the same way as electricity did. In the emerging future, people could be able to connect to the cloud to get the resources they need. Applications and services will be available from anywhere, through an internet connection. Moreover, processing power, storage and memory could be scaled on demand. However, cloud computing introduces certain security risks. Cloud security has been a topic of concern from the birth of the cloud, even a dedicated non-profit organization (Cloud Security Alliance (CSA) [10]) has been established. CSA provides education about the cloud, and helps people use best practices for ensuring security within cloud computing. It is a common mistake to think that with cloud computing, known threats to traditional IT systems magically disappear. On the contrary, they contribute to make cloud security an even more challenging task.

Even though the cloud is a hostile environment, organizations seem to find it highly attractive. There are several reasons for this, e.g., scalability, reduced operational and maintenance costs, etc. The cloud has become a new playground to developers, thus numerous new services arrises. Moreover, existing services and applications can benefit from the dynamic cloud. Many CMSs have become available as cloud services, hence customers could easily create scalable websites, with practically unlimited resources. CMSs have traditionally been tempting targets for attackers, due to web application vulnerabilities. The cloud provides some security mechanisms to the CMS, however, they are still exposed to common threats.

This report gives an overview of cloud computing and the associated security risks. Both known threats to IT systems and new emerging threats are discussed. Since CMSs have the potential to greatly benefit from cloud computing, this report examines them from a security perspective. These systems have a considerable audience, whereas many of them are non-technical. As a consequence, they become attractive targets for adversaries. The report analyzes a popular CMS (Joomla!), and looks into how an attacker might find vulnerabilities on it. Three open source tools are used to simulate how an adversary could access exploitable information. Moreover, the source code of Joomla! is investigated, to identify how security is assured.

2 Cloud Computing

In this chapter, the history and evolution of cloud computing is described in detail. Moreover, it will give an overview of the cloud service and delivery models. Finally, the drivers of cloud computing are discussed.

2.1 History of cloud computing

Many of the main concepts of cloud computing can actually be dated back to the 1950s. At that time, AT&T was developing a centralized infrastructure and storage space which customer could connect to, using advanced telephones and an enhanced telephone network. By using load-balancing, the companies could utilize their resources more effectively and economically. The development of this concept continued over the next decades, and concepts such as Internet Service Providers (ISPs) and Application Service/Infrastructure Providers (ASPs) were adopted. With the ISPs, servers are located at the Internet access point. An ASP is when a customer rents infrastructure at a remote location, and often used mainly by this one paying customer. The problem, however, with the ASPs was that the customer needed to calculate the computing and storage capacity needed before renting the infrastructure. As a result, an upgrade of the capacity led to expensive hardware upgrades and critical delays [1].

The name, cloud computing, was most likely derived from the graphical representations of the Internet, commonly seen in computer textbooks. However, the concept of cloud computing did not get much attention until Salesforce.com deployed their website in 1999, which delivered enterprise applications. After Amazon renewed their data centers after the dot-com bubble, they became a major actor within the cloud. They saw the potential of cloud computing, and the economical savings it introduced, and launched Amazon Web Service (AWS) in 2006. AWS is a pay-per-use platform in the cloud, which provides the infrastructure needed for storage and computation resources [11].

"We in academia and the government labs have not kept up with the times, universities really need to get on board."

This citation is from Randal E. Bryant, dean of the computer science school at Carnegie Mellon University [12]. The statement was in context with the collaboration between IBM, Google and several american universities. IBM and Google joined forces in 2007 and started a research initiative on cloud computing. They invested in large data-centers, where students from six universities in the USA could remotely connect to. These remote resources could be used for programming and research. The goal of this initiative, was to explore the possibilities and limitations of the cloud, and to get attention to this new era in computing.

2.2 The evolution towards cloud computing

The evolution towards the cloud began with the first Internet Service Providers (ISP 1.0), which provided access to the Internet for both organizations and individuals. Dial-up modems were often used to connect to the Internet. As the popularity of Internet grew, the ISPs added new services such as access to email and servers at their facilities (ISP 2.0). With the introduction of these services, organizations and individuals wanted to host their own servers. Specialized facilities and corresponding infrastructure were constructed to support the servers, and enable applications to be run on them. These specialized facilities are known as collocation facilities (ISP 3.0). The next step in the evolution, was the Application Service Providers (ASPs). An ASP focuses on providing value-adding applications to organizations, not just the computing infrastructure (ISP 4.0). The application software and the necessary infrastructure were controlled by the ASP. Although ASPs provided services to multiple customers, they did not do this through a shared environment, which is the case for Software-as-a-Service (SaaS) providers nowadays. With ASPs, each customer had their own instance of an application which ran on a dedicated server. The evolution has led to cloud computing (ISP 5.0), which defines the SPI model (Software/Platform/Infrastructure). In figure 2.1, the evolution of cloud computing is illustrated. The SPI model will be discussed in detail in the following section.

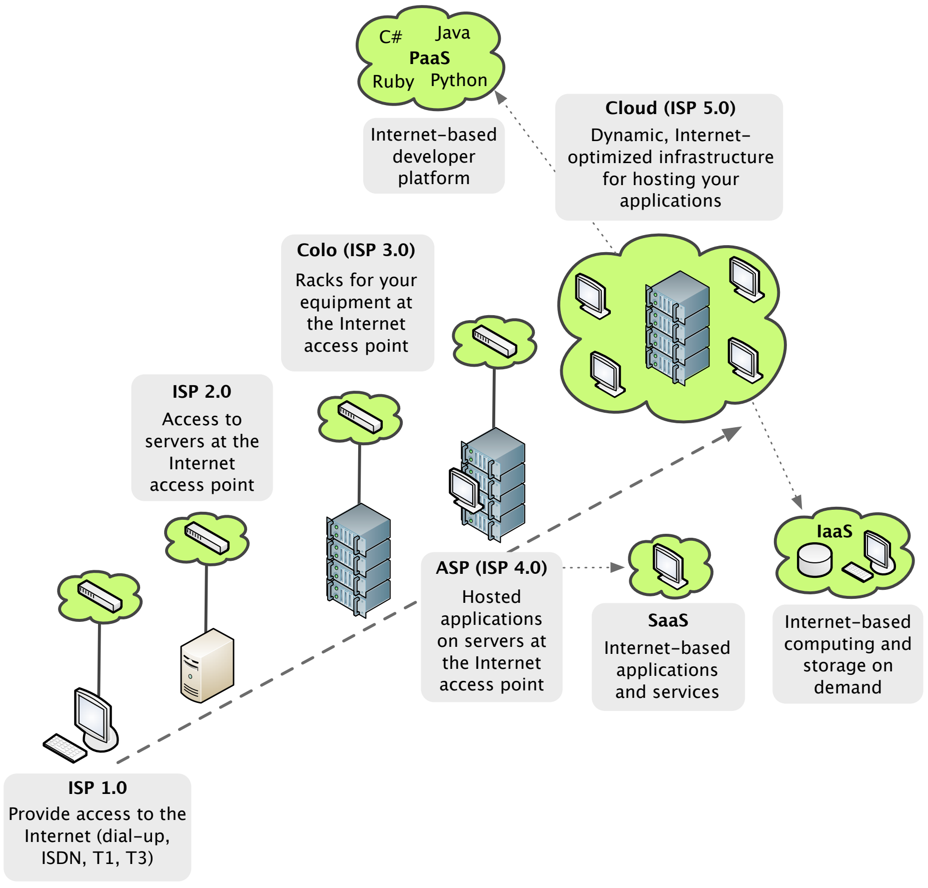

C#
Java
PaaS
Ruby
Python
Internet-based developer platform
Cloud (ISP 5.0)
Dynamic, Internet-optimized infrastructure for hosting your applications
Colo (ISP 3.0)
Racks for your equipment at the Internet access point
ISP 2.0
Access to servers at the Internet access point
ASP (ISP 4.0)
Hosted applications on servers at the Internet access point
SaaS
Internet-based applications and services
IaaS
Internet-based computing and storage on demand
ISP 1.0
Provide access to the Internet (dial-up, ISDN, T1, T3)

Figure 2.1: Evolution of cloud computing [2] (modified figure).

2.3 What is cloud computing?

"Cloud computing is a term used to describe a set of IT services that are provided to a customer over a network on a leased basis and with the ability to scale up or down their service requirements. Usually cloud computing services are delivered by a third party provider who owns the infrastructure." [1]

Cloud computing has gained popularity during the recent years, due to flexibility and the possibility to reduce operative costs. This concept can be described with four main characteristics:

- **Pay-as-you-go** – The customer pays for his/her consumption of a cloud service, i.e. the cost is variable.
- **Abstracted** – The hardware server, and related network architecture is abstracted from the customers.
- **Multi-tenant** – The multi-tenant architecture allows several customers to subscribe to the same cloud services, without compromising security and privacy.
- **Scalability** – Consumption and capacity (i.e., cost) can be scaled up and down transparently.

The customer only pays for the actual consumption, which could drastically decrease the customer's costs. Another benefit with cloud computing is that several users can use the same services, hence the utilization of resources is optimized. Earlier, the capacity has been predefined and static, thus upgrades became a costly and time-consuming operation. However, with cloud computing the capacity can be scaled transparently, without delays and extra costs.

2.3.1 Cloud delivery models

Cloud computing delivery models can be divided into four main categories. The difference between them, depends on the level of ownership and technical infrastructure.

- **Vendor cloud (external)** – this type of cloud delivery model can be accessed over the Internet or through a private network. The model utilizes virtualization technologies for rapid scaling, and can be used by multiple tenants. When sharing a service, one or more data centers can be utilized, and with different levels of access control. This cloud delivery model is also known as a public cloud. However, by having a shared environment, the customers have no dedicated resources, thus lack of control over them.

- **Private cloud (internal)** – The architecture of this cloud delivery model is similar to the vendor cloud model. However, it is built, managed and used by one single enterprise. This model is based on shared resources and variable use of virtual data resources, where the data is controlled by the enterprise. As a consequence, the enterprise can ensure both control and security over their cloud resources.

- **Hybrid cloud** – The hybrid cloud delivery model is a combination of the two models mentioned above, combined with a IT infrastructure. Therefore, this model is suitable for enterprises that wish to store non-confidential data externally, while keeping private data locally. The hybrid cloud model is flexible and can be adapted to the customer's needs.

- **Community cloud** – A community cloud is used between organizations with the same goals and concerns, thus they can share resources and services. This model can be deployed as one of the three models mentioned above.

2.3.2 The services of cloud computing

In cloud computing there are three main types of service models:

- Software-as-a-Service (SaaS)

- Platform-as-a-Service (PaaS)

- Infrastructure-as-a-Service (IaaS)

Additionally, there are many subsets of these three primary service models. In table 2.1 on page 8, the different subsets of the cloud service models are described in detail. Figure 2.2 on page 9 illustrates the SPI model as a hierarchy, and shows the relevant technologies.

Subservice type	**Description**
IaaS: DataBase-as-a-Service (DBaaS)	DBaaS allows the access and use of a database management system as a service.
PaaS: Storage-as-a-Service (STaaS)	STaaS involves the delivery of data storage as a service, including database-like servies, often billed on a utility computing basis, e.g. per gigabyte per month.
SaaS: Communication-as-a-Service (CaaS)	CaaS is the delivery of an enterprise communications solution, such as Voice over IP (VoIP), instant messaging, and video conferencing applications as a service.
SaaS: SECurity-as-a-Service (SECaaS)	SECaaS is the security of business networks and mobile networks through the Internet for events, database, application, transaction, and system incidents.
SaaS: Monitoring-as-a-Service (MaaS)	MaaS refers to the delivery of second-tier infrastructure components, such as log management and asset tracking, as a service.
PaaS: Desktop-as-a-Service (DTaaS)	DTaaS is the decoupling of a user's physical machine from the desktop and software he or she uses to work.
IaaS: Compute Capacity-as-a-Service (CCaaS)	CCaaS is the provision of "raw" computing resource, typically used in the execution of mathematically complex models from either a single "supercomputer" resource or a large number of distributed computing resources where the task performs well.

Table 2.1: Cloud sub-services [1]

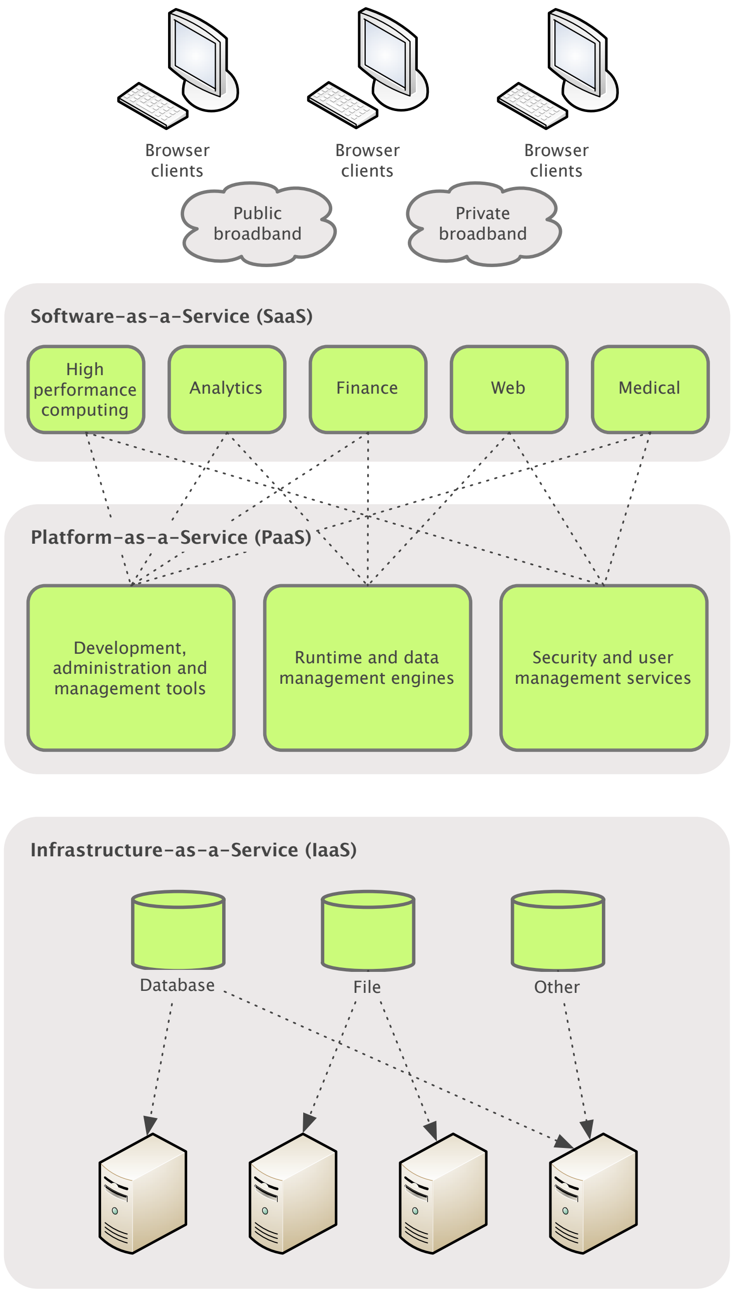

Browser clients
Browser clients
Browser clients
Public broadband
Private broadband
Software-as-a-Service (SaaS)
High performance computing
Analytics
Finance
Web
Medical
Platform-as-a-Service (PaaS)
Development, administration and management tools
Runtime and data management engines
Security and user management services
Infrastructure-as-a-Service (IaaS)
Database
File
Other

Figure 2.2: The architecture of cloud service models.

SaaS

This service model is based on the concept of one-to-many, i.e., multiple customers can subscribe to the same service simultaneously. SaaS could be described as a service which is accessed on a hosted server. Hence, the customers do not need to install the software on their local machines, only access the service over the Internet (e.g., Google Docs, Photoshop.com). The vendor is running the software on a cloud infrastructure, and makes sure updates and patches are installed continuously. The customers use the service on a subscription basis, where they pay for their actual usage. As a consequence, companies can reduce their expenses, since licenses for every employee are unnecessary. Considering that most computer are idle almost 70% of the time, SaaS can drastically decrease a company's expenses. Thus, this service model could be beneficial for companies which want a service, and do not want to spend money on infrastructure and the personnel to maintain it.

Many desktop software development companies have seen the benefits of SaaS, thus want to adapt their existing software to work on the new platform. However, this could be challenging because it often involves rewriting of software. This could be too costly and time-consuming for many companies. As a result, the movement to cloud computing has been a slow process for some companies. One solution is to release a highly scaled down version of their software, and incrementally increase the performance of it.

Another benefit with the introduction of SaaS, is that previously expensive software now is available for the general consumer. An example of this phenomenon is the popular photo-editing software; Adobe Photoshop. Nowadays, Adobe has also launched Photoshop as a SaaS, with reduced functionality. Many users do not need all the functionalities of the full version, therefore a light version offered as a SaaS would be suitable. This is an example of a freemium service, which is a term that describes a business model for scaled-down SaaS. The freemium model anticipates that a certain precent of the users will eventually buy the full retail version, or upgrade to a paid version of the software. A similar model is also found in desktop software.

Customer support is a simpler task with cloud comuting, hence a driver for moving services into the cloud. Developers can implement fixes shortly after bugs are found, without the need for customers to regularly download updates. As a consequence, most of the bugs are removed before the users encounter them, hence the number of support calls are drastically decreased. With the software running on the cloud, developers do not need to consider all the different platforms that exist. Developing a software, which is compatible with Windows, Linux and Mac OSX, could be a challenging task. Then, considering the many different versions of each operating systems, the task gets even more complicated. The economical benefits of using the cloud become obvious, as the control of the operating system and versioning is managed by the cloud vendor. By controlling the platform the software runs on, the developers can save money on testing and deployment of fixes or new features. Furthermore, this can all be done transparently to the users, which is advantegous for the user experience.

As the development and testing costs are notably reduced, the software companies can put more though into the user interface. Many of the new SaaS have been designed by a dedicated product team, which is a process known as User Experience Design (UXD). In conclusion, SaaS gives both developers and users several advantages. However, some desktop software companies could find it challenging to adapt their existing software to the new platform [13].

PaaS

Paas allows developers to build and deploy their applications on a hosted infrastructure. This service can be seen as the middle layer in a cloud stack, where SaaS is on the top and IaaS at the bottom. The cloud stack is shown in figure 2.3 on page 12. PaaS offers computing resources from a cloud infrastructure, which is only limited by the size of the infrastructure. An example of a PaaS provider is Google's App Engine. Taking into consideration that Google's infrastructure is estimated to contain one million x86-based computers [14], the computing resources could almost be seen as infinite. When developing a software, one of the most frustrating processes could be setting up the server, which often includes tasks like:

- Acquire and deploy the server
- Installing the operating system, run time environments and additional middleware

- Configuring the installation
- Move/copy existing code
- Testing and running of the code

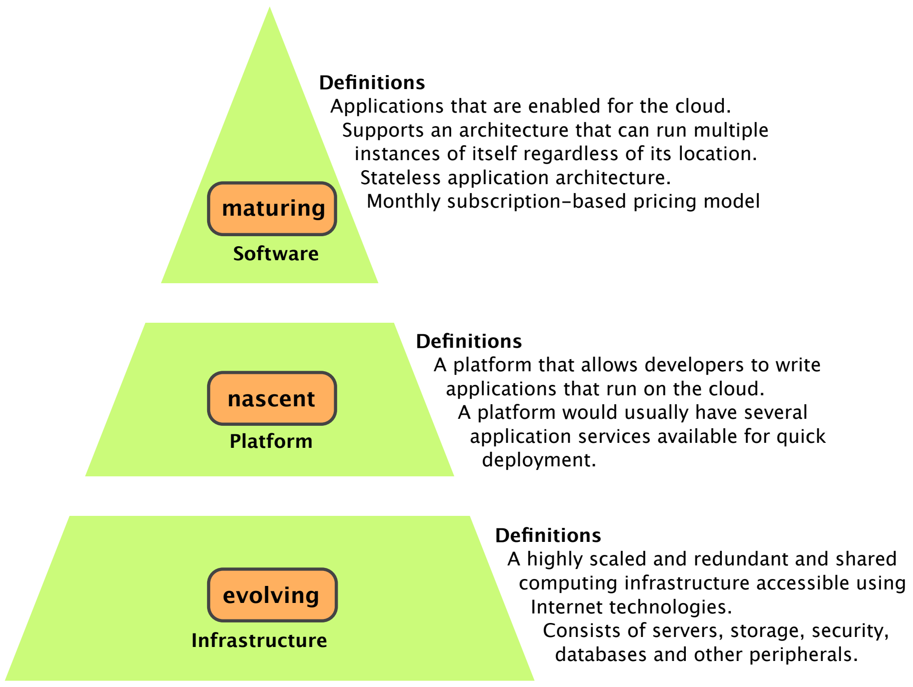

maturing
Software
Definitions
Applications that are enabled for the cloud.
Supports an architecture that can run multiple instances of itself regardless of its location.
Stateless application architecture.
Monthly subscription-based pricing model
nascent
Platform
Definitions
A platform that allows developers to write applications that run on the cloud.
A platform would usually have several application services available for quick deployment.
evolving
Infrastructure
Definitions
A highly scaled and redundant and shared computing infrastructure accessible using Internet technologies.
Consists of servers, storage, security, databases and other peripherals.

Figure 2.3: The service models of cloud computing as a hierarchy [2] (modified figure)

One advantage of PaaS is that it is possible to have a virtual machine (VM) containing the whole server environment for testing purposes. The VM could be put on a flash drive, thus easy to switch between clients. Basically there are two main components of PaaS; service stack and platform. The computing platform is the place were the service stack is deployed. Common platforms are Windows, Apple OSX and Linux for operating systems. Additionally, there are platforms for both mobile phones and software frameworks. The service consists of applications which will help in the testing and deployment process (e.g. operating system, run time environment, etc.).

Since there are many different platforms available, choosing a PaaS provider could be challenging. When choosing a provider, there are many factors to consider:

- Which frameworks and code languages are supported?
- How many applications can be created?
- What type of content is allowed?
- What kind of databases are supported?
- Does it support SSL? (Especially important with e-commerce services.)

These questions are important to ask when choosing a provider, to get the most out of your application. Another topic of relevance is vendor lock-in, which means that a customer is dependent of one vendor, thus unable to easily swap between them. A standardization of APIs and platform technologies are necessary to avoid this problem [14].

IaaS

As the name reveals, IaaS provide basic services such as data storage, databases and virtual servers. The service models of cloud computing can be seen as a cloud stack, where IaaS functions as this base layer. Without the base layer, the services can not be deployed and executed.

IaaS has several advantages, cloud-bursting being one of them. This terms refer to the process of moving tasks to the cloud when the compute resources are running low. The economical benefits of cloud-bursting is significant, because no additional investments for server equipment are necessary. Moreover, these servers normally use on average only a very small amount of their computing capacity. However, the process of off-loading tasks requires software which is able to reallocate processes to an IaaS cloud. Another term frequently used within IaaS is elasticity, which together with virtualization forms two important facets of IaaS. The elastic infrastructure of an IaaS can be described with an example: A customer needs to do statistical operations on a massive collection of data, which normally would take several weeks to process. By moving the collection to the cloud, the processing time could be drastically decreased. Firstly, it is necessary to create an instance of a server where the database software is implemented. This instance is called an image, and allows the customer to run queries on the collection of data. After deployment of the image, and putting the data into the database, it is possible to duplicate the image as many times as necessary. As a consequence, the data-processing can be run simultaneously on multiple instances. If a customer finds the data-processing too slow, he/she could simply add more duplications of the image. In other words, IaaS allows for easily configurations of resources for unexpected peaks of traffic.

The second facet of IaaS is virtualization, which handles infrastructure management tasks. This virtualization system runs beneath the operating system level. IaaS is platform independent, and consists of a combination of software and hardware resources. The software is low-level code and runs independently of the operating system. The software is called a hypervisor, and is responsible for allocating resources on demand. This process is called resource pooling, and makes virtualization possible. Virtualization enables a multi-tenant environment, which means that several customers can share the same infrastructure. In conclusion, IaaS provides an infrastructure with dynamic resource allocation [15].

2.4 Drivers of cloud computing

Table 2.2 lists the advantages of cloud computing over traditional client/server computing. Reduced complexity, costs and time to deploy a system are major drivers towards the cloud.

Traditional IT	**Cloud computing**
High upfront IT investmens for new builds	Low upfront IT investmens; pay-per-use model
High cost of reliable infrastructure	Reliability is built into the cloud architecture
High complexity of IT environment	Modular IT architecture environments
Complex infrastructure	No infrastructure

Table 2.2: Comparison of traditional IT systems and cloud computing.

2.4.1 Small initial investments and low ongoing costs

By utilizing a public cloud, no software, hardware or network equipment need to be purchased. As a result, a company can reduce their expenses massively. The pricing-model of the cloud is based on actual usage of the services. Due to the small initial cost, the barrier to enter the cloud becomes smaller. Since most applications are used only a small percentage of their lifetime, the pay-per-use model can be cost efficient.

2.4.2 Scalability

In most traditional development projects, it is difficult to predict the required computing resources. Usually, the developers need to calculate the requirements in advance, which could be a challenging task. Therefore, many projects get too much, or too little resources. These resources include storage, processing power and memory requirements for both development, testing and deployment of a project. With the flexibility of the cloud, the computing resources can be scaled on demand. Previously, a company needed to make huge investments when scaling their system. Moreover, an upgrade of the system meant downtime. However, by utilizing the cloud, with its dynamic nature, a project can adapt to changes more seamlessly.

2.4.3 Sustainability

The Cloud Service Providers (CSPs) have invested both money and thought into providing a sustainable environment for their customers. Traditionally, companies have struggled to maintain their services due to failures in the network, or simply adapting to rapid changes. CSPs, however, offer better resilience because of clustering, and have limited points of failure.

Chapter summary

In the recent years, cloud computing has been the major buzzword within IT. Many have predicted that it will have a serious impact on our lives. The concept of the cloud has been around for decades, however, it did not became publicly available until Salesforce.com released their SaaS website in 1999. Well-known companies, e.g. Google, IBM and Amazon, have brought cloud computing to a new level, thus the barrier of moving to the cloud has become lower. The economical benefits, together with the simplicity of development and testing, are major drivers for adopting the cloud.

3 Cloud Security

The following chapter sheds light on the different aspects of cloud security. The infrastructure of the cloud is discussed from three points of views; network, host and application level. Then, data security is evaluated using the CIA-triad[1]. Finally, cloud-specific security threats are described.

3.1 Infrastructure security

This section will be discussed in the context of the SPI service models. A common mistake is to assume that infrastructure security only is concerned with IaaS security. Although it is more relevant when using IaaS, the two other cloud service models should also be considered. Another important facet of infrastructure security is the cloud delivery models (e.g., private, public and hybrid). When utilizing a public cloud, the responsibility of infrastructure security has transferred from the organization to the CSP.

[1] The CIA triad is concerned with the three core security principles of information security; confidentiality, integrity and confidentiality

3.1.1 Network level

It is important to distinguish between private and public clouds, when considering infrastructure security at the network level. There are no specific threats associated with the topology of a private cloud. Thus an organization does not have to make significant changes to their existing network topology. A private cloud has many similarities to a secure private extranet, as seen in figure 3.1 on page 19. However, a change in the network topology is required when moving to a public cloud. An organization's network topology may have to be adapted to work with the CSP's network topology. There are four main risks associated with this use-case:

- Secure (confidentiality and integrity) transfer of data between an organization and its public cloud provider.
- Access control (authentication and authorization) when accessing public cloud resources.
- Availability of an organization's online resources from the public cloud provider.
- Domains replace the role of the traditional network zones and tiers.

Data integrity and confidentiality:

When using a public cloud, private data is exposed to the Internet and is located in a shared environment controlled by a third-party. Amazon Web Services (AWS) reported a security vulnerability in 2008, which is related to this topic. This vulnerability encompasses how Amazon constructed their digital signatures. These digital signatures were used when making queries (REST[2]) to Amazon SimpleDB, Amazon Elastic Compute Cloud (EC2) or Amazon Simple Queue Service (SQS) over HTTP. Thus, customers using HTTP instead of HTTPS were affected by this vulnerability.

[2] REST (representational state transfer) is an approach for getting information content from a website by reading a designated web page that contains an XML file that describes and includes the desired content [16].

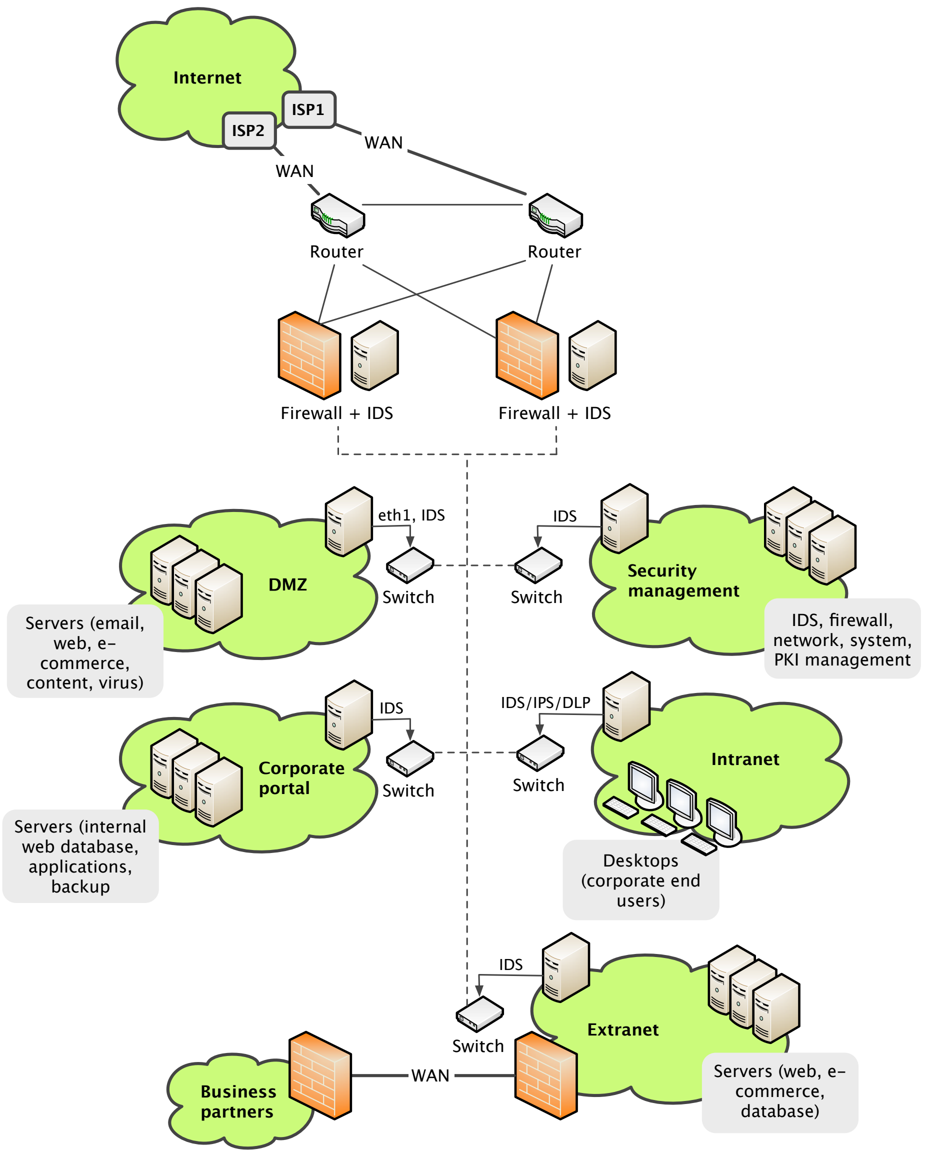

Internet
ISP1
ISP2
WAN
WAN
Router
Router
Firewall + IDS
Firewall + IDS
eth1, IDS
IDS
Switch
Switch
DMZ
Security management
Servers (email, web, e–commerce, content, virus)
IDS, firewall, network, system, PKI management
IDS
IDS/IPS/DLP
Corporate portal
Switch
Switch
Intranet
Servers (internal web database, applications, backup
Desktops (corporate end users)
IDS
Switch
Extranet
Business partners
WAN
Servers (web, e–commerce, database)

Figure 3.1: Generic network topology of private cloud computing [2] (modified figure).

"If you are making Query (aka REST) requests to Amazon SimpleDB, to Amazon Elastic Compute Cloud (EC2), or to Amazon Simple Queue Service (SQS) over HTTP, and there is any way for an attacker to provide you with data which you use to construct your request, switch to HTTPS or start using AWS signature version 2 now." [17]

Amazon forgot to implement collision-resistance, when inventing their digital signature scheme. It should be computationally infeasible to construct messages with the exactly same digital signature. An adversary could take advantage of a collision by using a substitution attack, where the digital signature is moved from a harmless message to the adversary's message. However, Amazon released a new digital signature version (AWS signature v2) which solved the problem.

Access control:

When moving resources to a public cloud, control and monitoring are administrated by a public cloud provider. Customers utilizing the public cloud, usually have limited access to network-level logs and data, hence limited possibilities to do forensic analysis. The lack of control could be problematic, especially if the CSP is reusing IP addresses. In a public cloud, the public cloud providers often reassign IP addresses instead of terminating them. IP addresses are a limited set, hence it makes sense to reuse them, from a public cloud provider's point of view. However, from the customer's point of view, this becomes a security issue. The IP address a customer used for accessing resources on the public cloud, could then be reused by adversaries. This problem is also applicable to the public cloud provider's internal network. Then, the resources of a customer could be reached internally by other customers of the same public cloud. Therefore, the security features of public cloud providers should be looked into before choosing whom to use.

Availability:

Network security has got an increased amount of attention the recent years, due to adoption of cloud services. Nowadays, organizations rely on the security of external network devices hosted by cloud providers. Border Gateway Protocol

(BGP)[3] prefix hijacking (i.e. modification of network layer route announcements) is an example of attack related to this use case. The prefix hijacking involves announcing routes to other Autonomous Systems (AS)[4] without permission. These announcements usually occur by configuration mistakes, however, it still affects the availability of the resources. An example of a failure which caused availability problems, happened in February 2008 when Pakistan Telecom tried to deny access to Youtube within Pakistan. The Pakistani government asked Pakistan Telecom to block Youtube for the Pakistani population (which is estimated to be around 8.2 million Internet users), because of blasphemous content. It is not unusual that countries block services for their population, another example is China's blocking of Google. However, Pakistan Telecom made two critical mistakes. They created a dummy route, which rerouted all Youtube requests to a black hole. Announcing the dummy route to their own telecommunication partner in Hong Kong (PCCW), was the first mistake. PCCW was responsible for the second mistake, by accepting the dummy route and relaying it to other ISPs around the world. Now, ISPs had two conflicting routes to Youtube. The BGP protocol favors longer routes, thus several ISPs chose the dummy route, leading to the Pakistan Telecom's black hole. As a result, millions of Internet users around the world were denied access to Youtube. Although, prefix hijacking is a well-known a quite old security issue, it is assumed to become increasingly relevant as the cloud continues to grow [20].

Another type of risk associated with availability is Domain Name System (DNS)[5] attacks. In addition to vulnerabilities in the DNS protocol and in the implementations of DNS, there exist attacks known as poisoning attacks. DNS cache poisoning attacks is an attack in which the server is tricked into accepting malicious information. This attack has been known for many years, however recently new variants of this attack has occurred.

[3] BGP is a protocol for exchanging routing information between gateway hosts (each with its own router) in a network of autonomous systems [18].

[4] On the Internet, an AS is the unit of router policy, either a single network or a group of networks that is controlled by a common network administrator (or group of administrators) on behalf of a single administrative entity (such as a university, a business enterprise, or a business division) [19].

[5] The DNS is the way that Internet domain names are located and translated into Internet Protocol addresses. A domain name is a meaningful and easy-to-remember "handle" for an Internet address [21].

The last examples of attacks associated with availability are Denial of Service (DoS) and Distributed DoS (DDoS) attacks. Anonymous, a group of hackers which act on behalf of WikiLeaks[6], are well known for their DDoS attacks against those who refuse to do business with them. They have initiated DDoS attacks against major companies, such as PayPal and Swiss Bank. The WikiLeaks site itself has also been a target for this type of attack, shortly after they revealed thousands of US embassy documents [22]. The DDoS attack hit WikiLeaks with 10Gbps of bogus data, which stopped their servers to work properly. The source of this attack was unknown, however, this massive DDoS attack forced WikiLeaks to change their service provider. They moved their site to Amazon's EC2, because this cloud would scale better in case of another DDoS attack. However, Amazon decided they did not want to assist WikiLeaks with their work, thus refused them to utilize their cloud [23].

New network model:

The traditional network model, consisting of network tiers and zones, is in public IaaS and PaaS clouds replaced with domains. Usually, zones have been used to differentiate intranet and extranet, or development and production. As a result, the network traffic is separated, hence improved security. In the “old" model, the zones and tiers had certain access rights associated with them. Therefore, only people with specific roles could access the different zones and tiers. SaaS clouds built on public IaaS or PaaS have similar characteristics.

In public cloud computing, “security groups”, “security domains” or “virtual data centers” have replaced the role of zones and tiers. The new model uses logical separation between tiers, which is less precise than the old model. In AWS, the security groups feature allows VMs to access each other through a virtual firewall, which filters traffic based on IP addresses, ports or packet types (e.g., UDP, TCP or ICMP). A test domain and a production domain could be located on the same server, hence the requirement of physical separation has disappeared. Furthermore, the logical separation at the network level does not longer exist, and has been replaced by logical separation at the host level. In other words, domains can run

[6] WikiLeaks is a non-profit organization which publishes secret, classified information from anonymous sources.

on the same physical server, separated by VM monitors (hypervisors).

Network-level mitigation:

Considering the previous sections, the network-level risks are not associated with the different cloud service models, but rather the cloud delivery models. The main risk factors are related to wether an organization choose to use private, public or hybrid clouds, not IaaS, PaaS or SaaS. Choosing the private cloud is the most secure option, however, also the most expensive. Using encryption on data in transit, reduces the confidentiality risks. Additionally, by adding digital signatures to the data makes it infeasible to tamper with it, hence integrity is ensured. The availability problems at the network-level are difficult to mitigate using public cloud computing. A solution is to use a private cloud which is internal to an organization's topology. However, the risks associated with availability are not more relevant with cloud computing, than with traditional public and private extranets.

3.1.2 Host level

The risks associated with the host level in cloud computing, are related to both the cloud service models (SaaS, PaaS and IaaS) and the cloud delivery models (private, public, hybrid and community). There exists cloud specific threats, however, these are discussed in a later section. Public cloud computing utilizes virtualization techniques to provide a dynamic environment, thus virtualization security becomes an important factor. Virtualization security threats include VM escape[7] , system configuration drift and insider threats due to poor access control. Moreover, the dynamic environment of the cloud leads to frequent changes of VM instances, hence vulnerability and patching management becomes more challenging.

[7] Virtual machine escape is an exploit in which the attacker runs code on a VM that allows an operating system running within it to break out and interact directly with the hypervisor [24].

SaaS and PaaS host security:

Usually, CSPs hide information about their host platforms, operating systems and security related processes. If hackers obtain this information, they could exploit it and access the system. Thus, most CSPs have transparent security mechanisms towards the customers. However, it is possible to get this information through a Non-Disclosure Agreement (NDA)[8] with the CSP. Both PaaS and SaaS platforms hide information about the operating system from customers using a host abstraction layer. A major difference between PaaS and SaaS is the accessibility to this layer. SaaS users are unable to access the abstraction layer, while PaaS users interact with it indirectly through the API. In conclusion, most of the issues concerning SaaS and PaaS host security are handled by the various CSPs. Hence, it is the customer's responsibility to find out how the CSPs manage them.

IaaS host security:

The IaaS customers are responsible for securing their own cloud hosts, unlike SaaS and PaaS customers. Normally, IaaS utilizes virtualization at the host layer. Hence both virtualization software security and virtual server security are of importance. Customers are able to create and destroy virtual instances, with the help of virtualization software. This software is located between the hardware and the virtual servers and is controlled solely by the CSP, thus customers are not able to view or access it. The virtualization itself can be accomplished using any of the virtualization models:

- OS-level virtualization (e.g., Solaris containers, BSD jails, Linux-VServer)
- Paravirtualization (a hybrid between the hardware version and versions of Xen and VMware)
- Hardware-based virtualization (e.g., Xen, VMware, Microsoft Hyper-V)

Both hardware and OS virtualization allows for VMs to share resources in the multi-tenant environment, without interfering with each other. The resources are allocated by a program called the hypervisor, which is the OS of the virtualization

[8] A NDA is a legal contract between at least two parties, where an outline of confidential information is shared.

system. Each OS appears to have the host's memory, processor and other resources all by themselves. To achieve this, the hypervisor carefully allocates resources to the VMs on demand. VMs are the main ingredient in IaaS, thus isolation and security of each VM is critical, due to the shared nature of the cloud. In figure 3.2, the virtualization system components are put into context. Each of these components have has been subject to security vulnerabilities.

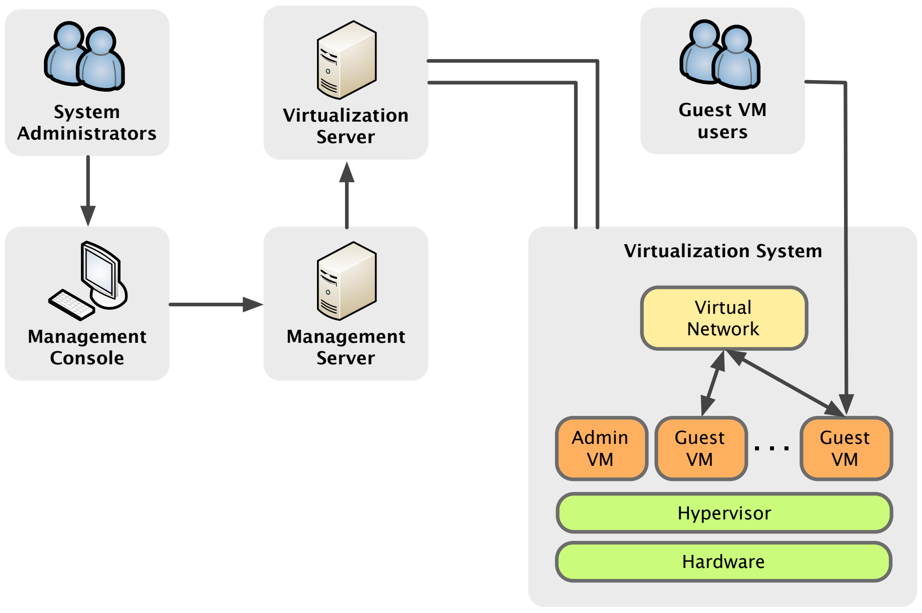


Figure 3.2: Virtualization System Components [3] (modified figure)

The customers of IaaS have full access to the guest VMs, which are isolated and managed by the hypervisor. Therefore, customers are responsible for the securing them. The EC2, a public IaaS, offers a web services API to its customers. This API is used for management functions, thus allows for scalability of resources when needed. Due to the dynamic lifetime of VMs, automated procedures need to be implemented to provide seamless management. Moreover, access control of the virtual instances is necessary, since the virtual server (Windows, Solaris or Linux) may be reachable to anyone on the Internet. Usually, all ports to the virtual instances are closed by the CSPs. Furthermore, the CSPs often recommend their customers to utilize port 22 (SSH[9]) for administration of the virtual instances. Nevertheless, host security threats in public IaaS still exist:

- Stealing keys used for access and manage hosts (e.g. SSH private keys)
- Vulnerabilities on unpatched services listening on standard ports (e.g., FTP)
- Hijacking of vulnerable accounts (e.g., weak passwords)
- Attacking systems without host firewalls
- Deployment of trojans in VMs

The paper "You are doing it wrong" by SecureNetwork [26], was presented at the BlackHat conference in Europe 2011. This paper discusses problems with the virtualization systems today, and proposes a new concept for security. The traditional access control rules to VMs, should be replaced by a semantic to enforce rules on "services". Then apply these rules on services or logical items, instead of on the physical system, as we see today. Furthermore, security should be controlled by dedicated security teams. The paper introduces two new components to realize their main goals; virtual cells (vCells) and a gatekeeper (vGatekeeper). VCells are logical items which one can enforce rules on. The vGatekeeper is responsible for enforcing the rules on the vCells, and should be able to do so even though the vCells are compromised. As a result, the compromised vCell is isolated. Access to vCells is only allowed through the vGatekeeper, in the same way as traffic flows through a firewall. This suggested structure for a secure public IaaS is illustrated in figure 3.3 on page 27.

[9] Secure Shell (SSH), sometimes known as Secure Socket Shell, is a UNIX-based command interface and protocol for securely getting access to a remote computer [25].

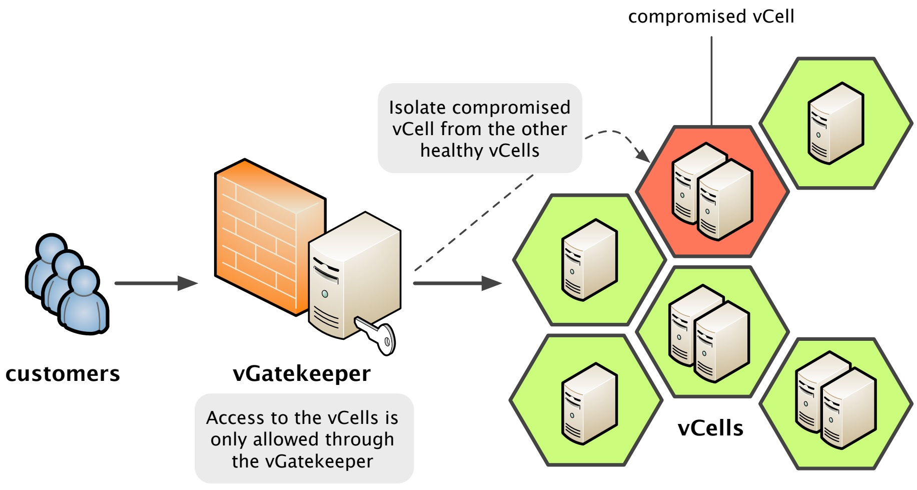

compromised vCell
Isolate compromised vCell from the other healthy vCells
customers
vGatekeeper
Access to the vCells is only allowed through the vGatekeeper
vCells

Figure 3.3: A new approach to a secure public IaaS structure

3.1.3 Application level

Application security is an important aspect of an organization's total security program. However, the implementation and design of this security could be a challenging task due to the huge variations in both platform and application type. Web applications come in several variants, spanning from small single-user applications to massive and complex multi-user e-commerce systems. An example of a web application used by both small and large organizations are Content Management Systems (CMS), which are discussed in detail in the following chapter. Another challenging factor when securing web applications is the variety in web frameworks used, e.g., PHP, .NET, Python, Java, etc. Adding the different cloud delivery and service models to the mix, makes it even more complex. Since cloud applications are accessed through a web browser (e.g., Google Chrome, Mozilla Firefox, Internet Explorer, Safari, Opera), browser security needs to be integrated into the application security program. In conclusion, the developers of cloud application security face many challenges to ensure confidentiality, integrity and confidentiality of data.

Security threats:

Web application vulnerabilities are found in both open source and custom built applications. The Open Web Application Security Project[10] (OWASP) has made list of the ten most critical web application security risks. This list with explanations is included in appendix A on page 87. OWASP points out injections, Cross-Site Scripting (XSS) and broken authentication and session management as the tree most vulnerable risks. The IBM X-Force publishes a yearly report, considering the trends and risks of the recent year. According to IBM X-Force, web application vulnerabilities covers 49% of all disclosures in 2010 [3]. In figure B.1 in appendix B on page 92, a graph based on data collected from hundreds of vulnerability-tests done by IBM, shows the average number of vulnerabilities found in each threat-category (e.g. XSS, Cross-Site Request Forgery (CSRF)). There are numbers of conclusions to be drawn from this graph. For example, CSRF vulnerabilities have increased drastically until 2009, where it reached a turning point. Furthermore, XSS vulnerabilities have followed an almost similar path as CSRF. This is likely due to greater awareness of this risks recently, thus better detection techniques have been implemented. Moreover, in 2010 improper use of SSL is most likely to cause a vulnerability in web applications, according to IBM X-Force's report. Hackers are continuously scanning web applications to find vulnerabilities to exploit. Numerous scanners are easily accessible on the Internet, thus finding known vulnerabilities becomes frighteningly easy. The motivation for exploiting vulnerabilities in web applications is widespread, including financial gain, converting trusted servers into malicious servers (i.e., creating botnets) or phising scams. Traditionally, a combination of perimeter security, network- and host-based access controls are used to provide a defense against attacks. A majority of the same threats also apply to applications applied in the public cloud, thus many of the same defense mechanisms are necessary. As a result, web applications deployed in the public cloud, must implement security in every step of the Software Development Life Cycle (SDLC), as illustrated in figure 3.4 on page 29.

Another threat to the application-level in public cloud systems, is DoS attacks. This kind of attack could potentially disrupt a cloud service for a severe amount of time. Usually, DoS attacks originate from large networks of compromised servers

[10] OWASP is an open community dedicated to enabling organizations to develop, purchase, and maintain applications that can be trusted [27].

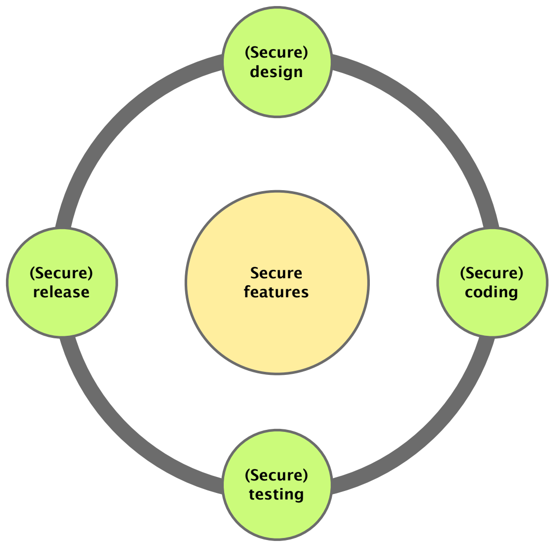
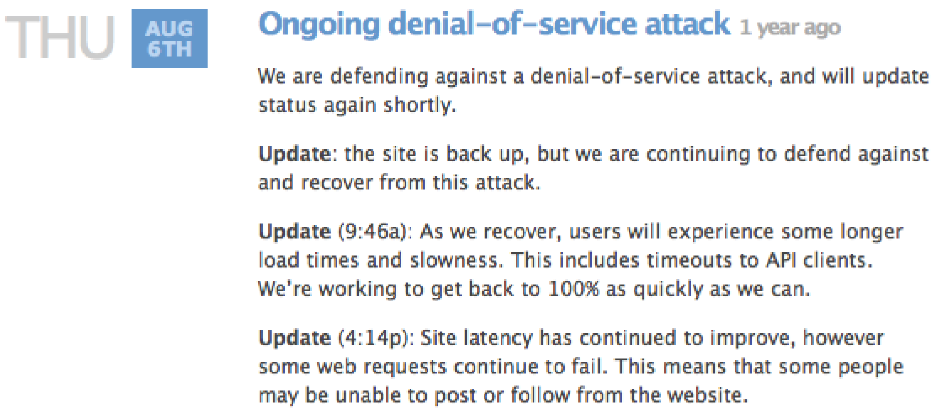


Figure 3.4: The SDLC

(botnets), which simultaneously send massive amounts of bogus data to a victim server. This kind of DoS attacks is called Distributed DoS (DDoS). This attack forces the receiving server to handle enormous amounts of request, which causes the services to become unavailable. However, customers utilizing the cloud can better scale with such an attack, because of the elasticity of the cloud. This could potentially be a costly procedure, since the cloud pricing-model is based on actual usage. When attacked, the usage of resources (e.g. network bandwidth, CPU power and storage) increases dramatically, as well does the costs. This variant of the DoS attacks is known as Economical Denial of Sustainability (EDoS). It is difficult to filter out traffic from DoS attacks, because they blend in with legitimate traffic. Twitter was attacked by a DDoS attacks on August 2009, whereas the whole service became unavailable for several hours. Figure 3.5 on page 30 shows the status Twitter posted on their blog, shortly after the attack was initiated.. This shows how powerful DoS attacks are, and with the adoption of cloud technologies, new variants of DoS will appear. In the near future, we may be witness to malicious accounts in IaaS or PaaS clouds, launching DDoS attacks with almost unlimited compute resources available. This botnet of cloud-accounts is characterized as dark clouds.


THU AUG 6TH

Ongoing denial-of-service attack 1 year ago

We are defending against a denial-of-service attack, and will update status again shortly.

Update: the site is back up, but we are continuing to defend against and recover from this attack.

Update (9:46a): As we recover, users will experience some longer load times and slowness. This includes timeouts to API clients. We're working to get back to 100% as quickly as we can.

Update (4:14p): Site latency has continued to improve, however some web requests continue to fail. This means that some people may be unable to post or follow from the website.

Figure 3.5: DDoS attack on Twitter [4]

SaaS application security:

In the SaaS model, the CSPs are responsible for securing the applications they offer. The customers, however, are responsible of the operational control of their application. This includes access and authentication control, which usually is provided as a web-based interface by the CSPs. Since the application security is transparent to the customers, a NDA is often used to ensure the customers how security is provided. The NDA encompasses architecture, design, development process and testing methodic. Moreover, some customers hire third-party companies to perform penetration-testing on the applications. However, these tests are not always allowed by the CSPs, and can be quite costly.

The access control and authentication management offered by the CSPs tend to be too simplistic for many organizations. Weaknesses are discovered on these features in many well-known CSPs. An example is when several weaknesses was found in Google Docs, a popular SaaS text editor, in March 2009 [28]. Embedded images in a document were still available after the document was deleted. Furthermore, if a user removed other users from a shared document, they could regain access to the same document without permission. This is just a couple of examples illustrating the importance of security regarding SaaS applications. Many of the major SaaS providers (e.g., Google, Salesforce.com and Microsoft) have invested in software security as part of their SDLC. However, since no industry standard exist, it is difficult to benchmark their security performances.

3.2 Data security

Data security is an important area, also when it comes to cloud computing. Many levels of the infrastructure are involved in the various cloud service models, thus several considerations must be taken. Additionally, the multi-tenant environment of the cloud makes data security crucial. People often think of encryption techniques when talking about data security. However, there are several aspects of this topic besides encryption of data-at-rest, including:

- Data-in-transit
- Processing of data (multi-tenancy)
- Data lineage
- Data provenance
- Data remanence

The encryption algorithm is of importance considering data-in-transit. Usually, only algorithms which are publicly accepted as strong are used. Furthermore, it is important to utilize safe protocols, e.g., FTPS, HTTPS, Secure Copy (SCP). These protocols are built to provide both confidentiality and integrity of the data. However, when using traditional FTP and HTTP only confidentiality is achieved, due to the use of symmetric stream ciphers.

There are many factors related to storing data securely. Encryption of data-at-rest is usually possible (thus recommended) when using IaaS cloud services. However, data used by an application running in the cloud is not encrypted. Since encryption would prevent indexing and searching of the data, PaaS and SaaS cloud services let data be unencrypted at some time during its lifecycle. Although this data is encrypted during transit and at rest, they are vulnerable for a certain time if used by an application. There is much research on the field of homomorphic encryption, which basically is processing of data without decryption.

Another aspect of data security is multi-tenancy, which makes the cloud environment dynamic. The customer's data is stored in a mixed manner, hence good authorization controls are necessary. Normally, a tokenization-scheme is used to tag and separate the data. However, exploits of application vulnerabilities could be used to get unauthorized access.

How the data of an organization is stored and encrypted is of importance, as well is the location of the data. A log-file containing metadata about where the data has been, and by which application they have been used, could be important in case of forensics or a failure. This concept is known as data-lineage. Data-lineage is extremely time consuming, and there is yet no practical implementations of it.

Data-provenance is to proof both integrity and provenance of the data. Integrity is to ensure the data has not been altered during transit by someone unauthorized. Usually, integrity is provided through a checksum added to the data, which both parties need to calculate using a secret key. To ensure provenance, the data needs to be calculated correctly. Hence, this concept is critical in scientific and financial equations, where the requirements for accuracy are especially important.

The last facet of data security is data-remanence, which is how to properly delete and remove data from a system. CSPs tend not to mention data-remenance in their security plans. For instance, if a customer of a CSP want to stop subscribing to their service, good routines for deletion of the customer's data need to be in place. Moreover, how CPSs physically destroys storage devices, is of interest from a customer's perspective.

3.2.1 CSP data security

The CSPs capture and store massive amounts of metadata, which is used for both auditing and security purposes. Log files from firewalls, Intrusion Prevention Systems (IPS) and router flow data, are examples of data stored from the network level. Furthermore, system log files (host level) and application log data (application level) are stored. The traditional CIA triad is used as a basis by the CSPs, to ensure security of the data stored in the cloud. This triad has expanded during its lifetime to include accountability and non-repudation[11]. Figure 3.6 on page 33

[11] Non-repudation is the assurance that someone is unable to deny to have received certain data.

shows a typical cloud data storage architecture.

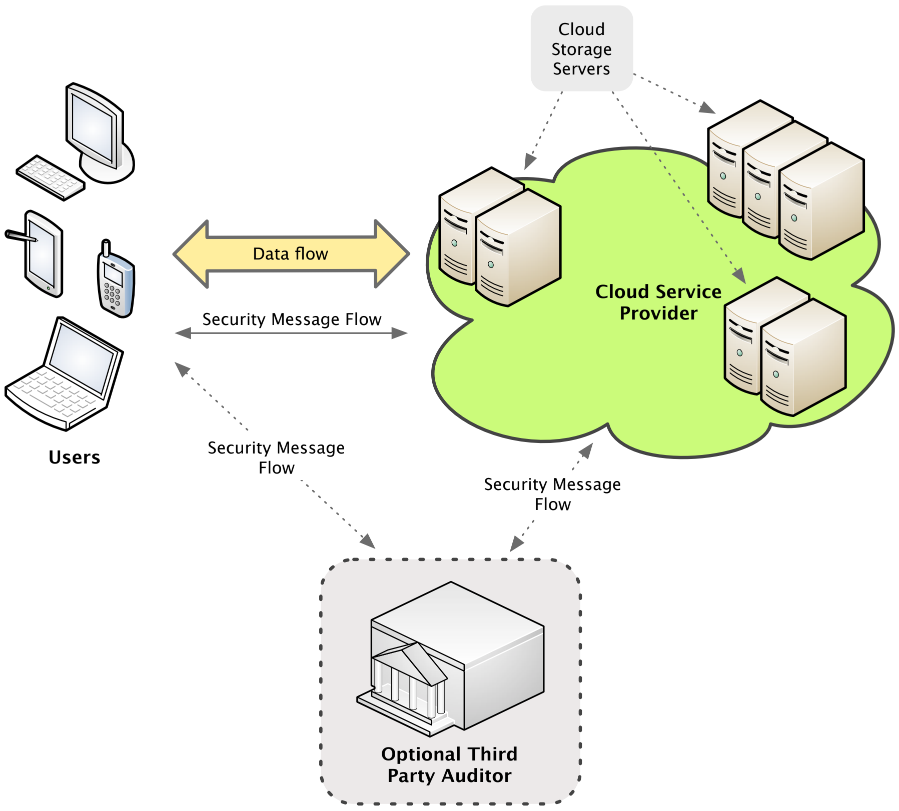


Figure 3.6: Cloud data storage architecture [5] (modified figure)

Confidentiality

There are two main facets of confidentiality; access control and encryption. Access control consists of both authentication and authorization. Usually, a CSP only uses a password and username for its authentication scheme, often with no requirements to password strength. Moreover, the authorization scheme provided by the CSPs are often too generic, where user-authorization and administrator-authorization are the only options. Many mid-size and large organizations need a more granular and customizable access control scheme, where more levels of access are allowed.

The second facet of confidentiality is encryption of customer's data. Whether the stored data is encrypted or not, depends entirely on the various CSPs. The Amazon S3, for example, does not provide encryption for customer's data at-rest. However, the customers could encrypt their data prior to uploading if they find it necessary [29]. Dropbox[12] , which utilize Amazon S3 for its storage, encrypts the customer's data with 256-bit AES[13] [30]. Numerous encryption algorithms are available, however, only those which are publicly approved by formal standards or the cryptographic community should be used (e.g., AES, 3DES[14]). Since CSPs store massive amounts of data, only symmetric encryption is practical because of its speed and efficiency. Symmetric encryption is illustrated in figure 3.7 on page 35, where a single secret key is used for both encryption and decryption. Both key size and management are critical in symmetric encryption. Some attacks (e.g., brute force) can be avoided by setting a proper key size. The recommended minimum key size depends on the encryption algorithm, for example, 112-bit should be minimum if 3DES is used. However, the length of the key is irrelevant if the key management is poor.

The key management could be provided by the CSP, outsourced to a trusted third party, or the customers handle their own keys. If the CSP provides the key management, administer the whole set of customer's keys is complex. Therefore, some CSPs ease the management task by using a single key to encrypt all data of a customer. Generally, an encryption key should only be used once to provide confidentiality, thus reuse of keys defaces security. In conclusion, the CSP manage encryption and key management differently, hence it is important to read the Service Level Agreement (SLA) thoroughly to understand how confidentiality is implemented. As an example, Dropbox has recently received a complaint [31], where a security researcher accuses Dropbox of deceiving their customers about their security program. Dropbox say they use 256-bit AES encryption when storing customer's data, thus keeping “your stuff safe”. However, they forgot to mention that their employees have access to all encryption keys, hence are in theory

[12] Dropbox is a widely popular STaaS, which helps their users upload and transparently back up files.

[13] The Advanced Encryption Standard (AES) is a block encryption algorithm, supporting key sizes at 128, 192 and 256 bit.

[14] Triple Data Encryption Standard (3DES) is DES used three times, because the key size in the original DES (56 bit) was found too weak.

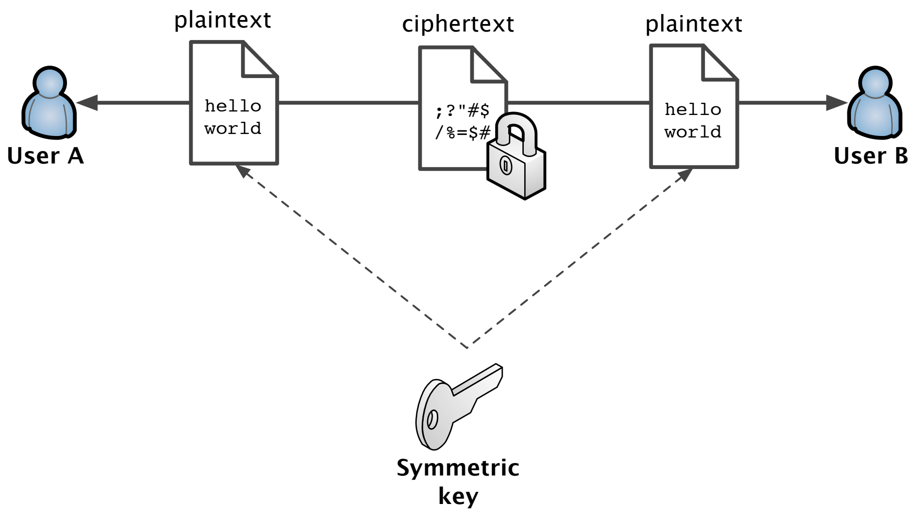


Figure 3.7: Symmetric encryption scheme

available to obtain data from any customer.

Integrity

Confidentiality does not imply integrity. Even though the data is encrypted during transit, a receiver does not know that the data is unaltered. Usually, a Message Authentication Code (MAC) is used to verify the integrity of data. The MAC is calculated with a one-way hash function (e.g., SHA1, MD5) with the data and a secret key as input, then appended to the data. The receiving part must calculate the MAC with the same inputs, then compare the two MACs to verify the integrity.

The cloud is a dynamic environment, where the location of the data changes rapidly. Therefore, traditional methods for integrity verification becomes impractical. Another concern is that the customer should be able to verify the integrity of the data while it floats in the cloud. Moreover, verification of the data should be done without the knowledge of the whole data set. These unique facets of cloud computing makes integrity a major topic of research. Homomorphic encryption has been considered as a solution. However, practical implementations are yet to be developed. The paper "Data Integrity Proofs in Cloud Storage"[32] proposes a scheme to provide Proof Of Retrievability (POR), i.e., proof of data integrity. This paper focuses on efficiency and low computational costs, both at the client and server side, which also leads to lower bandwidth consumption. However, this scheme only works with static data sets, thus further development is needed to handle the dynamic clouds.

Availability

The third asset of the CIA triad is availability, which has many risks associated with it. However, none of these threats are cloud-specific. The first risk is network-based attacks (e.g., DoS attacks), which is discussed in detail in subsection 3.1.1 on page 18. Another concern is how availability is provided by the CSPs. Usually, a certain percentage of uptime per year is put in the SLA. Table 3.1 shows the total downtime (HH:MM:SS) of typical availability percentages. Although a major cloud vendor such as Amazon is able to scale with availability attacks, they have suffered downtime caused by DDoS attacks. As a result, companies are not able to put many “9s" in their SLA.

Availability	**Per day**	**Per month**	**Per year**
99.999%	00:00:00:4	00:00:26	00:05:15
99.99%	00:00:08	00:04:22	00:52:35
99.9%	00:01:26	00:43:49	08:45:56
99%	00:14:23	07:18:17	87:39:29

Table 3.1: Percentage of uptime [2]

The facilities where the data is stored are also of importance considering availability. Which routines does the CSP have in case of a fire, or a power failure? Does the CSP offer redundancy of all customer's data? These are questions which needs to be asked, when choosing a CSP. As an example, the Amazon S3 redundantly stores the customer data in multiple locations. However, the PaaS and SaaS services of Amazon does not offer backups of data used in running instances [29].

3.3 Cloud insecurity

Cloud computing is an immature technology, which is constantly developing. Equally, cloud security is in its early age. Luckily, it seems that security has been a priority from the birth of cloud computing. However, the tremendous pursuit of moving services to the cloud, has caused many vulnerabilities to be discovered. Hackers know that cloud computing is in the beginning-phase, they try to find clever ways to exploit the immaturity.

3.3.1 Cloud-specific threats

- **Intrusion detection** – In a traditional IT architecture, a firewall can be thought of as a front door, which controls the access to the entire system. In cloud computing, however, security is much more granular. Dedicated firewalls for individual servers can be provided, typically at an extra cost. Additionally, several security applications to protect agains well-known attacks (e.g., SQL injection) are available.

- **Data location** – This is another area where cloud computing faces unique challenges. Often, the location of the data is unknown, depending on the service provider. Usually, this will not cause any problems. However, the data is bound to the legal jurisdictions of the country it is located in. If a customer's data is located in Russia, and there is a dispute on the part of the provider, accessing it could be a challenging task. The access of the data would then be controlled by russian law, thus making the process time-consuming and expensive. Many CSPs guarantee where the data is physically located, thus eliminates this security risk.

- **Regulatory compliance** – One of the most challenging topics related to cloud computing is achieving compliance with well-known security standards, e.g. Payment Card Industry Data Security Standard (PCI/DSS), HIPAA, Sarbanes-Oxley and Graham-Leach-Bliley. These standards were developed when one-server/one-application deployments were the norm, and the servers were controlled by the entity running the application. Thus, problems arise when adapting these standards into the multi-tenant environments of cloud computing [33].

3.3.2 Poisoning of VMs

Traditionally, dedicated stand-alone machines have been used in most organizations. Administrators and IT-personnel have been in charge of buying network equipment (e.g., servers), installed the necessary software, then monitored and maintained the equipment. However, with cloud computing VMs are used in a multi-tenant environment to run an organization's application. The administrator is allowed to configure the OS of a VM in any way he/she wants. Moreover, the administrator can run any desired code on the VM. Configurations made on one single VM do not affect other VMs in the same environment, because of the isolation mechanisms in PaaS. Although a VM is protected from other VMs, it could be compromised by malicious users of the same VM. An attacker could alter the VM settings, or modify the application running, to gain access to an organization's data running on that cloud. An organization is in charge of preventing unauthorized access by carefully configuring the VM.

The integrity of each VM is of importance, therefore it is crucial to know where it came from, and who configured it. As an example, the Amazon EC2 gives their customers three choices of VM selection; a fully configured Amazon Machine Image (AMI), a pool of community-shared AMIs, or providing their own AMI. The latter option is the simplest way of ensuring the requirements of an organization, however, it requires knowledge of AMI development. Choosing the Amazon-configured AMI gives an assurance of a clean VM, without hidden malicious code. If one choose to utilize a community-shared AMI, all trust is put on the creator. An adversary could easily hide code in an AMI, e.g., a rootkit[15]. Amazon is aware of this problem, and has a warning on their webpage, which informs the users about the possibility of malicious AMIs. Furthermore, they have also included a launch confirmation process, which helps users detect malicious AMIs [35]:

1. Check the SSH authorized keys file. The only key in the file should be the

[15] A rootkit is a collection of tools (programs) that enable administrator-level access to a computer or computer network [34].

key you used to launch the AMI.

2. Check open ports and running services.
3. Change the root password if it is not randomized on startup.
4. Check if SSH allows root password logins.
5. Check whether there are any other user accounts that might allow backdoor entry to your instance. Accounts with super user privileges are particularly dangerous.
6. Verify that all cron jobs are legitimate.

Even though a developer of an AMI has good intentions, he could accidentally introduce security-breaches, e.g., using an outdated library or software package, or reusing private keys. In conclusion, AMIs could intentionally or unintentionally be poisoned. Both situations could introduce vulnerabilities, which an adversary could exploit.

3.3.3 Attacks against the management console

Normally, each CSP provides a web-interface to their customer for cloud-management. The purpose of this management console is to have a centralized, user-friendly environment towards their customers. Regardless of their good intentions, management consoles introduce vulnerabilities to the cloud. If an attacker gets access to the management console, he could easily change the environment of the application running. Although this application is protected against attacks and the VM is isolated, unauthorized access to the management console could still cause considerable damage.

Most PaaS cloud providers offer a web-interface as a management console on the domain of their organization. As a consequence, all vulnerabilities on this domain could possibly affect the console. Google has realized this fact, hence their management console of their Google App Engine has very limited functionality. If a customer wants to modify or upload code to an application, he/she has to use a python script in the command line. Although this is not particularly user-friendly, it prevents damage caused by a vulnerabilities on the Google.com domain.

Unlike the Google App Engine, the Amazon EC2 gathers all management functionalities in the console. Thus, vulnerabilities found on Amazon.com domain could disturb the management console. For instance, if an adversary finds a XSS vulnerability, he/she could exploit this for an attack towards the EC2-console. Another concern regarding the management consoles is weak access-management. Usually, a username and password is all that is required to gain access to the console. An attacker could capture these fields with e.g. XSS, SQL injection or phising, then get access to the EC2-instances of the compromised account. Moreover, unauthorized access to the management console means that the attacker gets all necessary information (e.g., X.509 certificates) he needs to access the running instances of that specific user. In other words, one single XSS vulnerability found anywhere on the domain, could lead to compromised secret keys and certificates. CRSF attacks are also a threat to the management console. If an adversary gets a user to click on a malicious link, while logged into the management console, he/she could get unauthorized access. This type of attack is discussed in detail in section 4.3.1 on page 51.

Chapter summary

Cloud security has been a priority since the birth of the cloud. However, due to immaturity of the technology, adversaries have managed to disrupt services even at major CSPs. They try to take advantage of the rapid adoption of the cloud, while security still is under development. Perhaps the most difficult attacks to withstand is the DDoS attack, which can cause even a highly elastic CSP to stop functioning. Traditional threats to IT system does not magically disappear with cloud services, hence the CIA-triad is still used to evaluate security. However, ensuring confidentiality, integrity and availability in the cloud are challenging tasks. Since the cloud is constantly changing, it is difficult to verify the integrity of the data. Moreover, proofing integrity should be done without knowing the whole data-set. It is also necessary to improve the key management schemes found in many CSPs. In addition to the well-known vulnerabilities from IT systems, new threats emerge to the cloud (e.g., VM poisoning, eDOS). In the emerging future, we may encounter dark clouds, which is DDoS attacks originating from clouds. This attack could have enormous impact, because of the cloud's scalability. However, the benefits of moving services to the cloud are too severe for organizations to be frightened by plausible future threats. By better understanding how an attacker thinks, developers can enhance security, and to a greater degree withstand attacks.

4 Open source Content Management Systems (CMS)

In this chapter, CMSs are described in detail. Then, a comparison between open and closed source CMSs are made. Moreover, the security of CMSs is analyzed. Finally, common attacks towards these systems are discussed.

4.1 What is a CMS?

A CMS is a software which manages the contents (e.g., text, pictures, videos, music, documents) on a web site. One major advantage of the CMS is that it requires minimal amount of technical skill to publish a professional site. With CMS, an organization could easily develop a professional web site, without outsourcing the project. Many of the popular CMSs do not depend on programming skills, which makes it attractive to a broader public. However, having a web site requires maintenance and updates. From a security point of view, these two factors are critical to prevent vulnerabilities. Since many CMS web sites are published by people without IT-background, they are an attractive target for hackers.

All CMSs require a web host for their services. The web host provides and maintains the servers, where the CMSs are deployed. With the increasing popularity of cloud computing, some CSPs have looked into the CMS marked. As a consequence, certain open source CMSs are available as PaaS on CSPs. In other words, these CSPs have performed the installation and setup of the CMS on their cloud platform, thus offer a ready-to-use system. Then the customers are able to modify and enhance their site as they want, unaware of which mechanisms that run beneath the surface. Simplicity and transparency are tempting factors for most customers. Moreover, the cloud is a dynamic environment, which let the users scale their services on demand. Therefore, organizations are able to expand their services without buying new, expensive network equipment.

Most CMS have a front-end and a back-end. The front-end is what people see when they visit the web site, while the back-end is where the site's configurations are made. A password and username are normally required to access the back-end. Alterations to the design, publishing of articles, installation of modules, are examples of configurations to a CMS site. The content of articles is usually formatted using a rich text editor, which creates a XML, HTML or XHTML markup. Then the text is rendered using style sheets (e.g., CSS) to personalize the output. The administrator can set the access-rights for each user, hence controlling what they are able to view and modify on the web site.

The three most widely used open source CMSs are; Drupal, Joomla! and WordPress [36]. Especially WordPress has become increasingly favored, due to its simplicity. More professional users would perhaps prefer Joomla! or Drupal, because they offer more flexibility. However, Joomla! is also quite straightforward to install and maintain. WordPress is considered the most popular CMS, with Joomla! not far behind, as seen in figure 4.1 on page 45, where the popularity of the three major CMSs are compared. This figure also shows that WordPress seized the throne from Joomla!, as the most popular CMS, in 2009. The figure is constructed using numbers from Google's "Insight for Search[1] ".

[1] Google's Insight for search is a service which measures the amount of search queries on specified items, then normalizes them to a scale from 0-100 [37].

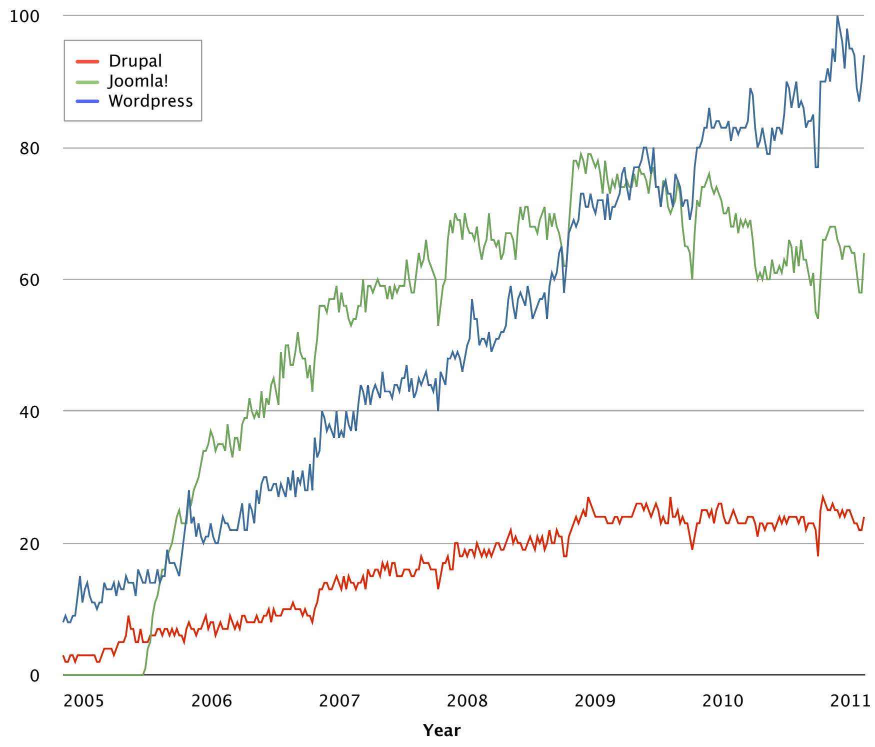


Figure 4.1: CMS trends (2005–2011)

4.2 Open source vs. closed source

An open source CMS means that everyone is able to look into how the system is constructured. The code is available to anyone, hence both developers and adversaries can make use of it. The main advantage of open source, besides the cost, is that there is a community of developers constantly working to maintain and upgrade the system. Another benefit is that it is customizable, hence an organization could implement features which separate them from the majority. Usually, it is straightforward to find documentation, online guides and "how-tos" on development and design. The open source CMS consist of a set of basic functionalities, called the "core". However, generally there exist several modules and add-ons to enhance the web site. These extensions are normally made by third-parties, hence it is necessary to ensure the security of the code. Most of the vulnerabilities related to CMSs are due to poorly coded, or malicious third-party extensions.

The main drawback of open source is the availability for anyone to find out how the system is built. Adversaries can take their time analyzing every part of the “core", trying to find vulnerabilities to exploit. If an exploit is found, this could possibly affect every system using the same code. Normally, a fix is created shortly after a vulnerability is reported. As a consequence, the customers are responsible to update regularly, hence avoid known threats. A good community is critical when choosing an open source CMS, since vulnerabilities eventually will be found. Normally, an open source CMS also has a forum, which is an important source of knowledge. Any questions regarding the CMS can be posted there, and they are usually answered quickly.

The closed source CMS is licensed, and the source code is not available. However, a closed source CMS often equates with better security. With open source CMSs, the developers spend much time securing the code. If a security-breach is found in a closed source CMS, the provider of the software is usually more than happy to assist. Since the software is licensed, the barrier to entry is higher. The main advantage of open source is perhaps the main drawback of the closed source, namely the community support. The numerous amount of developers contributing on every aspect of the CMS is valuable. Hence, the popularity of the closed source CMSs are limited [38].

4.3 Security in open source CMSs

In software systems, vulnerabilities are caused by weaknesses at either the design or implementation level. Although a vulnerability exists, it is not necessarily exploited by adversaries. There are many tools available on the Internet, which automates such attacks. As a result, script-kiddies[2] use these tools and causes

[2] A person, normally someone who is not technologically sophisticated, who randomly seeks out a specific weakness over the Internet in order to gain root access to a system without really understanding what it is he/she is exploiting because the weakness was discovered by someone else [39].

disruptions on vulnerable web sites. Since CMSs are often used by non-technical users, they tend to be attractive targets. Therefore, CMSs are a major source of vulnerabilities. Moreover, since the source code is available for everyone, the attackers are aware of how the system works. Open source CMS operate in a hostile environment, and the threats include:

- **Data manipulation:** Manipulation compromises the integrity of the data. Common attacks in this category are SQL injections and parameter manipulation.

- **Accessing confidential data:** In this attack, an adversary gain unauthorized access to confidential data by utilizing SQL injections or XSS attacks.

- **Phising:** An attacker gathers confidential data (e.g., bank account information, passwords) by sending emails to people, pretending to be a service they use. For instance, an adversary sends a phising-mail to a customer of eBay, requesting his username and password for their site's maintenance. Phising could also be accomplished by utilizing XSS on a vulnerable CMS, to place malicious input forms and gather confidential data. Phising is illustrated in figure 4.2 on page 48.

- **Code execution:** If a CMS does not carefully validate input, adversaries could exploit this to execute code. If an input field assume graphic-files, only files with certain extensions and sizes should be allowed (e.g, .jpg, .png). This attack could possibly harm all applications running on the server.

The CIA-triad, as discussed in section 3.2.1 on page 32, is a good measurement for security. Implementing the three facets of the triad, could prevent numerous attacks. However, because the environment of an open source CMS is quite complex, especially in the cloud, the developers face many challenges. For instance, ensuring availability is troublesome because of the powerful DDoS attacks seen today.

Open source CMSs utilize several technologies. Certain technologies are common for most CMSs, e.g., MySQL, HTML, XML and CSS. Furthermore, there are four main technologies, in which the various CMSs fall under; PHP, Java, Perl and Python. An illustration of these technologies and related open source CMSs, are shown in figure 4.3 on page 49. To add more complexity to an open source

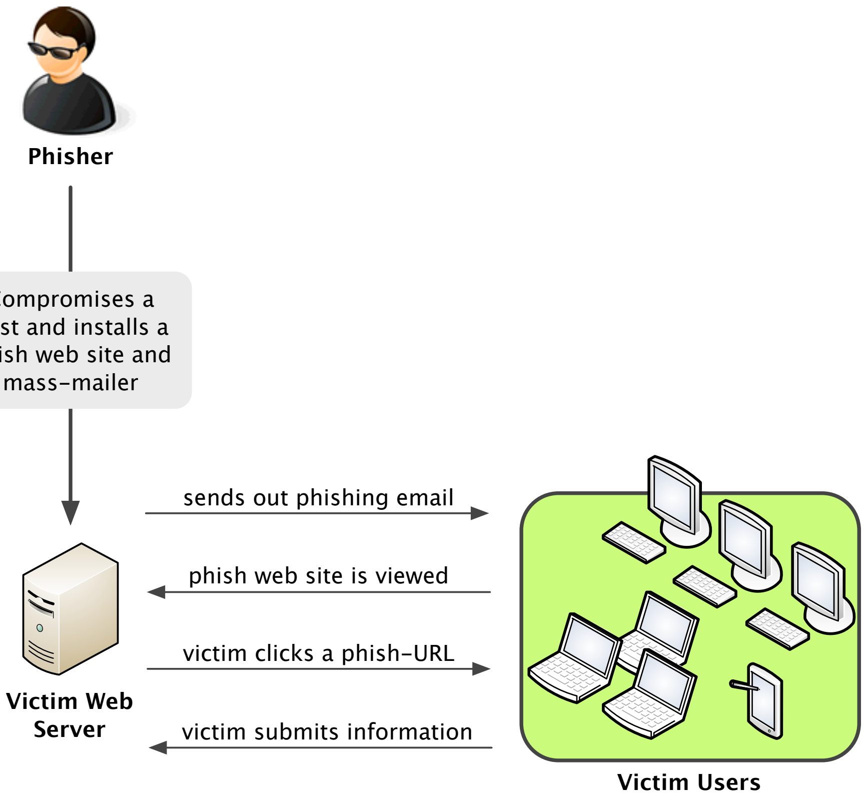

Phisher
Compromises a host and installs a phish web site and mass-mailer
sends out phishing email
phish web site is viewed
victim clicks a phish-URL
victim submits information
Victim Web Server
Victim Users

Figure 4.2: How a phising-attack is executed

CMS's security program, these technologies also have spesific vulnerabilities. For example, PHP handles global variables differently than, e.g., Python.

4.3.1 Common attacks towards CMSs

Web applications in general are tempting targets, because of several known vulnerabilities. OWASP has made a list over the top 10 web applications threats in 2010, which is listed in appendix A on page 87. Moreover, if the application is running on the cloud, more possible threats could be added to the equation. Since, the open source CMSs live in an unfriendly neighborhood, it is necessary to take precautions.

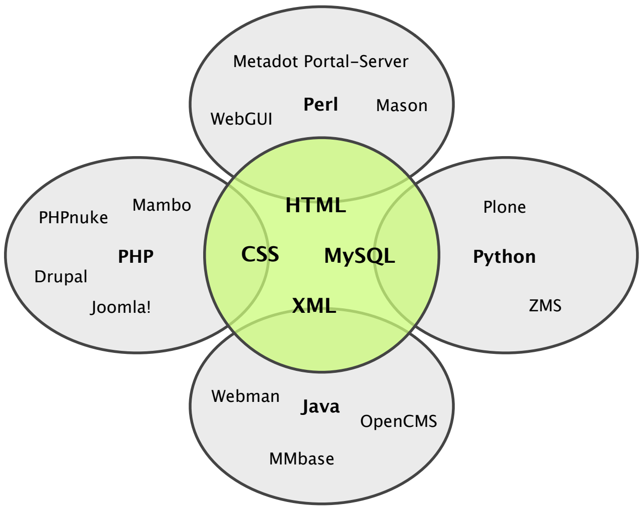


Figure 4.3: Open source CMSs and related technologies [6] (modified figure)

Numerous attacks have similar characteristics. Thus adversaries makes use of attack patterns, which are generalizations of the necessary steps, to perform a successful attack. They consist of several phases of discovery and exploitation. Furthermore, the patterns are usually made available for other adversaries, so they likewise can carry out similar attacks. The pattern itself contains useful information, such as resources needed, time consumption and techniques. If the attack is successful, the adversary may obtain confidential customer data. If the open source CMS is an e-commerce web site, then the customer data could hold valuable credit-card information. Another critical consequence of an attack, is the loss of confidence to the service. If a CMS fails to provide the adequate security, their reputation could be harmed. As a result, customers could move to a competitive service. An attacker could also damage the reputation of a CMS by altering the company's website, e.g., if a security vulnerability allows the attacker to add bogus content to the site.

SQL injections

SQL injections are improperly filtered input which is sent to the SQL server. This input could be SQL queries, which could possibly access sensitive data. An adversary could use escape characters to include SQL queries in an input field. For instance, if a malicious user append `'1'='1'` to an input field, this could lead to unwanted disclosure of data. Since the boolean expression `OR '1'='1'` is always true, the query in which the expressions is appended would also be allowed. The adversary could exploit this, by requesting sensitive data in a query, e.g., usernames and passwords, as shown in the SQL-query below:

```
SELECT username,password FROM users WHERE name = 'John Doe'
OR '1'='1';
```

The various technologies used by open source CMSs have mechanisms to prevent SQL injections. These mechanisms examine input strings to prevent exploits of escape characters. For example, PHP uses the function `mysql_escape_string()` to mask all kinds of special characters.

Cross-site Scripting (XSS)

This vulnerability occurs when an application takes untrusted input data and sends it to the web browser, without proper escaping and validating. An adversary could exploit this vulnerability by including script code (e.g., Javascript) on a web page. Proper mechanisms for always treating output as text, are necessary to prevent script code from being executed in a browser. Since applications uses different browser side interpreters (e.g., ActiveX, Flash, Silverlight), detection of XSS is challenging. Consider the following HTML snippet:

```
(String)page+="<input name='creditcard' type='text' value="'
+request.getParameter("CC")+"'>";
```

The attacker could modify the CC parameter in the web browser to include the following script:

```
'><script>document.location='http://www.attacker.com/cgi-bin
/cookie.cgi?foo='+document.cookie</script>'
```

This script would forward the victim to the attacker's malicious web site, hence allows the attacker to hijack the victim's current session [40].

Cross-site Forgery Requests (CRSF)

A CSRF is an attack where a user performs unwanted actions on a vulnerable application, in which he is currently authenticated in. An adversary could trick a user to load a page with a malicious request, then inherit the victim's identity and privileges to perform actions on the vulnerable application. The application would think that the requests made by the adversary, actually are legitimate request from the victim. Social engineering, e.g., sending a link via email or chat, could fool the victim to open the "evil" web site. Links and forms that involves state-changing functions are the main targets for CSRF attacks, hence special attention must be given to these. As an example, consider the following state-changing request:

```
http://domain.com/vulnerableApp/transferMoney?amount=1500
&account=123456789
```

This request does not contain any secrets, however, an adversary could exploit this request with a CSRF attack. The adversary wants to transfer the money to his own account, instead of the victim's account. Hence, the adversary embeds the following image request on various web sites under his control:

```
<img src="http://domain.com/vulnerableApp/transferMoney?
amount=1500&account=attackerAccount#" width="0" height="0"/>
```

If the victim visits one of the malicious web sites, while still is authenticated at domain.com, the attacker could forge request that includes the victim's session info. As a result, the vulnerable application authorizes the malicious request, because it appears to be the victim.

The easiest way to prevent CSRF attacks is to include an unpredictable token in the body or URL of every HTTP requests. These token should be unique to each user session, or unique to each request. A good practice is to include the token in a hidden field. Then the token is sent in the body of the HTTP request, thus not exposed in the URL. The token could also be included in the URL. However, then it could be compromised due to exposure to adversaries [40].

Chapter summary

CMSs are widely popular services, especially open source CMSs, because of their low cost and simplicity. There are several additional advantages of choosing an open source CMS, e.g., good documentation, online-guides, community of developers. However, security is a big concern. Nowadays, many open source CMSs offer their services as a PaaS. Therefore, security threats to the cloud should also be considered. Adversaries have found several ways of exploiting vulnerabilities, particularly on the application level (e.g., CSRF, SQL injections, XSS). The huge amount of non-technical CMS-users make these systems attractive targets. However, most of the attacks could easily be prevented by simple means, e.g., proper validation and escaping of input. Usually, the vulnerabilities are not found in the "core" of the CMSs, thus examining extensions and plugins for vulnerabilities are of importance.

5 Joomla! Security

This chapter gives an overview of Joomla! and its functionalities. The latest upgrade of Joomla! is discussed from different perspectives. Then, the framework and some security related functions are investigated, i.e., how the Joomla! “core" is protected against attacks. After that, the strength and weaknesses of extensions are looked into. Finally, how Joomla! can benefit from cloud computing is examined.

5.1 About Joomla!

> *“Joomla is an award-winning content management system (CMS), which enables you to build websites and powerful online applications. Many aspects, including its ease-of-use and extensibility, have made Joomla the most popular website software available. Best of all, Joomla is an open source solution that is freely available to everyone.”* [7]

Joomla! is one of the most widely used CMSs today and powers 2.7% of the entire web [7]. The first version (Joomla! 1.0.0) was released in September 2005. Well-known companies that uses Joomla! include Harvard University, Linux, Citibank and IKEA. Joomla! is an open source CMS, hence it has no license costs. However, Joomla! requires a service provider to be deployed on. There are numerous of CSPs which offer Joomla! as a PaaS, e.g., Cloudaccess.net, Uptimehost.com, Hoodukucloud.com, etc. These CSPs have pre-installed Joomla! and maintains the system transparently. Furthermore, the cloud can scale on demand, allowing the Joomla! website to grow without having to buy expensive network equipment. Although the installation procedure of Joomla! is quite straightforward, many customers want to solely focus on the contents of their website.

The reason for Joomla's popularity is manifold. However, the simplicity of building a professional working website, is a major factor. There are thousands of professionally designed templates to Joomla!, which are installed with a few clicks. With a little knowledge of CSS and HTML, customers could easily customize the templates themselves. Furthermore, the Joomla! community is another attractive asset. The community provides help with every aspect of the system, e.g., security related questions, documentations, development guides. The strength of the community is of importance when choosing an open source CMS.

Joomla! allows the users to expand their system easily. Its framework consists of three layers, as seen in figure 5.1 on page 55; framework, application and extension. This framework is based on object-oriented design, which makes the "core" very agile. As a consequence, customers can easily expand and customize their system with modules, templates and components. The Joomla! community has built around 7000 extentions [41], which is installed in a few seconds. These extensions include forums, shopping carts, online storage, photo galleries, to mention a few. However, poorly developed extensions could lead to security vulnerabilities. Moreover, malicious users can hide evil code in their extensions, which could compromise a system. Although extensions could introduce certain security risks, some of them are actually built to prevent them. Several extensions are security related, e.g., enhance authentication, access control and site security. For instance, if a customer wants increased authentication control, he could install a CAPTCHA[1] module.

The access control in Joomla! gives control of who is allowed to view specific content, and who is able to edit or publish content. The latest version of Joomla! (version 1.6) offers a more granular access control than the prior versions. For example, sports-journalists could be allowed to only publish content in the sports-section of a newspaper.

[1] A CAPTCHA is a program that protects websites against bots by generating and grading tests that humans can pass, but current computer programs cannot [42].

The Joomla! platform consists of a front-end and a back-end system. The front-end is how the website is viewed to the public, whereas the back-end provides an interface for customizing the system. Publishing of articles, installation of extensions and access control, are some examples of assignments accomplished in the back-end. The intuiative interface of the back end makes it easy for non-technical users to perform advanced customizations and enhancements to the website.

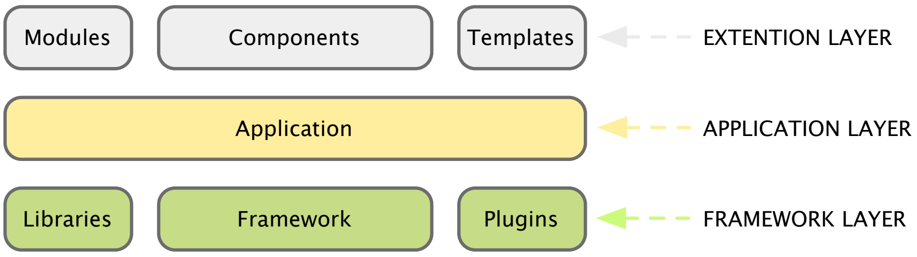


Figure 5.1: The Joomla! framework [7] (modified figure)

5.1.1 What is new in Joomla! 1.6?

The latest release of Joomla! (version 1.6) arrived on January 10th 2011 [43]. This update had many improvements over the previous version, e.g., enhanced Access Control List (ACL), SEO[2] and back-end. A significant feature of Joomla! 1.6 is that is does not work with "old" web host. As a minimum requirement, the servers must run PHP 5.2 and MySQL 5.0. Additionally, extensions made for earlier versions of Joomla! are not compatible with version 1.6., due to a different architecture. As a result, users of Joomla! 1.6 avoid web hosts with outdated servers and the corresponding vulnerabilities. Moreover, developers of version 1.6 extensions can benefit from the new features, e.g., granular ACL, automatic updating and bundling of components.

[2] Search Engine Optimization (SEO) is to tweak a website to rank high in search engine returns. These tweaks is done by optimizing certain elements in the HTML code of each page, e.g., meta-keywords.

Designer

From a designer's perspective, the new release of Joomla! introduces many attractive features. HTML5 is supported, and every output files are written in XHTML[3] 1.0 strict. Therefore, the designer is able to utilize the latest of technologies to create dynamic websites. Furthermore, it is easier to modify specific pages, or just alter a certain module within a page. Moreover, it is possible to add meta-keywords and meta-descriptions to elements, to improve SEO. This feature has received increasing amount of attention in the recent years, and organizations even hire dedicated personnel to enhance their SEO. Since websites appearing on the first page on a search engine's result list are more likely to be visited, the economical benefits are significant. Another feature included in version 1.6 is automatic updates of extensions, hence the users can keep up to date with just a click of a button. Previously, the users had to manually download updates from the developer's websites. As a consequence, many users had outdated extensions with exploitable vulnerabilities.

Developer

A new feature for developers in Joomla! 1.6 is the possibility to bundle components with the corresponding language files. Then the developers can create bundled packages, which is installed with one click. Moreover, if the developer adds the automatic update functionality, the users can easily keep track with the latest updates. Because of a different architecture than earlier, extensions made to previous versions of Joomla! are not compatible with version 1.6. However, this ensures that extensions utilize the new features of version 1.6, hence provides a more secure and user-friendly environment towards the customers. Outdated, vulnerable add-ons have probably been the main cause of security risks in Joomla!. However, making it more intuitive for users to keep their add-ons up to date, can drastically reduce this security-threat. Additionally, developers can implement a more granular ACL, thus providing a more secure environment.

[3] XHTML is a family of current and future document types and modules that reproduce, subset, and extend HTML 4. XHTML family document types are XML based, and ultimately are designed to work in conjunction with XML-based user agents [44]

Administrator

Joomla! 1.6 provides a more user-friendly interface to the back-end, where simplicity clearly is a focus. However, the most noteworthy upgrade is the new ACL, which gives a range of opportunities [45]:

- Unlimited user groups
- A user can be assigned to multiple groups.
- Any combination of groups can be assigned to any access level.
- Access permissions can be set at multiple levels in the hierarchy; site, component, category and object.
- Permissions can be inherited from parent groups and parent categories for faster user management.

The granularity the new ACL presents, allows for more flexibility for mid-size to large organizations. For instance, a specific user group is only allowed to publish articles within a certain site. Hence, this group can only view their site in the back-end.

In conclusion, the newest version of Joomla! introduces important features for the designer, developer and the administrator. The most noteworthy features are enhanced ACL and automatic extension updates. Furthermore, the source code itself is reduced by over 30% compared to the previous version. Although the core has shrunk, the total amount of comments are higher than before. This makes it more intuiative for developers to understand the code, thus easier to customize. Since extensions prior to version 1.6 are not compatible with the new architecture, developers can utilize the new features in their extensions. However, Joomla! still has no validation of extensions published in their community. As a consequence, vulnerable extensions are still available, thus a major threat to Joomla! users. These insecure extensions could either be caused by poor code, or by malicious users. Developers are encouraged to use the Joomla! API, which has several advantages concerning security (e.g., functions for input validation, prevention of forgeries, etc.). Moreover, users of Joomla! 1.6 can more easily update their extensions, hence preventing vulnerabilities to be left open. Anyhow, vulnerable extensions do not magically disappear with Joomla! 1.6.

5.2 How security is provided in the "core"

The heart of Joomla! is the "core", which is an alias for the source code. Since Joomla! is an open source CMS, anyone can download the source code and learn how security is implemented. Therefore, it is necessary that the "core" is properly coded, without vulnerabilities to exploit. Although Joomla! 1.6 introduces several new functionalities, the "core" is significantly reduced in number of code-lines. The Joomla! framework contains a number of security related functions, e.g., input validation, password encryption. Proper validation and filtering mechanisms are of importance, since the majority of attacks are XSS attacks and SQL injections.

5.2.1 Protection against common attacks

The Joomla! "core" provides various functions for input filtering. Some of these functions are based on blacklisting or whitelisting, as shown in listing 5.1. This shows various HTML tags which will be filtered out at input fields or URL parameters. However, adversaries are not restricted to write the HTML tags exactly as they are written in the blacklist. The HTML tag "body" could be written in several ways, e.g., "bOdy", "b o DY", etc., thus an attacker is able to trick the this blacklist.

Listing 5.1: Blacklisting

```
/**
 * @var array A list of the default blacklisted tags.
 * @since 1.5
 */
var $tagBlacklist = array ('applet', 'body', 'bgsound','base', '
    basefont', 'embed', 'frame', 'frameset', 'head','html', 'id',
    'iframe', 'ilayer', 'layer', 'link', 'meta','name', 'object',
    'script', 'style', 'title', 'xml');


/**
 * @var array A list of the default blacklisted tag attributes
 * @since 1.5
 */
```

```
var $attrBlacklist = array ('action', 'background', 'codebase', '
    dynsrc', 'lowsrc');
```

Another interesting issue is found in the same source code file, regarding XSS prevention. As listing 5.2 shows, the framework assumes that HTML tags used in XSS attacks always starts with a '<'. However, this is not always true. As a fact, many XSS cheat-sheets[4] suggest to use other characters. Hence, adversaries are able to go undetected through the filtering mechanisms, when inserting their XSS attack. Instead of using the '<' character, they can simply start the XSS string with, e.g., `"><script>alert(document.cookie)</script>` [46].

Listing 5.2: Filter mechanism

```
// Is there a tag? If so it will certainly start with a '<'
$tagOpen_start  = strpos($source, '<');
```

One of the most common attacks against web applications is SQL injections (see subsection 4.3.1 on page 49). This type of attack takes advantage of poorly filtered input, which allows the attacker to inject malicious queries to the database. Thus, it is important to correctly escape and quote SQL statements when they are constructed. Numeric fields, however, should not be quoted when using MySQL. They should be typecasted as either integers or floats. The Joomla! framework provides a function for escaping strings; `JDatabase::getEscaped()`, which can be seen in listing 5.3. This method utilizes two PHP functions; `mysql_real_escape_string()` and `addcslashes()`. The first PHP function escapes SQL-sensitive characters (\x00, \n, \r, \, ', ", and \x1a) [47], while the second function escapes user-defined characters. The Joomla! framework use this latter function escape the percentage-symbol, which is used in some SQL statements (e.g., LIKE).

Listing 5.3: Escaping of strings

```
/**
 * Get a database escaped string
 * @return string
 */
public function getEscaped($text, $extra = false)
{
```

[4] A XSS cheat-sheet is a collection of methologies to create XSS attacks.

```
    $result = mysql_real_escape_string($text, $this->_connection)
        ;
    if ($extra) {
        $result = addcslashes($result, '%_');
    }
    return $result;
}
```

Joomla! has included a set of functions for protection against CSRF attacks. As mentioned in subsection 4.3.1 on page 50, CSRF attacks are basically hijacking of a user's session. A simple way of avoiding these attacks is to utilize tokens, which are randomly generated strings, i.e., a unique key. The Joomla! framework has a function for creating tokens, as seen in listing 5.4. This function generates a 32 characters string, consisting of hexadecimal characters. The string, concatenated with the session-name, is then encrypted using a MD5 hash function. When a form is sent, a token is included in a hidden field, and a copy of the token is placed into the user's session. Hence, it is possible to validate the token using the function, `JRequest::checkToken()`, as shown in listing 5.5 on page 60. These functions are illustrated in figure 5.2 on page 62. Moreover, this function redirects a user to the site's front page if the session has expired.

Listing 5.4: Creation of tokens

```
/**
 * Create a token-string
 * @return string generated token
 */
protected function _createToken($length = 32)
{
    static $chars    = '0123456789abcdef';
    $max             = strlen($chars) - 1;
    $token           = '';
    $name            = session_name();
    for ($i = 0; $i < $length; ++$i) {
        $token .= $chars[ (rand(0, $max)) ];
    }
    return md5($token.$name);
}
```

Listing 5.5: Function which checks for a form token in the request

```
/**
 * Checks for a form token in the request.
 * Use in conjuction with JHtml::_('form.token').
 * @return boolean True if found and valid, false otherwise.
 */
public static function checkToken($method = 'post')
{
    $token = JUtility::getToken();
    if (!self::getVar($token, '', $method, 'alnum'))
    {
        $session = JFactory::getSession();
        if ($session->isNew()) {
            // Redirect to login screen.
            $app = JFactory::getApplication();
            $return = JRoute::_('index.php');
            $app->redirect($return, JText::_('
                JLIB_ENVIRONMENT_SESSION_EXPIRED'));
            $app->close();
        } else {
            return false;
        }
    } else {
        return true;
    }
}
```

5.2.2 Passwords

A Joomla! website requires a password for both users and the administrator. At the front end, certain components utilize username and passwords. Similarly, the back end has a login-module to authenticate the users. The Joomla! framework has a password generating function, to help users constructing passwords. However, this generator only uses letters and digits. To strengthen the password, symbols should be added to mix. The length of the password is also of importance and should be a minimum of eight characters, which the generator fulfills. Adversaries use dictionaries to brute force passwords, hence words should not be used. These

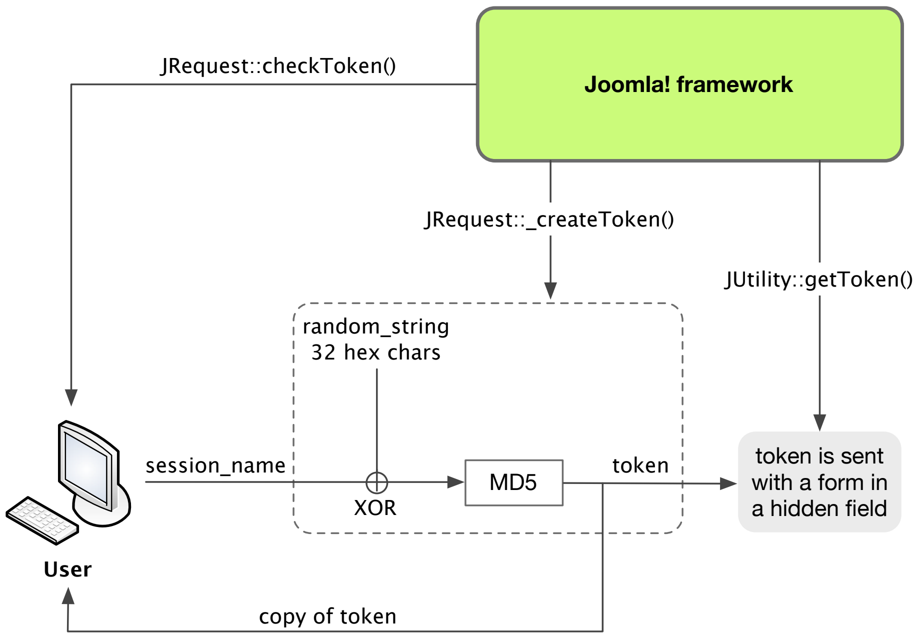


Figure 5.2: How tokens are used by the Joomla! framwork

dictionary-attacks utilize various dictionaries with certain “rules" applied, e.g., append a digit after the words, first letter is a capital letter, etc. As a result, adversaries are able to crack relatively strong passwords. However, these type of attacks are extremely time consuming. If a password consists of randomly chosen digits, letters and symbols, and is at least eight characters in length, it is unfeasible to crack it.

Weak passwords are a major threat to users of any Internet service. Surprisingly often, “1234" or “password" are used as passwords. Since many users have such passwords, it is critical to hide the login-page at the back-end. However, every Joomla! website have its back-end in “example.com/administrator" directory. As a consequence, it is easy for an attacker to start cracking the login module. Some extensions add an extra layer of security, by requesting a password to access the back-end. Furthermore, it is possible to change the name of this folder (e.g., “/admin-user", “/4dmInI$tr4t0r") to make it more difficult for adversaries to find.

The Joomla! framework uses MD5 to hash passwords in their database. The MD5 encryption algorithm is 128-bit, usually represented by 32 hexadecimal characters, and was developed by Ron Rivest in 1992 (see appendix F for more details on how the MD5 hash algorithm works). Although MD5 is one of the most widely used cryptographic hash function, it has been proven weak against several attacks (e.g., collision attacks) [48]. Furthermore, every cryptographic hash function suffers from the birthday paradox. This principle gets its name from the surprising fact that in a group of 23 people, the probability of two persons having the exact same birthday is greater than 50% [49]. Therefore, hash-algorithms are designed to produce an output twice the size of the cipher-key they are intended to be used with. For instance, to find a collision in a 128-bit MD5 hash would take on average 2^{64} hash operations. To avoid this attack, the output length of the hash function must be large enough, thus the attack becomes computational infeasible. When using the 128-bit MD5 hash function, additional enhancements (e.g. salt[5]) are needed to provide sufficient security.

The passwords stored in the Joomla! database are hashed with a 32 character salt, which is appended to the password string. The salt is a randomly generated 32 character string, consisting of hexadecimal characters. Since the final encrypted password consists of two separate variables, it becomes more difficult for an attacker to crack it. As seen in listing 5.6, the default encryption algorithm is MD5-hex, however, other algorithms are also available (e.g., SHA, DES, Blowfish). Some password systems require to prepend the type of encryption used. Therefore, as seen in listing 5.6, a `$show_encrypt` option is available.

Listing 5.6: Password encryption

```php
/**
 * Formats a password using the current encryption.
 */
public static function getCryptedPassword($plaintext, $salt = '',
    $encryption = 'md5-hex', $show_encrypt = false)
{
    // Get the salt to use.
    $salt = JUserHelper::getSalt($encryption, $salt, $plaintext);

```

[5] A salt is a random string added to a cryptographic function to enhance security.

```
    // Encrypt the password.
    switch ($encryption)
    {

    // ...other encryption cases

    case 'md5-hex' :
    default :
        $encrypted = ($salt) ? md5($plaintext.$salt) : md5(
            $plaintext);

    return ($show_encrypt) ? '{MD5}'.$encrypted : $encrypted;
    }
}
```

5.3 Choose your extensions wisely

The huge variety of extensions is an advantegous feature of Joomla!, compared to its competitors. On joomla.org, there is an extensive list of extensions, divided into several categories. Over 7500 extensions are registered at this point (May 2011) [50]. Examples of extensions are CAPTCHA modules, social media components, forums, shopping carts, etc. Anyone could create and publish extensions, which leads to a security risk. Joomla! has no mechanisms for validating each extension, thus poorly coded add-ons are difficult to discover. A worst-case scenario is if a malicious user hide evil code (e.g., root-kit) inside a seemingly harmless extension. The majority of vulnerabilities in Joomla! is found in extensions. Even though it introduces a great risk, it is also one of the drivers to choose this CMS.

Joomla! has its own security team, which constantly searches for vulnerabilities or possible security issues with every aspect of this CMS. If a vulnerability is found in an extension, usually an an update with a security-fix is quickly released. Adversaries are able to make smart searches, where they find Joomla! websites with specific vulnerable components. For instance, if a certain forum-component is vulnerable against a XSS attack, an adversary could automate an attack against numerous websites with this component installed. Hence, it is critical to keep all extensions up to date. With the new automatic updating mechanism in Joomla! 1.6, this problem could be drastically reduced. However, having to many extensions could complicate the task. Therefore, extensions which are seldom used, should be removed. Joomla! provides a vulnerability list, which is available as a RSS feed. This helps users to quickly react to potential vulnerable components, if they have one of them installed.

5.4 The cloud and Joomla!

One of the main reasons to the popularity of Joomla!, is that even non-technical customers could easily create and publish professional customized websites. Cloud computing makes Joomla! even more easy to use. Usually, the customers were required to deploy Joomla! on a host, using Apache web server and a MySQL database. This could be a frustrating task for non-technical customers. However, Joomla! can be utilized as a PaaS. Hence, the CSP has already installed and configured Joomla!, and offers a pool of resources to their customers. Moreover, the CSP normally provides good security mechanisms (e.g., firewall, backups). With a click of a button, customers are able to start creating their own professional website, in a scaleable and safe environment. Many CSPs have realized that open source CMSs are perfect for the cloud environment, and launched them as PaaS. In figure 5.3 on page 66, an architecture of a CSP which offers Joomla! as a PaaS is illustrated.

On Joomla!'s official website, there is an advertisement to a CSP, which offer Joomla! as a PaaS. It is also possible to try this service freely for 30 days. Thus, customers can start creating their website, and find out if the CMS fulfills their demands. The payment model in the cloud is usually based on a monthly or a yearly fee. The scalability of the Joomla! instances depends entirely on the infrastructure of the CSP. For example, Cloudaccess.net provides unlimited bandwidth using a 10Gbps fiber ring as backbone. Furthermore, the memory capacity is also unlimited. The variables deciding the monthly cost are storage and CPU capacity. However, since there are no software, hardware or maintenance cost, most companies can significantly reduce their expenses by moving to the cloud. Moreover, the CSPs often manage the software updates on their platform. Hence, the users can solely focus on their website, instead of, e.g., server configurations, troubleshooting

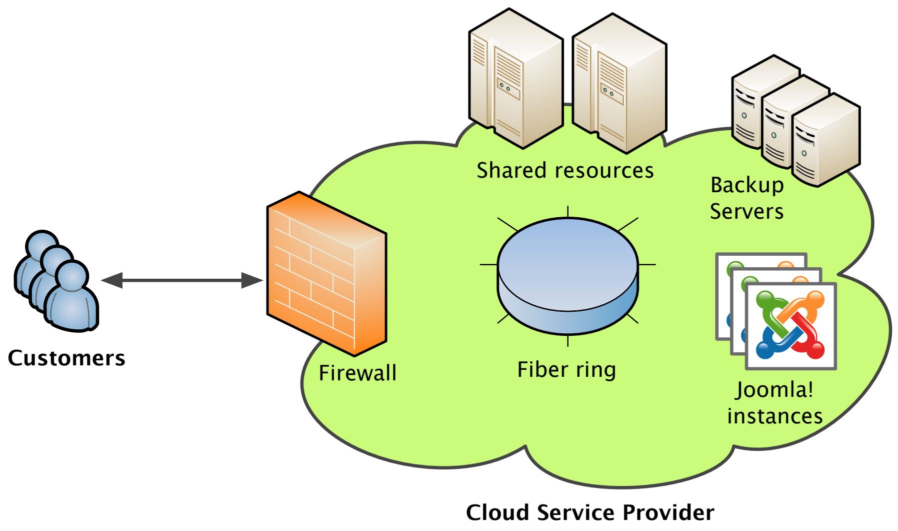


Figure 5.3: A typical architecture of a CSP, which offers Joomla! as a PaaS

virtualization problems.

The CSPs have a SLA, which encompasses various aspects of QoS, e.g., availability. Availability is a critical asset, which is difficult to guarantee with small margins. Due to the increase of DoS attacks, CSPs have problems with providing several nines in their SLAs. For instance, Cloudaccess.net states in their SLA that [51]:

> *"If a third party not associated with CloudAccess.net initiates a "Denial of Service" or other form of disabling attack against our Joomla! clusters or major portions of our network, CloudAccess.net will do everything in its power to stop the attack, but cannot guarantee a resolution time."*

This citation is listed under network and power exclusions, which is beyond the scope of the CSP, thus is not a part of the SLA. In other words, if a major DDoS attacks disrupt their services for a significant amount of time, it is not considered as downtime in terms of the SLA.

Commonly, the CSPs offer a web interface as a management console towards their customers. This console is used to manage the environment, in which the application is running. As discussed in section 3.3.3 on page 39, these management consoles are highly attractive targets for adversaries. Since most of these web interfaces operate on the same domain as its PaaS service provider, all vulnerabilities on the domain also affects the management console. In other words, one single XSS vulnerability on the domain could cause a compromised management console. Another concern regarding the management console, is weak passwords. If an attacker is able to launch a successful dictionary attack, or brute force attack, he could easily compromise the secret keys and certificates of that user. Although the Joomla! framework has protective functions for CSRF attacks, they only apply to the Joomla! back-end. It depends on the various CSPs, whether they have implemented the proper security mechanisms in their code or not.

Chapter summary

Joomla! has been, and still is, one of the most popular open source CMSs on the marked. It is difficult to pinpoint the main reason for its wide adoption, however, simplicity and ease of use may have something to do with it. Furthermore, this PHP-based CMS has a community of developers, which are willing to answer to any questions regarding Joomla!. The numerous amount of extensions is also a driver to choose this CMS. With just a click of a button, one is able to install a component that enhances the functionality of the website (e.g., forum, shopping cart). However, these third-party extensions introduce some security concerns. Joomla! does not validate each and every extension which is available on their website, hence poorly coded and malicious add-ons are easily accessible for the users.

The latest version of Joomla! has some significant improvements, including a more granular ACL, automatic updates and a more user-friendly back-end. Extensions are now updated with a single click in the back-end, which makes it easier for the customers be up to date. However, vulnerabilities do not magically disappear with this new release. Even though the Joomla! core provides several mechanisms to prevent attacks, adversaries still manage to exploit vulnerabilities on Joomla! websites. A majority of these attacks are due to vulnerabilities in poorly coded extensions, and the lack of maintenance of websites. Moreover, weak passwords also leaves many websites vulnerable to attacks.

Cloud computing makes open source CMSs even more straightforward to use, since Joomla! already is deployed on their servers. Furthermore, the maintenance and updating of software are managed by the CSP. The website on a cloud can easily scale, because of the pool of resources. This is an attractive solution to many customers, since the cost of getting a working professional site is minimal. The cloud introduce some new security issues (e.g,. management console attacks, VM poisoning). However, the CSPs provide firewalls and sometimes dedicated security software which makes attacks more difficult. Nonetheless, DoS attacks are difficult to prevent, thus the availability on the website is affected. DDoS attacks have become increasingly powerful during the last few years, and with the elasticity of the cloud, they have the potential to become even more extensive.

6 Vulnerability testing on Joomla!

In this chapter, various methods of vulnerability testing are analyzed on a Joomla! setup. The tools (BlindElephant, Nmap, OWASP Joomla! vulnerability scanner), used in these tests are described in detail. Furthermore, the results and their significance are discussed.

6.1 The Joomla! test-setup

The vulnerability testing was performed on a Joomla! 1.5 PaaS website. The goal of this investigation was not to find specific version's vulnerabilities, however, give a brief overview of security on default Joomla! websites. The tools utilized in the analysis, are publicly available as open source. Hopefully, this will give an insight in how an attacker thinks, and how he/she might extract valuable information from websites. The tests performed on the Joomla! websites, includes penetration testing (pen-tests), fingerprinting and injection-scanners.

On the official homepage of Joomla!, two options for getting a fully working demo of Joomla! are available. The first option is a pre-configured virtual application, which can be run locally. The second option, is a pre-configured Joomla! 1.5 web-site deployed on a PaaS. This CSP gives a 30-days trial to explore the functionalities of Joomla!, without any costs. The analysis done in this chapter is based on the PaaS option with a default setup, i.e., no configurations were made and no extensions were installed. This CSP protects the data center with a Fortinet[1] firewall. Moreover, the backbone consists of a fiber ring (10Gbps), hence can manage vast amounts of data traffic.

6.2 Available tools

There are several open source tools which can be used for vulnerability testing on CMS systems. These tools have different objectives, e.g., find SQL injection vulnerabilities, searching for open ports, fingerprinting software versions, etc. However, they have one thing in common; they are useful for both developers and adversaries.

6.2.1 Visual fingerprinting

From an attacker's perspective, it is valuable to find out what system is used on a website. There are various ways to determine whether a website is running Joomla!, or not. The easiest way is to append `?tp=1` to the site's URL as seen below, which triggers the modules debugging mode. This is a clear indication of that the website is running Joomla!.

```
http://example.net/index.html?tp=1
```

Another way of detecting a Joomla! site, is to use the query parameter; `tmpl`. By using the offline template (i.e., the template used when a Joomla! site is down), one can force a Joomla! website to appear offline.

```
http://example.net/index.html?tmpl=offline
```

It is possible, however, to prevent adversaries from getting this information. By adding a set of rules to the `.htaccess`[2] file, these attempts will end up in a

[1] http://www.fortinet.com/solutions/firewall.html

[2] The purpose of .htaccess files is to provide a means to configure Apache for users who cannot modify the main configuration file [52].

404-page[3] [53], as seen in the code-snippet below.

```
RewriteCond %QUERY_STRING (&|%3F)1,1 tp=[OR]
RewriteCond %QUERY_STRING (&|%3F)1,1 template=[OR]
RewriteCond %QUERY_STRING (&|%3F)1,1 tmpl=[NC]
RewriteRule ^(.*)$ - [R=404,L]
```

6.2.2 BlindElephant Web Application Fingerprinter

The BlindElephant Web Application Fingerprinter [8] tries to discover the version of a web application, by looking at the static files of that application. These static files are often located at the same places, thus it is possible to determine the web application version by comparing them to precomputed hashes of every static files from all releases. Since comparison of hashes are fast operations, the BlindElephant use low bandwidth and can be used for automated operations.

When installing Joomla! on a web host, static files and folders are automatically deployed. These files include language files, images, CSS and scripts. Each version of Joomla! has a slightly different setup of static resources, than the others. As a result, using the BlindElephant tool could give a relatively accurate guess of what version of Joomla! is running. In figure 6.1 on page 72, the fingerprinting process of static files in Joomla! is illustrated. One way to prevent similar fingerprinters, is to remove or rename the static resources (e.g., stock images). In appendix C on page 93, the BlindElephant tool is used on the test website running in the cloud. The tool was able to recognize the correct version of Joomla!, simply by searching for static files. An adversary can use this information, to get a picture of which vulnerabilities that apply to the system. Hence, the adversary could easily find possible exploits.

[3] A 404-page (not found error message) is a HTTP standard response when a server was unable to locate the requested page.

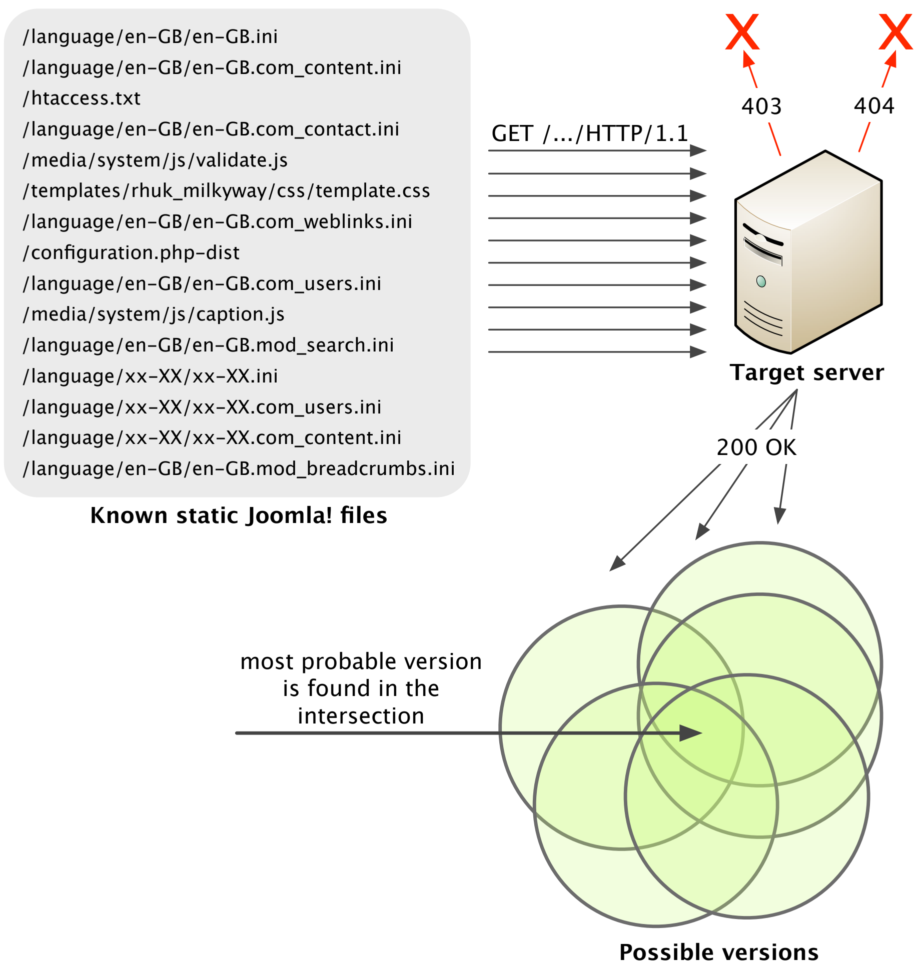

/language/en-GB/en-GB.ini
/language/en-GB/en-GB.com_content.ini
/htaccess.txt
/language/en-GB/en-GB.com_contact.ini
/media/system/js/validate.js
/templates/rhuk_milkyway/css/template.css
/language/en-GB/en-GB.com_weblinks.ini
/configuration.php-dist
/language/en-GB/en-GB.com_users.ini
/media/system/js/caption.js
/language/en-GB/en-GB.mod_search.ini
/language/xx-XX/xx-XX.ini
/language/xx-XX/xx-XX.com_users.ini
/language/xx-XX/xx-XX.com_content.ini
/language/en-GB/en-GB.mod_breadcrumbs.ini
Known static Joomla! files
GET /.../HTTP/1.1
X
X
403
404
Target server
200 OK
most probable version is found in the intersection
Possible versions

Figure 6.1: Static file fingerprinting on Joomla! [8] (modified figure)

6.2.3 Nmap – Network mapping tool

Nmap [54] is an open source tool for analyzing network security. It was originally designed to scan large networks, however, is also suitable for a single host. This tool use raw IP-packets to find useful information about network hosts and their services (name and version), operating systems (OS versions), firewalls, among others. From an attacker's point of view, this tool could discover open ports with potentially vulnerable services running. On the other hand, system administrators can use Nmap to search for unauthorized servers on their network. Figure 6.2 and figure 6.3 on page 74 illustrate how Nmap uses SYN packets to retrieve information about open or closed ports.

As seen in appendix D on page 95, Nmap was used to find valuable information about the host' system, on the test website. With just two simple scans, the information in table 6.1 was retrieved. Furthermore, the operating system was detected to be Linux 2.6.9–2.6.27. An adversary could do a quick search on "exploits lighttpd 1.4.28", and probably get interesting results. This is a very simple example, however, it shows the significance of hiding sensitive information. For instance, many 404-errors reveals information about the server's software. Nmap is an extremely powerful tool if used correctly, and is quite easy to get started with.

Port	**State**	**Service**	**Version**
21/tcp	open	ftp	ProFTPD
22/tcp	open	ssh	OpenSSH 5.5 (protocol 2.0)
70/tcp	open	http	lighttpd 1.4.28
80/tcp	open	http	Apache httpd 2.2.3 (CloudLinux)

Table 6.1: System information retrieved from the Joomla! setup, using Nmap

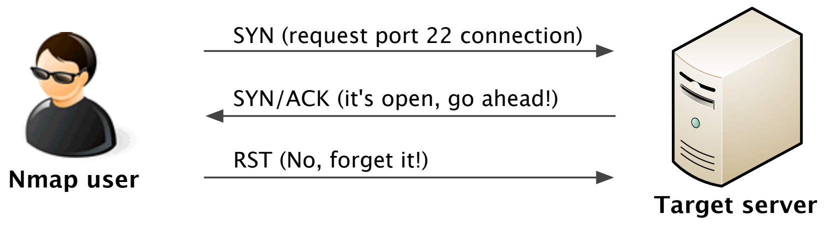

SYN (request port 22 connection)
SYN/ACK (it's open, go ahead!)
RST (No, forget it!)
Nmap user
Target server

Figure 6.2: SYN-scan of open port 22

6.2.4 OWASP Joomla! vulnearbility scanner

The third tool used to analyze the Joomla! test site is the OWASP Joomla! vulnerability scanner. The features of this open source project include:

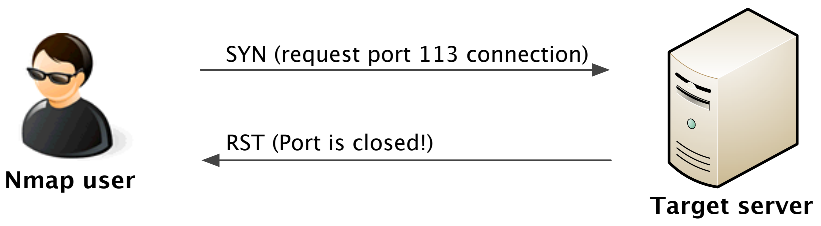


Figure 6.3: SYN-scan of closed port 113

- Exact version probing
- Common Joomla! firewall detection
- Searching through known vulnerabilities of Joomla and its components

This scanner can find known vulnerabilities such as XSS, SQL injections and file inclusions to name a few. Moreover, this tool is updated regularly, since new vulnerabilities are discovered continuously. Developers and system administrators can utilize this vulnerability scanner to find security flaws. This pen-test allows developers and system administrators to get ahead of the attackers. However, this scanner does not encompass every facet of vulnerabilities. For example, the vulnerability database used by the tool, does not include vulnerabilities whereas the exploitation method is unknown. Hence, one should not blindly trust that a system is secure if no vulnerabilities are found with this scanner.

The OWASP Joomla! vulnerability scanner found six vulnerabilities on the test website, as seen in appendix E on page 99. Since the Joomla! website was pre-configured with default settings and no extensions installed, few components were exploitable. The first vulnerability detected, was that the file `htaccess.txt` was not renamed. Since many hosts provide their own `.htaccess` file, Joomla! has given another file-ending to it. Then it is possible for the system administrator to chose which one to use. However, the `htaccess.txt` file is readable, thus adversaries can gather information about the system. The second vulnerability on the test system was an unprotected administrator directory, which is discussed in section 5.2.2 on page 61. Furthermore, the scanner found a CSRF vulnerability in the same directory. An adversary can trick a user to click on a malicious link, while logged into the back-end of a Joomla! site. Then the adversary can forge requests to the system, pretending to be the correct user. The vulnerability scanner also discovered a security issue regarding a core plugin, and two vulnerable components.

Chapter summary

The intention of these tests were to give a brief overview of how an attacker can get information from a website running Joomla!. Even though a pre-configured Joomla! setup from a CSP was used, several potential vulnerabilities were discovered. With the help of the BlindElephant tool, the exact version of Joomla! was determined. Furthermore, Nmap provided detailed information about the system, including software version numbers and open ports. Then the Joomla! vulnerability scanner analyzed the test website, and found six exploitable vulnerabilities. By utilizing freely available open source tools, sensitive information about a system can easily be collected. These tools should be used as a preventive mechanism, thus staying one step in front of the attackers. Another conclusion to be drawn from the test is to hide system information, e.g., with proper error handling mechanisms.

7 Conclusion

As seen throughout this report, cloud computing could be the next evolution in the history of computing. Although there still are several questions which need to be answered regarding cloud security, cloud computing has made a serious impact in the world of technology. Organizations have found many interesting ways to make use of the possibilities of the cloud. Popular cloud services, e.g., Dropbox, SoundCloud[1] and GoogleDocs, have made people open their eyes to this new technology. Furthermore, companies invest in enormous data-centers to provide cloud services. Cloud computing is becoming a utility, where an Internet connection is the only necessity to access a world of services and resources.

Both customers and organizations can benefit from this technology. The reduction in maintenance and operational costs are significant. These tasks are the responsibility of the CSPs, hence transparent to the customers. As a result, organizations are not depended of their own dedicated operational personnel. Furthermore, the elasticity of the cloud allows for an organization to easily expand their service. Upgrading a system has previously meant buying new expensive network equipment, however, the cloud scales on demand.

Although the cloud seems like a God-given technology, there are some concerns regarding security. Cloud security has received a vast amount of attention, and not without reason. In addition to the traditional web application vulnerabilities (e.g. XSS, CSRF, SQL injections), new threats emerges. The multi-tenancy and

[1] Amazon's SoundCloud is a platform which allows users to easily upload and share their music.

shared environment introduce cloud-specific security issues. Moreover, dark clouds could possibly cause serious harm to any service. The elasticity of the cloud could be exploited by malicious users to perform tremendous DoS attacks. If the power of several clouds could be combined, in a similar way as bot-nets, DDoS attacks would seem unstoppable.

CMSs are attractive targets for adversaries, due to their many potential vulnerabilities. With some knowledge of information security, a majority of these threats could be avoided. However, a vast amount of CMS users have limited technical background, thus their websites are easily exploited. CMSs allow customers to build professional websites, without the need of writing a single line of code. They can simply add the contents they need, and customize the design with pre-built templates. Moreover, they can enhance the functionality of their site by installing components. The CMS analyzed in this report (Joomla!), has a huge collection of extensions. These extensions introduce risks to the users, since they are made by third-party developers. Joomla! does not validate the security of these extensions, hence they could obtain malicious code or simply be poorly coded. If an extension is vulnerable, adversaries could exploit them to compromise a web site, or access sensitive information. Chapter 6.1 on page 69, illustrates how facile it is to extract valuable information about a system, only by using open source tools. Moreover, adversaries are able to automate attacks against websites with vulnerable components, which shows how important it is to have them updated. Every facet of a system need to be up to date. Cloud computing makes this task a bit simpler, since the CSP transparently updates software used on their cloud. Furthermore, some CMSs (e.g., Joomla!) offer automatic updates of extensions within the back-end of the system.

It is difficult to predict what the future would bring to cloud computing. However, it is likely that more and more services move to the cloud. Amazon, Apple, Google and Microsoft are rapidly developing cloud services, hence they would have a significance on people's everyday life. CMSs have started their migration to the cloud, and it is probable that their customers would do the same. However, CMSs must perform some cleaver moves to be considered secure. Since most of the vulnerabilities found in CMSs are the customers's responsibility, features which makes it more effortless to keep up to date, are of importance. Additionally, CMSs could provide more intuitive APIs to the developers, i.e. make it more simple to add preventive functions against common attacks.

Bibliography

[1] Centre for the Protection of National Infrastructure (CPNI). Information Security Briefing - Cloud Computing, March 2010.

[2] Tim Mather, Subra Kumaraswamy and Shahed Latif. *Cloud Security and Privacy - An Enterprise Perspective on Risks and Compliance*, volume First Edition. O'Reilly, September 2009.

[3] IBM X-Force. 2010 Trend and Risk Report. `http://www-935.ibm.com/services/us/iss/xforce/trendreports/`, March 2011. *Accessed 17.03.2011*.

[4] Twitter.com. Ongoing denial-of-service attack. `http://status.twitter.com/post/157191978/ongoing-denial-of-service-attack`, August, 6th 2009. *Accessed 28.04.2011*.

[5] Cong Wang and Qian Wang and Kui Ren and Wenjing Lou. Ensuring data storage security in Cloud Computing. In *Quality of Service, 2009. IWQoS. 17th International Workshop on*, pages 1 –9, July 2009.

[6] Meike, M. and Sametinger, J. and Wiesauer, A. Security in Open Source Web Content Management Systems. *Security Privacy, IEEE*, 7(4):44 –51, July-August 2009.

[7] Joomla.org. What is Joomla? `http://www.joomla.org/about-joomla.html`. *Accessed 24.01.2011*.

[8] Patric Thomas. BlindElephant Web Application Fingerprinter. `http://blindelephant.sourceforge.net/`. *Accessed 04.04.2011*.

[9] Nicholas G Carr. The Big Switch: Rewiring the World, from Edison to Google. `http://www.nicholasgcarr.com/bigswitch/interview.shtml`, 2008. *Accessed 01.03.2011*.

[10] The Cloud Security Alliance (CSA).
https://cloudsecurityalliance.org/. *Accessed 07.02.2011*.

[11] CloudTweaks - Sourya Biswas. A History of Cloud Computing.
http://www.cloudtweaks.com/2011/02/a-history-of-cloud-computing/, February 2011. *Accessed 01.03.2011*.

[12] The New York Times - Steve Lohr. Google and I.B.M. Join in 'Cloud Computing' Research.
http://www.nytimes.com/2007/10/08/technology/08cloud.html?_r=3&ex=1349496000&en=92627f0f65ea0d75&ei=5090&partner=rssuserland&emc=rss&oref=slogin, October 2007. *Accessed 03.03.2011*.

[13] IBM Corporation, Dan Orlando. Cloud computing service models, Part3: Software as a service.
http://www.ibm.com/developerworks/cloud/library/cl-cloudservicemodels/index.html, January 2011. *Accessed 07.02.2011*.

[14] IBM Corporation, Dan Orlando. Cloud computing service models, Part2: Platform as a service.
http://www.ibm.com/developerworks/cloud/library/cl-cloudservicemodels/index.html, January 2011. *Accessed 07.02.2011*.

[15] IBM Corporation, Dan Orlando. Cloud computing service models, Part1: Infrastructure as a service.
http://www.ibm.com/developerworks/cloud/library/cl-cloudservicemodels/index.html, January 2011. *Accessed 07.02.2011*.

[16] SearchSOA.com. Definition of REST (representational state transfer).
http://searchsoa.techtarget.com/definition/REST. *Accessed 02.05.2011*.

[17] Colin Percival. AWS signature version 1 is insecure, from the blog Daemonic Dispatches.
http://www.daemonology.net/blog/2008-12-18-AWS-signature-version-1-is-insecure.html, December 2008. *Accessed 23.03.2011*.

[18] SearchNetworking.com. Definition of Border Gateway Protocol (BGP).
http://searchtelecom.techtarget.com/definition/BGP. *Accessed 06.04.2011*.

[19] SearchNetworking.com. Definition of Autonomeous System (AS).

http://searchnetworking.techtarget.com/definition/autonomous-system. *Accessed 06.04.2011*.

[20] Brad Stone - The New York Times. Pakistan Cuts Access to YouTube Worldwide.
http://www.nytimes.com/2008/02/26/technology/26tube.html, February 2008. *Accessed 23.03.2011*.

[21] SearchNetworking.com. Definition of Domain Name System (DNS).
http://searchtelecom.techtarget.com/definition/domain-name-system. *Accessed 06.04.2011*.

[22] NYTIMES.com - Scott Shane and Andrew W. Lehren. Leaked Cables Offer Raw Look at U.S. Diplomacy.
http://www.nytimes.com/2010/11/29/world/29cables.html?hp, November 2009. *Accessed 01.03.2011*.

[23] FoxNews.com. Amazon Pulls Plug on WikiLeaks.
http://www.foxnews.com/scitech/2010/12/01/amazon-severs-ties-wikileaks/, December 2010. *Accessed 28.04.2011*.

[24] Whatis?com. What is Virtual Machine Escape.
http://whatis.techtarget.com/definition/virtual-machine-escape.html. *Accessed 07.04.2011*.

[25] SearchNetworking.com. Definition of Secure Shell (SSH).
http://searchtelecom.techtarget.com/definition/Secure-Shell. *Accessed 06.04.2011*.

[26] Claudio Cristone - SecureNetwork. You are doing it wrong, Failures in Virutalization Systems, BlackHat Europe.
https://www.blackhat.com/html/bh-eu-11/bh-eu-11-archives.html, March 2011. *Accessed 24.03.2011*.

[27] The Open Web Application Security Project (OWASP). About OWASP.
https://www.owasp.org/index.php/About_The_Open_Web_Application_Security_Project. *Accessed 17.03.2011*.

[28] Ade Barkah (peekay.org). Security issues with Google Docs.
http://peekay.org/2009/03/26/security-issues-with-google-docs/, March, 26th 2009. *Accessed 28.04.2011*.

[29] Amazon Web Services (AWS). Overview of Security Processes. http://aws.amazon.com/articles/1697?_encoding=UTF8&jiveRedirect=1, December, 5th 2008. *Accessed 28.04.2011*.

[30] Dropbox. How secure is Dropbox? https://www.dropbox.com/help/27, April, 23rd 2011. *Accessed 28.04.2011*.

[31] Ryan Single - Wired.com. Dropbox Lied to Users About Data Security, Complaint to FTC Alleges. http://www.wired.com/threatlevel/2011/05/dropbox-ftc/, May, 13th 2011. *Accessed 13.05.2011*.

[32] Sravan Kumar, R. and Saxena, A. Data integrity proofs in cloud storage. In *Communication Systems and Networks (COMSNETS), 2011 Third International Conference on*, pages 1 –4, January 2011.

[33] Mike Stevens - Hosting.com and IT Business Edge. How Real Are Cloud Security Concerns?, Separating Fact from Fiction for Infrastructure-as-a-Service (Iaas) Cloud Computing, 2010.

[34] SearchNetworking.com. Definition of Rootkit. http://searchmidmarketsecurity.techtarget.com/definition/rootkit. *Accessed 06.04.2011*.

[35] Amazon Web Services. Using Shared AMIs. http://docs.amazonwebservices.com/AWSEC2/latest/UserGuide/index.html?AESDG-chapter-usingsharedamis.html#usingsharedamis-security. *Accessed 25.03.2011*.

[36] Laura Quinn and Heather Gardner-Madras from Idealware. 2010 Comparing Open Source Content Management Systems: WordPress, Joomla, Drupal and Plone. http://www.idealware.org/reports/2010-os-cms, December 2010. *Accessed 22.04.2011*.

[37] Google Insights for Search. http://www.google.com/insights/search/#. *Accessed 23.04.2011*.

[38] Lisa Wehr. Which CMS is Right for Your Business? http://mashable.com/2011/04/05/best-cms-for-business/, April, 5th 2011. *Accessed 22.04.2011*.

[39] WEBOPEDIA. Definition of Script Kiddies.
`http://www.webopedia.com/TERM/S/script_kiddie.html`. *Accessed 02.05.2011*.

[40] THE OPEN WEB APPLICATION SECURITY PROJECT (OWASP). OWASP Top 10 - 2010, The Ten Most Critical Web Application Security Risks.
`https://www.owasp.org/index.php/OWASP_Top_Ten_Project`, April 2010. *Accessed 17.03.2011*.

[41] CLOUDACCESS.NET. Joomla! Features Tour.
`http://www.cloudaccess.net/joomla-features-tour.html`. *Accessed 31.01.2011*.

[42] CAPTCHA.NET. CAPTCHA: Telling Humans and Computers Apart Automatically.
`http://www.captcha.net/`. *Accessed 29.04.2011*.

[43] JOOMLA.ORG. Joomla! 1.6 Has Arrived!
`http://www.joomla.org/announcements/general-news/5348-joomlar-16-has-arrived.html`, January, 10th 2011. *Accessed 24.01.2011*.

[44] W3C. XHTML$^{\mathrm{TM}}$ 1.0 The Extensible HyperText Markup Language (Second Edition).
`http://www.w3.org/TR/xhtml1/`, August 2002. *Accessed 29.04.2011*.

[45] JOOMLA.ORG. ACL Tutorial for Joomla 1.6.
`http://docs.joomla.org/ACL_Tutorial_for_Joomla_1.6`, March 2011. *Accessed 22.02.2011*.

[46] AN AUTONOMOUS ZONE. XSS Cheat Sheet.
`http://anautonomouszone.com/blog/xss-cheat-sheet`. *Accessed 05.04.2011*.

[47] PHP.NET. mysql_real_escape_string().
`http://php.net/manual/en/function.mysql-real-escape-string.php`. *Accessed 29.04.2011*.

[48] WANG YU AND CHEN JIANHUA AND HE DEBIAO. A New Collision Attack on MD5. In *Networks Security, Wireless Communications and Trusted Computing, 2009. NSWCTC '09. International Conference on*, volume 2, pages 767 –770, April 2009.

[49] JAVVIN - NETWORK MANAGEMENT AND SECURITY. Birthday Attack.

`http://www.javvin.com/networksecurity/BirthdayAttack.html`. *Accessed 26.04.2011*.

[50] Joomla.org. Joomla! Extensions Directory (JED).
`http://extensions.joomla.org/`. *Accessed 22.02.2011*.

[51] Cloudaccess.net. CloudAccess.net Network Power Service Level Agreement (SLA).
`http://www.cloudaccess.net/network-sla.html`. *Accessed 28.04.2011*.

[52] The Apache Software Foundation. Htaccess.
`http://wiki.apache.org/httpd/Htaccess`. *Accessed 04.04.2011*.

[53] Joomla! Community Magazine - Nicholas K. Dionysopoulos. Only a Ninja can kill another Ninja.
`http://magazine.joomla.org/component/k2/item/214`, October 2010. *Accessed 04.04.2011*.

[54] Gordon Lyon - Insecure.org. Nmap documentation.
`http://nmap.org/docs.html`. *Accessed 04.04.2011*.

[55] Ron Rivest. MIT Laboratory for Computer Science and RSA Data Security Inc., The MD5 Message-Digest Algorithm, Network Working Group, Request for Comments: 1321.
`http://tools.ietf.org/pdf/rfc1321.pdf`, April 1992. *Accessed 26.04.2011*.

A
OWASP Top 10 Application Security Risks - 2010

The ten most critical web application security risks in descending order, according to OWASP [40]:

- **A1 – Injection:**
  Injection flaws, such as SQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing unauthorized data.

- **A2 – Cross-Site Scripting (XSS):**
  XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation and escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

- **A3 – Broken Authentication and Session Management:**
  Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, session tokens, or exploit other implementation flaws to assume other users' identities.

- **A4 – Insecure Direct Object References:**
  A direct object reference occurs when a developer exposes a reference to an

internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.

- **A5 – Cross-Site Request Forgery (CSRF):**
  A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.

- **A6 – Security Misconfiguration:**
  Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. All these settings should be defined, implemented, and maintained as many are not shipped with secure defaults. This includes keeping all software up to date, including all code libraries used by the application.

- **A7 – Insecure Cryptographic Storage:**
  Many web applications do not properly protect sensitive data, such as credit cards, SSNs, and authentication credentials, with appropriate encryption or hashing. Attackers may steal or modify such weakly protected data to conduct identity theft, credit card fraud, or other crimes.

- **A8 - Failure to Restrict URL Access:**
  Many web applications check URL access rights before rendering protected links and buttons. However, applications need to perform similar access control checks each time these pages are accessed, or attackers will be able to forge URLs to access these hidden pages anyway.

- **A9 - Insufficient Transport Layer Protection:**
  Applications frequently fail to authenticate, encrypt, and protect the confidentiality and integrity of sensitive network traffic. When they do, they sometimes support weak algorithms, use expired or invalid certificates, or do not use them correctly.

- **A10 – Unvalidated Redirects and Forwards**
  Web applications frequently redirect and forward users to other pages and

websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.

B

Annual trends for web application vulnerability types

The data used to construct the graph in figure B.1 on page 92, is extracted from the IBM X-Force's 2010 trend and risk report [3].

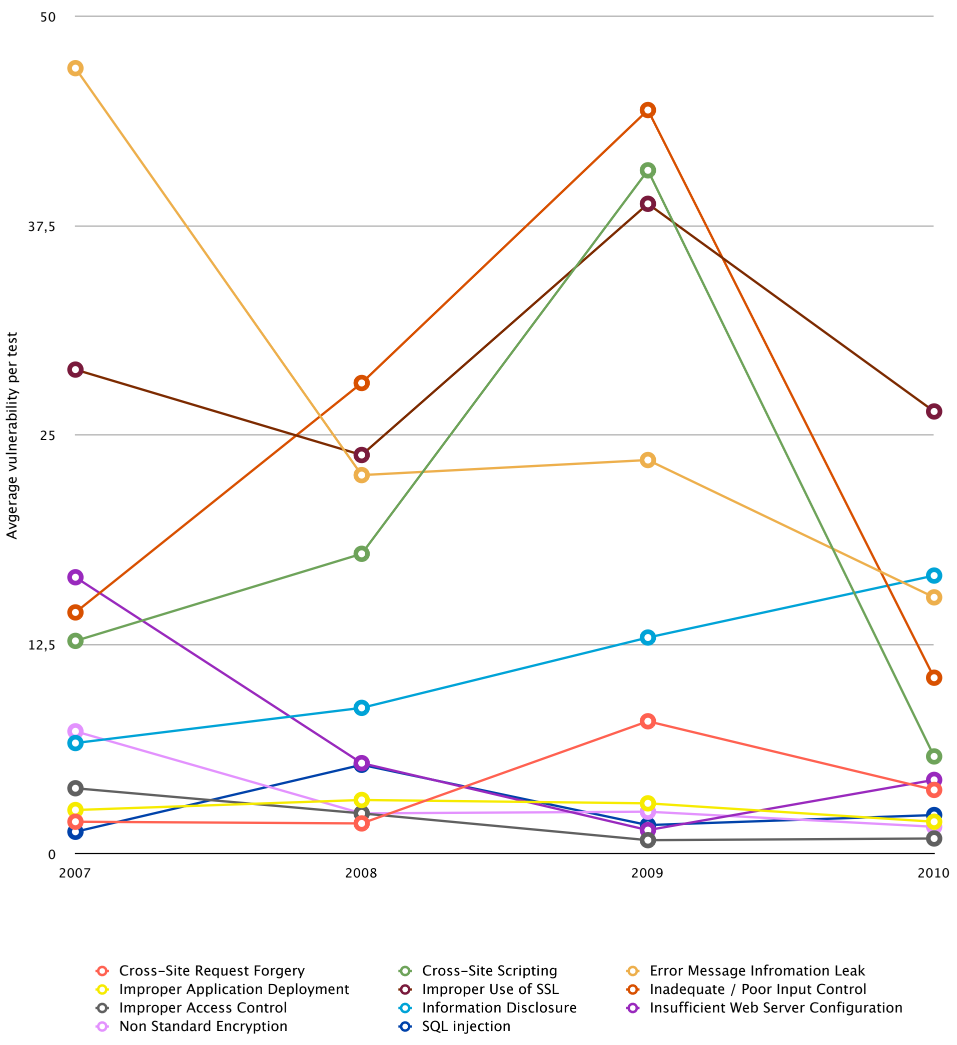


Figure B.1: Annual trends for web application vulnerability types

C
BlindElephant fingerprinting

```
user$ python BlindElephant.py example.net guess
Probing...
Possible apps:
joomla

user$ python BlindElephant.py example.net joomla
Loaded .~/blindelephant/dbs/joomla.pkl
with 39 versions, 3789 differentiating paths, and 140 version groups.
Starting BlindElephant fingerprint for version of joomla at
http://example.net

Hit http://example.net/language/en-GB/en-GB.ini
Possible versions based on result: 1.5.18

Hit http://example.net/language/en-GB/en-GB.com_content.ini
Possible versions based on result: 1.5.17, 1.5.18

Hit http://example.net/htaccess.txt
Possible versions based on result: 1.5.15

Hit http://example.net/language/en-GB/en-GB.com_contact.ini
Possible versions based on result: 1.5.17, 1.5.18

Hit http://example.net/media/system/js/validate.js
Possible versions based on result: 1.5.17, 1.5.18

Hit http://example.net/templates/rhuk_milkyway/css/template.css
Possible versions based on result: 1.5.17, 1.5.18

Hit http://example.net/language/en-GB/en-GB.com_weblinks.ini
```

```
Possible versions based on result: 1.5.17, 1.5.18

Hit http://example.net/configuration.php-dist
File produced no match. Error: Error code: 404 (Not Found)

Hit http://example.net/language/en-GB/en-GB.com_users.ini
File produced no match. Error: Error code: 404 (Not Found)

Hit http://example.net/media/system/js/caption.js
Possible versions based on result: 1.5.17, 1.5.18

Hit http://example.net/language/en-GB/en-GB.mod_search.ini
Possible versions based on result: 1.5.17, 1.5.18

Hit http://example.net/language/xx-XX/xx-XX.ini
File produced no match. Error: Error code: 404 (Not Found)

Hit http://example.net/language/xx-XX/xx-XX.com_users.ini
File produced no match. Error: Error code: 404 (Not Found)

Hit http://example.net/language/xx-XX/xx-XX.com_content.ini
File produced no match. Error: Error code: 404 (Not Found)

Hit http://example.net/language/en-GB/en-GB.mod_breadcrumbs.ini
Possible versions based on result: 1.5.17, 1.5.18


Fingerprinting resulted in:
1.5.18


Best Guess: 1.5.18
```

D

Nmap and Joomla!

D.1 Version trace

```
Starting Nmap 5.51 ( http://nmap.org ) at 2011-03-07 09:30 CET
PORTS: Using top 1000 ports found open (TCP:1000, UDP:0, SCTP:0)
--------------- Timing report ---------------
  hostgroups: min 1, max 100000
  rtt-timeouts: init 1000, min 100, max 10000
  max-scan-delay: TCP 1000, UDP 1000, SCTP 1000
  parallelism: min 0, max 0
  max-retries: 10, host-timeout: 0
  min-rate: 0, max-rate: 0
---------------------------------------------
NSE: Loaded 8 scripts for scanning.
Overall sending rates: 1.23 packets / s.
mass_rdns: Using DNS server 129.241.208.40
mass_rdns: Using DNS server 129.241.206.252
mass_rdns: Using DNS server 129.241.0.200
mass_rdns: Using DNS server 129.241.0.201
mass_rdns: 0.47s 0/1 [#: 4, OK: 0, NX: 0, DR: 0, SF: 0, TR: 1]
DNS resolution of 1 IPs took 0.47s. Mode: Async [#: 4, OK: 1, NX: 0, DR: 0,
SF: 0, TR: 1, CN: 0]
Increased max_successful_tryno for xxx.xxx.xxx.xxx to 1 (packet drop)
Overall sending rates: 28.44 packets / s.

NSOCK (110.2190s) TCP connection requested to xxx.xxx.xxx.xxx:21 (IOD #1)
EID 8
NSOCK (110.2190s) TCP connection requested to xxx.xxx.xxx.xxx:22 (IOD #2)
EID 16
```

```
NSOCK (110.2200s) TCP connection requested to xxx.xxx.xxx.xxx:70 (IOD #3)
EID 24
NSOCK (110.2200s) TCP connection requested to xxx.xxx.xxx.xxx:80 (IOD #4)
EID 32
NSOCK (110.3640s) Callback: CONNECT SUCCESS for EID 8 [xxx.xxx.xxx.xxx:21]
Service scan sending probe NULL to xxx.xxx.xxx.xxx:21 (tcp)
NSOCK (110.3640s) Read request from IOD #1 [xxx.xxx.xxx.xxx:21] (timeout:
6000ms) EID 42
NSOCK (110.3650s) Callback: CONNECT SUCCESS for EID 16 [xxx.xxx.xxx.xxx:22]
Service scan sending probe NULL to xxx.xxx.xxx.xxx:22 (tcp)
NSOCK (110.3650s) Read request from IOD #2 [xxx.xxx.xxx.xxx:22] (timeout:
6000ms) EID 50
NSOCK (110.3650s) Callback: CONNECT SUCCESS for EID 24 [xxx.xxx.xxx.xxx:70]
Service scan sending probe NULL to xxx.xxx.xxx.xxx:70 (tcp)
NSOCK (110.3650s) Read request from IOD #3 [xxx.xxx.xxx.xxx:70] (timeout:
6000ms) EID 58
NSOCK (110.3650s) Callback: CONNECT SUCCESS for EID 32 [xxx.xxx.xxx.xxx:80]
Service scan sending probe NULL to xxx.xxx.xxx.xxx:80 (tcp)
NSOCK (110.3650s) Read request from IOD #4 [xxx.xxx.xxx.xxx:80] (timeout:
6000ms) EID 66
NSOCK (110.5150s) Callback: READ SUCCESS for EID 50 [xxx.xxx.xxx.xxx:22]
(21 bytes): SSH-2.0-OpenSSH_5.5..
Service scan match (Probe NULL matched with NULL): xxx.xxx.xxx.xxx:22 is ssh.
Version: |OpenSSH|5.5|protocol 2.0|
NSOCK (110.5210s) Callback: READ SUCCESS for EID 42 [xxx.xxx.xxx.xxx:21]
(23 bytes): 220 FTP Server ready...
Service scan match (Probe NULL matched with NULL): xxx.xxx.xxx.xxx:21 is ftp.
Version: |ProFTPD|||
NSOCK (116.3680s) Callback: READ TIMEOUT for EID 58 [xxx.xxx.xxx.xxx:70]
Service scan sending probe GetRequest to xxx.xxx.xxx.xxx:70 (tcp)
NSOCK (116.3680s) Write request for 18 bytes to IOD #3 EID 75
[xxx.xxx.xxx.xxx:70]: GET / HTTP/1.0....
NSOCK (116.3680s) Read request from IOD #3 [xxx.xxx.xxx.xxx:70] (timeout:
5000ms) EID 82
NSOCK (116.3680s) Callback: READ TIMEOUT for EID 66 [xxx.xxx.xxx.xxx:80]
Service scan sending probe GetRequest to xxx.xxx.xxx.xxx:80 (tcp)
NSOCK (116.3680s) Write request for 18 bytes to IOD #4 EID 91
[xxx.xxx.xxx.xxx:80]: GET / HTTP/1.0....
NSOCK (116.3680s) Read request from IOD #4 [xxx.xxx.xxx.xxx:80] (timeout:
5000ms) EID 98
NSOCK (116.3680s) Callback: WRITE SUCCESS for EID 75 [xxx.xxx.xxx.xxx:70]
NSOCK (116.3680s) Callback: WRITE SUCCESS for EID 91 [xxx.xxx.xxx.xxx:80]
NSOCK (116.5130s) Callback: READ SUCCESS for EID 82 [xxx.xxx.xxx.xxx:70]
(1708 bytes)
Service scan match (Probe GetRequest matched with GetRequest):
```

```
xxx.xxx.xxx.xxx:70 is http.  Version: |lighttpd|1.4.28||
NSOCK (117.3490s) Callback: READ SUCCESS for EID 98 [xxx.xxx.xxx.xxx:80]
(4344 bytes)
Service scan match (Probe GetRequest matched with GetRequest):
xxx.xxx.xxx.xxx:80 is http.  Version: |Apache httpd|2.2.3|(CloudLinux)|

Starting RPC scan against example.net (xxx.xxx.xxx.xxx)
NSE: Starting runlevel 1 (of 1) scan.
Nmap scan report for example.net (xxx.xxx.xxx.xxx)
Host is up (0.14s latency).
rDNS record for xxx.xxx.xxx.xxx:
xxx-xxx-xxx-xxx.static.trcy.mi.charter.com
Scanned at 2011-03-07 09:30:23 CET for 117s
Not shown: 982 filtered ports
PORT      STATE  SERVICE     VERSION
20/tcp    closed ftp-data
21/tcp    open   ftp         ProFTPD
22/tcp    open   ssh         OpenSSH 5.5 (protocol 2.0)
25/tcp    closed smtp
70/tcp    open   http        lighttpd 1.4.28
80/tcp    open   http        Apache httpd 2.2.3 ((CloudLinux))
12345/tcp closed netbus
60020/tcp closed unknown
60443/tcp closed unknown
61532/tcp closed unknown
61900/tcp closed unknown
62078/tcp closed iphone-sync
63331/tcp closed unknown
64623/tcp closed unknown
64680/tcp closed unknown
65000/tcp closed unknown
65129/tcp closed unknown
65389/tcp closed unknown
Service Info: OS: Unix
Final times for host: srtt: 144419 rttvar: 405  to: 146039

Nmap done: 1 IP address (1 host up) scanned in 117.45 seconds
```

D.2 OS detection

```
Starting Nmap 5.51 ( http://nmap.org ) at 2011-03-07 09:42 CET
Nmap scan report for example.net (xxx.xxx.xxx.xxx)
Host is up (0.14s latency).
```

```
rDNS record for xxx.xxx.xxx.xxx:
xxx-xxx-xxx-xxx.static.trcy.mi.charter.com
Not shown: 984 filtered ports
PORT      STATE  SERVICE
20/tcp    closed ftp-data
21/tcp    open   ftp
22/tcp    open   ssh
70/tcp    open   gopher
80/tcp    open   http
60020/tcp closed unknown
60443/tcp closed unknown
61532/tcp closed unknown
61900/tcp closed unknown
62078/tcp closed iphone-sync
63331/tcp closed unknown
64623/tcp closed unknown
64680/tcp closed unknown
65000/tcp closed unknown
65129/tcp closed unknown
65389/tcp closed unknown
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.9 - 2.6.27

OS detection performed. Please report any incorrect results at
http://nmap.org/submit/ .

Nmap done: 1 IP address (1 host up) scanned in 12.82 seconds
```

E

OWASP Joomla vulnerability scanner

```
=================================================================
OWASP Joomla! Vulnerability Scanner v0.0.3-b
(c) Aung Khant, aungkhant]at[yehg.net
YGN Ethical Hacker Group, Myanmar, http://yehg.net/lab

=================================================================

Vulnerability Entries: 466
Last update: August 18, 2009

Use "update" option to update the database
Use "check" option to check the scanner update
Use "download" option to download the scanner latest version package
Use svn co to update the scanner
svn co https://joomscan.svn.sourceforge.net/svnroot/joomscan joomscan

Target: http://example.net

Server: Apache/2.2.3 (CloudLinux)
X-Powered-By: PHP/5.2.10


## Checking if the target has deployed an Anti-Scanner measure
[!] Scanning Passed ..... OK


## Detecting Joomla! based Firewall ...
```

```
[!] No known firewall detected!


## Fingerprinting in progress ...
~Generic version family ....... [1.5.x]
~1.5.x en-GB.ini revealed [1.5.12 - 1.5.14]
* Deduced version range is : [1.5.12 - 1.5.14]
## Fingerprinting done.


Vulnerabilities Discovered
==========================

# 1
Info -> Generic: htaccess.txt has not been renamed.
Versions Affected: Any
Check: /htaccess.txt
Exploit: Generic defenses implemented in .htaccess are not available, so
exploiting is more likely to succeed.
Vulnerable? Yes

# 2
Info -> Generic: Unprotected Administrator directory
Versions Affected: Any
Check: /administrator/
Exploit: The default /administrator directory is detected. Attackers can
bruteforce administrator accounts. Read: http://yehg.net/lab/pr0js/
view.php/MULTIPLE%20TRICKY%20WAYS%20TO%20PROTECT.pdf
Vulnerable? Yes

# 3
Info -> Core: Multiple XSS/CSRF Vulnerability
Versions Affected: 1.5.9 <=
Check: /?1.5.9-x
Exploit: A series of XSS and CSRF faults exist in the administrator
application.  Affected administrator components include com_admin,
com_media, com_search.  Both com_admin and com_search contain XSS
vulnerabilities, and com_media contains 2 CSRF vulnerabilities.
Vulnerable? No

# 4
Info -> Core: JSession SSL Session Disclosure Vulnerability
Versions effected: Joomla! 1.5.8 <=
Check: /?1.5.8-x
Exploit: When running a site under SSL (the entire site is forced to be
under ssl), Joomla! does not set the SSL flag on the cookie.  This can
allow someone monitoring the network to find the cookie related to the
session.
Vulnerable? No

# 5
Info -> Core: Frontend XSS Vulnerability
Versions effected: 1.5.10 <=
Check: /?1.5.10-x
Exploit: Some values were output from the database without being properly
escaped.  Most strings in question were sourced from the administrator
panel. Malicious normal admin can leverage it to gain access to super
admin.
Vulnerable? No

# 6
Info -> Core: Missing JEXEC Check - Path Disclosure Vulnerability
Versions effected: 1.5.11 <=
Check: /libraries/phpxmlrpc/xmlrpcs.php
Exploit: /libraries/phpxmlrpc/xmlrpcs.php
Vulnerable? No

# 7
Info -> Core: Missing JEXEC Check - Path Disclosure Vulnerability
Versions effected: 1.5.12 <=
Check: /libraries/joomla/utilities/compat/php50x.php
Exploit: /libraries/joomla/utilities/compat/php50x.php
Vulnerable? No

# 8
Info -> Core: Frontend XSS - HTTP_REFERER not properly filtered Vulnerability
Versions effected: 1.5.11 <=
Check: /?1.5.11-x-http_ref
Exploit: An attacker can inject JavaScript or DHTML code that will be
executed in the context of targeted user browser, allowing the attacker to
steal cookies. HTTP_REFERER variable is not properly parsed.
Vulnerable? No

# 9
Info -> Core: Frontend XSS - PHP_SELF not properly filtered Vulnerability
Versions effected: 1.5.11 <=
Check: /?1.5.11-x-php-s3lf
Exploit: An attacker can inject JavaScript code in a URL that will be
executed in the context of targeted user browser.
Vulnerable? No
```

```
# 10
Info -> Core: Authentication Bypass Vulnerability
Versions effected: Joomla! 1.5.3 <=
Check: /administrator/
Exploit: Backend accepts any password for custom Super Administrator when
LDAP enabled
Vulnerable? No

# 11
Info -> Core: Path Disclosure Vulnerability
Versions effected: Joomla! 1.5.3 <=
Check: /?1.5.3-path-disclose
Exploit: Crafted URL can disclose absolute path
Vulnerable? No

# 12
Info -> Core: User redirected Spamming Vulnerability
Versions effected: Joomla! 1.5.3 <=
Check: /?1.5.3-spam
Exploit: User redirect spam
Vulnerable? No

# 13
Info -> Core: joomla.php Remote File Inclusion Vulnerability
Versions effected: 1.0.0
Check: /includes/joomla.php
Exploit: /includes/joomla.php?includepath=
Vulnerable? No

# 14
Info -> Core: Admin Backend Cross Site Request Forgery Vulnerability
Versions effected: 1.0.13 <=
Check: /administrator/
Exploit: It requires an administrator to be logged in and to be tricked
into a specially crafted webpage.
Vulnerable? Yes

# 15
Info -> Core: Path Disclosure Vulnerability
Versions effected: Joomla! 1.5.12 <=
Check: /libraries/joomla/utilities/compat/php50x.php
Exploit: /libraries/joomla/utilities/compat/php50x.php
Vulnerable? No
```

```
# 16
Info -> CorePlugin: Xstandard Editor X_CMS_LIBRARY_PATH Local Directory
Traversal Vulnerability
Versions effected: Joomla! 1.5.8 <=
Check: /plugins/editors/xstandard/attachmentlibrary.php
Exploit: Submit new header X_CMS_LIBRARY_PATH with value ../ to  /plugins/
editors/xstandard/attachmentlibrary.php
Vulnerable? No

# 17
Info -> CoreTemplate: ja_purity XSS Vulnerability
Versions effected: 1.5.10 <=
Check: /templates/ja_purity/
Exploit: A XSS vulnerability exists in the JA_Purity template which ships
with Joomla! 1.5.
Vulnerable? No

# 18
Info -> CoreLibrary: phpmailer Remote Code Execution Vulnerability
Versions effected: Joomla! 1.5.0 Beta/Stable
Check: /libraries/phpmailer/phpmailer.php
Exploit: N/A
Vulnerable? No

# 19
Info -> CorePlugin: TinyMCE TinyBrowser addon multiple vulnerabilities
Versions effected: Joomla! 1.5.12
Check: /plugins/editors/tinymce/jscripts/tiny_mce/plugins/tinybrowser/
Exploit: While Joomla! team announced only File Upload vulnerability, in
fact there are many. See: http://www.milw0rm.com/exploits/9296
Vulnerable? Yes

# 20
Info -> CoreComponent: Joomla Remote Admin Password Change Vulnerability
Versions Affected: 1.5.5 <=
Check: /components/com_user/controller.php
Exploit: 1. Go to url : target.com/index.php?
option=com_user&view=reset&layout=confirm  2. Write into field "token" char
' and Click OK.  3. Write new password for admin  4. Go to url :
target.com/administrator/  5. Login admin with new password
Vulnerable? No

# 21
Info -> CoreComponent: com_content SQL Injection Vulnerability
Version Affected: Joomla! 1.0.0 <=
Check: /components/com_content/
Exploit: /index.php?
option=com_content&task=blogcategory&id=60&Itemid=99999+UNION+SELECT
+1,concat(0x1e,username,0x3a,password,0x1e,0x3a,usertype,0x1e),3,4,5+FROM
+jos_users+where+usertype=0x53757065722041646d696e6973747261746f72--
Vulnerable? No

# 22
Info -> CoreComponent: com_search Remote Code Execution Vulnerability
Version Affected: Joomla! 1.5.0 beta 2 <=
Check: /components/com_search/
Exploit: /index.php?option=com_search&Itemid=1&searchword=%22%3Becho%20md5
(911)%3B
Vulnerable? No

# 23
Info -> CoreComponent: com_admin File Inclusion Vulnerability
Versions Affected: N/A
Check: /administrator/components/com_admin/admin.admin.html.php
Exploit: /administrator/components/com_admin/admin.admin.html.php?
mosConfig_absolute_path=
Vulnerable? No

# 24
Info -> CoreComponent: MailTo SQL Injection Vulnerability
Versions effected: N/A
Check: /components/com_mailto/
Exploit: /index.php?option=com_mailto&tmpl=mailto&article=550513+and
+1=2+union+select+concat(username,char(58),password)+from+jos_users+where
+usertype=0x53757065722041646d696e6973747261746f72--&Itemid=1
Vulnerable? No

# 25
Info -> CoreComponent: com_content Blind SQL Injection Vulnerability
Versions effected: Joomla! 1.5.0 RC3
Check: /components/com_content/
Exploit: /index.php?option=com_content&view=%' +'a'='a&id=25&Itemid=28
Vulnerable? No

# 26
Info -> CoreComponent: com_content XSS Vulnerability
Version Affected: Joomla! 1.5.7 <=
Check: /components/com_content/
Exploit: The defaults on com_content article submission allow entry of
dangerous HTML tags (script, etc).  This only affects users with access
```

```
level Author or higher, and only if you have not set filtering options in
com_content configuration.
Vulnerable? No

# 27
Info -> CoreComponent: com_weblinks XSS Vulnerability
Version Affected: Joomla! 1.5.7 <=
Check: /components/com_weblinks/
Exploit: [Requires valid user account] com_weblinks allows raw HTML into
the title and description tags for weblink submissions (from both the
administrator and site submission forms).
Vulnerable? No

# 28
Info -> CoreComponent: com_mailto Email Spam Vulnerability
Version Affected: Joomla! 1.5.6 <=
Check: /components/com_mailto/
Exploit: The mailto component does not verify validity of the URL prior to
sending.
Vulnerable? No

# 29
Info -> CoreComponent: com_content view=archive SQL Injection Vulnerability
Versions effected: Joomla! 1.5.0 Beta1/Beta2/RC1
Check: /components/com_content/
Exploit: Unfiltered POST vars - filter, month, year  to /index.php?
option=com_content&view=archive
Vulnerable? No

# 30
Info -> CoreComponent: com_content XSS Vulnerability
Version Affected: Joomla! 1.5.9 <=
Check: /components/com_content/
Exploit: A XSS vulnerability exists in the category view of com_content.
Vulnerable? No

# 31
Info -> CoreComponent: com_installer CSRF Vulnerability
Versions effected: Joomla! 1.5.0 Beta
Check: /administrator/components/com_installer/
Exploit: N/A
Vulnerable? No

# 32
Info -> CoreComponent: com_search Memory Comsumption DoS Vulnerability
```

```
Versions effected: Joomla! 1.5.0 Beta
Check: /components/com_search/
Exploit: N/A
Vulnerable? No

# 33
Info -> CoreComponent: com_poll (mosmsg) Memory Consumption DOS Vulnerability
Versions effected: 1.0.7 <=
Check: /components/com_poll/
Exploit: Send request  /index.php?
option=com_poll&task=results&id=14&mosmsg=DOS@HERE<<>AAA<><>
Vulnerable? No

# 34
Info -> CoreComponent: com_banners Blind SQL Injection Vulnerability
Versions effected: N/A
Check: /components/com_banners/
Exploit: /index.php?option=com_banners&task=archivesection&id=0'+and
+'1'='1::/index.php?option=com_banners&task=archivesection&id=0'+and+'1'='2
Vulnerable? Yes

# 35
Info -> CoreComponent: com_mailto timeout Vulnerability
Versions effected: 1.5.13 <=
Check: /components/com_mailto/
Exploit: [Requires a valid user account] In com_mailto, it was possible to
bypass timeout protection against sending automated emails.
Vulnerable? Yes

# 36
Info -> Component: Dada Mail Manager Component Remote File Inclusion Vulnerability
Version Affected: 2.6 <=
Check: /administrator/components/
Exploit: /administrator/components/com_dadamail/config.dadamail.php?GLOBALS
[mosConfig_absolute_path]=
Vulnerable? No


There are 6 vulnerable points in 36 found entries!
~[*] Time Taken: 6 min and 9 sec
```

F
MD5 ALGORTITHM DESCRIPTION

This appendix is an extraction of RFC1321, which encompasses Ron Rivest's MD5 message-digest algorithm [55].

[. . .] We begin by supposing that we have a b-bit message as input, and that we wish to find its message digest. Here b is an arbitrary nonnegative integer; b may be zero, it need not be a multiple of eight, and it may be arbitrarily large. We imagine the bits of the message written down as follows:

$m_0 m_1 ... m_{b-1}$

The following five steps are performed to compute the message digest of the message.

Step 1: Append Padding Bits

The message is "padded" (extended) so that its length (in bits) is congruent to 448, modulo 512. That is, the message is extended so that it is just 64 bits shy of being a multiple of 512 bits long. Padding is always performed, even if the length of the message is already congruent to 448, modulo 512.

Padding is performed as follows: a single "1" bit is appended to the message, and then "0" bits are appended so that the length in bits of the padded message becomes congruent to 448, modulo 512. In all, at least one bit and at most 512 bits are appended.

Step 2: Append Length

A 64-bit representation of b (the length of the message before the padding bits were added) is appended to the result of the previous step. In the unlikely event that b is greater than 2^{64}, then only the low-order 64 bits of b are used. (These bits are appended as two 32-bit words and appended low-order word first in accordance with the previous conventions.)

At this point the resulting message (after padding with bits and with b) has a length that is an exact multiple of 512 bits. Equivalently, this message has a length that is an exact multiple of 16 (32-bit) words. Let M[0 ... N-1] denote the words of the resulting message, where N is a multiple of 16.

Step 3: Initialize MD Buffer

A four-word buffer (A,B,C,D) is used to compute the message digest. Here each of A, B, C, D is a 32-bit register. These registers are initialized to the following values in hexadecimal, low-order bytes first):

```
word A: 01 23 45 67
word B: 89 ab cd ef
word C: fe dc ba 98
word D: 76 54 32 10
```

Step 4: Process Message in 16-Word Blocks

We first define four auxiliary functions that each take as input three 32-bit words and produce as output one 32-bit word.

```
F(X,Y,Z) = XY v not(X) Z
G(X,Y,Z) = XZ v Y not(Z)
H(X,Y,Z) = X xor Y xor Z
I(X,Y,Z) = Y xor (X v not(Z))
```

In each bit position F acts as a conditional: if X then Y else Z. The function F could have been defined using + instead of v since XY and not(X)Z will never have 1's in the same bit position.) It is interesting to note that if the bits of X, Y, and Z are independent and unbiased, the each bit of F(X,Y,Z) will be independent and unbiased.

The functions G, H, and I are similar to the function F, in that they act in "bitwise parallel" to produce their output from the bits of X, Y, and Z, in such a manner that if the corresponding bits of X, Y, and Z are independent and unbiased, then each bit of G(X,Y,Z), H(X,Y,Z), and I(X,Y,Z) will be independent and unbiased. Note that the function H is the bit-wise "xor" or "parity" function of its inputs.

This step uses a 64-element table T[1 ... 64] constructed from the sine function. Let T[i] denote the i-th element of the table, which is equal to the integer part of 4294967296 times abs(sin(i)), where i is in radians. The elements of the table are given in the appendix.

Do the following:

```
/* Process each 16-word block. */
For i = 0 to N/16-1 do
```

```
/* Copy block i into X. */
For j = 0 to 15 do
  Set X[j] to M[i*16+j].
end /* of loop on j */

/* Save A as AA, B as BB, C as CC, and D as DD. */
AA = A
BB = B
CC = C
DD = D

/* Round 1. */
/* Let [abcd k s i] denote the operation
     a = b + ((a + F(b,c,d) + X[k] + T[i]) <<< s). */
/* Do the following 16 operations. */
[ABCD  0  7  1]  [DABC  1 12  2]  [CDAB  2 17  3]  [BCDA  3 22  4]
[ABCD  4  7  5]  [DABC  5 12  6]  [CDAB  6 17  7]  [BCDA  7 22  8]
[ABCD  8  7  9]  [DABC  9 12 10]  [CDAB 10 17 11]  [BCDA 11 22 12]
[ABCD 12  7 13]  [DABC 13 12 14]  [CDAB 14 17 15]  [BCDA 15 22 16]

/* Round 2. */
/* Let [abcd k s i] denote the operation
     a = b + ((a + G(b,c,d) + X[k] + T[i]) <<< s). */
/* Do the following 16 operations. */
[ABCD  1  5 17]  [DABC  6  9 18]  [CDAB 11 14 19]  [BCDA  0 20 20]
[ABCD  5  5 21]  [DABC 10  9 22]  [CDAB 15 14 23]  [BCDA  4 20 24]
[ABCD  9  5 25]  [DABC 14  9 26]  [CDAB  3 14 27]  [BCDA  8 20 28]
[ABCD 13  5 29]  [DABC  2  9 30]  [CDAB  7 14 31]  [BCDA 12 20 32]

/* Round 3. */
/* Let [abcd k s t] denote the operation
     a = b + ((a + H(b,c,d) + X[k] + T[i]) <<< s). */
/* Do the following 16 operations. */
[ABCD  5  4 33]  [DABC  8 11 34]  [CDAB 11 16 35]  [BCDA 14 23 36]
[ABCD  1  4 37]  [DABC  4 11 38]  [CDAB  7 16 39]  [BCDA 10 23 40]
[ABCD 13  4 41]  [DABC  0 11 42]  [CDAB  3 16 43]  [BCDA  6 23 44]
[ABCD  9  4 45]  [DABC 12 11 46]  [CDAB 15 16 47]  [BCDA  2 23 48]

/* Round 4. */
/* Let [abcd k s t] denote the operation
     a = b + ((a + I(b,c,d) + X[k] + T[i]) <<< s). */
/* Do the following 16 operations. */
[ABCD  0  6 49]  [DABC  7 10 50]  [CDAB 14 15 51]  [BCDA  5 21 52]
[ABCD 12  6 53]  [DABC  3 10 54]  [CDAB 10 15 55]  [BCDA  1 21 56]
```

```
  [ABCD  8  6 57]  [DABC 15 10 58]  [CDAB  6 15 59]  [BCDA 13 21 60]
  [ABCD  4  6 61]  [DABC 11 10 62]  [CDAB  2 15 63]  [BCDA  9 21 64]

  /* Then perform the following additions. (That is increment each
     of the four registers by the value it had before this block
     was started.) */
  A = A + AA
  B = B + BB
  C = C + CC
  D = D + DD

end /* of loop on i */
```

Step 5: Output

The message digest produced as output is A, B, C, D. That is, we begin with the low-order byte of A, and end with the high-order byte of D. [. . .]