Large Scale Vulnerability Scanning

Development of a large-scale web scanner for detecting vulnerabilities

Torjus Dahle

Thesis submitted for the degree of
Master in Informatics: Programming and Networks
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2020

Large Scale Vulnerability Scanning

Development of a large-scale web scanner for detecting vulnerabilities

Torjus Dahle

Large Scale Vulnerability Scanning

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Abstract

The number of services that connect to the Internet is steadily increasing. Applications integrate with each other more than ever before. As such, the possible attack surface of a given entity is ever increasing as well. Often, a single vulnerability or weakness in security can seriously undermine the security of entire systems. Other times, several flaws in tandem can prove fatal when chained together by a proficient attacker.

This thesis explores the use of non-intrusive methods for vulnerability discovery. Several vulnerabilities and methods for detecting them are discussed, and a proof of concept (PoC) scanner that can detect some vulnerabilities, while still being non-intrusive, is implemented.

Several methods were applied to the Alexa top 1 million sites on the WWW. And a fraction of sites proved to be vulnerable to each of the vulnerabilities the scanner scans for. This points toward the possible conclusion that even big actors on the Internet need to take fundamental security elements seriously. The so-called "low hanging fruit" is still out there; many entities' digital security can improve considerably with simple security measures. There is no need for expensive solutions, like advanced intrusion detection systems, when one is better served focusing on the basics.

Contents

List of Figures

List of Tables

Preface

I have been fascinated by computer security and privacy for as long as I can remember. With these topics becoming increasingly relevant in a digital world, they garner more attention in the press, and the general public as a whole. Many impactful events have taken place in recent years; Stuxnet, the Snowden revelations, the 2017 Equifax data breach impacting 140 million Americans. There are continuous developments and new discoveries all the time, which makes this an exciting field of research.

First of all, I want to thank my supervisors, Nils Gruschka and Laszlo Erdodi. Nils, especially for his guidance and patience in meetings along the way.

I also would like to thank my lovely family for their support, and my girlfriend, Ingrid, for supporting me and allowing me to spend some time on this project.

Chapter 1

Introduction

For criminals, the Web has become a place to participate in various cybercrimes and spread malware. Common activities include identity theft, fraud, espionage and intelligence gathering. [8] Web-based vulnerabilities outnumbered buffer overflows as early as 2007, [17] and as measured by Google, about one in ten webpages may contain malicious code. [7] Most Web-based attacks take place on legitimate websites, and most of them, as measured by Unit 42, are hosted in the United States, China, Hong Kong, and Russia. [66] The highest-ranking of all malware threats are injection attacks against websites according to OWASP. [45] HTML and uniform resource identifiers (URIs) combined with JavaScript, made web sites vulnerable to attacks like Cross-Site Scripting (XSS) [21, p. 68–69] and were worsened to some degree by Web 2.0 and Ajax Web design that rely on the use of scripts for interacting with web content. [55] Today according to one estimate, 70% of all websites are open to XSS attacks on their users. [9] Phishing is another common threat to the Web. In February 2013, RSA estimated the global losses from phishing at $1.5 billion in 2012. [1] Two examples of phishing methods are covert redirect and open redirect. These are used in phishing attacks that make links appear legitimate, but actually redirect a victim to an attacker's website.

More and more services are moving to the cloud. The shift towards the cloud helps tackle some old challenges; flexibility, scalability, and more but brings new ones as well. Sophos believes that the vast majority of security incidents involving cloud computing platforms result from misconfiguration. The platforms themselves are complex, and change frequently, which can make it challenging to understand the ramifications or consequences of toggling a specific setting in an Amazon S3 bucket, as an example. [61, p. 19] All of these factors contribute to making it easier to commit simple misconfigurations.

In 2020 large scale changes in the use of telecommunications took place at the time of the coronavirus outbreak. More people were staying at home and working remotely. As a result, general Internet usage skyrocketed. There were also changes with security implications. Notably Firefox (among other browsers) decided to re-enable Transport Layer Security (TLS) 1.0 and 1.1 to allegedly "enable access to critical government sites

sharing COVID19 information". [20] The increase in people working from home caused a surge in the use of video chat services and services assisting in remote work, like virtual private networks (VPNs) and remote login services such as Remote Desktop Protocol (RDP). As stated in a recent Shodan blog post, The number of devices exposing RDP directly to the Internet saw a sharp increase as organizations started to work remotely. 8% of these services remain vulnerable to BlueKeep (CVE-2019-0708). [38] It is generally considered by security experts to be malpractice to expose RDP on internal endpoints to the public-facing Internet. According to Sophos, attacks targeting RDP have been at the core of some of the largest and most painful ransomware incidents they have investigated in the past year. [61, p. 15]

Amid the coronavirus outbreak, in March 2020, two 0-day vulnerabilities were noticed being actively exploited in targeted attacks in the wild. The two vulnerabilities, known as CVE-2020-1020 and CVE-2020-0938 were classified as critical and allow remote code execution (RCE) and resided in the Windows Adobe Type Manager Library. The timing was particularly unlucky, as more people than usual were working on their computers from home, many using their Windows computers. On top of that, the patch Tuesday of April was on the latest day possible; April 14th. Microsoft has a patch schedule where patches come out every second Tuesday of the month. Additionally, the company stated that it would not fix the vulnerabilities before then.

In 2020, the number of daily active users of Zoom rose 67% from the start of the year to mid-March as schools and companies adopted the platform for remote work in response to the coronavirus pandemic. [34]

With the rise of videoconferencing, incidents of "Zoombombing," where participants unexpectedly appear in conferences, and in some cases, send pornography or other offensive material to other attendees, have occurred, causing some organizations to abandon the use of Zoom.

Google notably banned Zoom from company-owned computers, and SpaceX and NASA banned employees from using Zoom. [72] Zoom's data security and privacy practices have also come under increased scrutiny. Consequently, Zoom's CEO released a statement apologizing for the security issues. Some of the issues were a result of Zoom having been designed for "large institutions with full IT support." To combat these issues, Zoom has said it would focus on data privacy and issue a transparency report. Zoom published a guide to reduce the chances of Zoombombing and other breaches of security or privacy on the platform. The company has also actively taken steps to make its services more secure.

Despite recent security and privacy issues, many academic institutions and organizations use zoom as their primary provider of video conferences and meetings, including the university of Oslo (UiO). [67]

The 2020 coronavirus outbreak, although tragic, had some interesting effects from an information security perspective, both with regards to the change in usage patterns during a crisis and with malicious actors taking advantage of the chaos of the situation. It was also likely not the only pandemic humanity will face in the near future. [62]

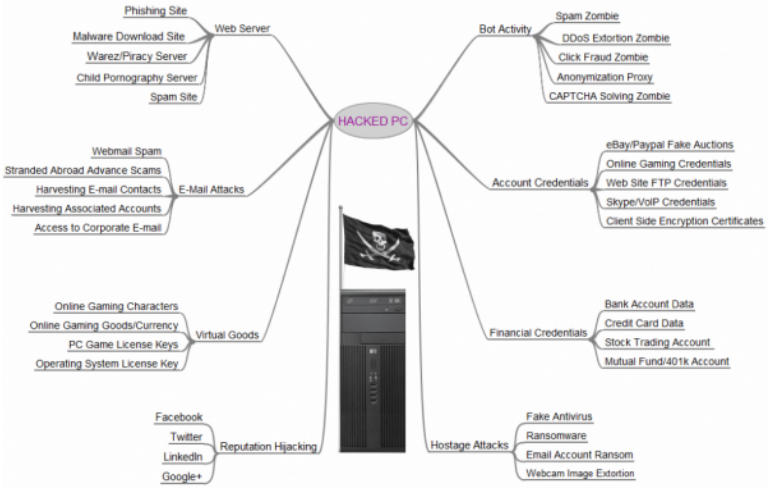

Figure 1.1: Attackers have plenty of uses for a compromised computer. [30]

1.1 Motivation

A large part of websites today is still vulnerable to attacks. About 38% of them are vulnerable to XSS attacks attack [68]. OWASP has a list of the top then vulnerabilities in web applications worldwide at any given time.

Mostly the same ten security risks have remained on top of the list for the last ten years with minor variations. [45] [46] As discussed in chapter 5, simple security mistakes do occur, even in the top 1000 visited websites in the world.

For cybercriminals, a compromised computer can be utilized for a vast array of nefarious purposes, as illustrated by figure 1.1. Their motivations can be many, ranging from financial or political to fame and fortune. As the cybercrime industry keeps growing, one would expect to find an increasingly motivated opposing force that wants to protect its systems.

Enumeration of networks and vulnerability scanning are two of the first and most important steps of any practical information security assessment. It is used by malicious actors to find ways to harm, find vulnerabilities to exploit, open information, and so on. On the other hand, security professionals also use the techniques to find vulnerabilities to patch, and weaknesses to address within a network or organization.

Thorough vulnerability assessment is still an undervalued and often forgotten process that can help detect many of these security defects. This is the motivation for exploring methods for data collection and vulnerability discovery in chapter 3. And in this day and age, with an increasing use of cloud solutions enabling rapid changes in infrastructure, there are significant benefits to automating the process of scanning for vulnerable services and security misconfigurations. Chapter 4 goes into designing and implementing a scanner for large scale automated scans.

Misconfiguration is the primary driver of security incidents in the cloud according to Sophos. [61, p. 19] That makes processes, tools, and routines for detecting such mistakes vital.

Given that many of these security risks continue to live on and are prevalent in the wild, having a web scanner to detect the presence of one or more of these is very beneficial to a potential threat actor. In the same vein, if one can write a scanner that can somewhat reliably detect common security misconfigurations or any of the top ten OWASP risks, chances are some vulnerabilities can be detected, even in the world's most popular sites.

1.2 Objectives

The focus is going to be on the security of on the WWW, and especially the exploration of methods for non-intrusive vulnerability scanning and the development of a scanner that is capable of detecting some vulnerabilities non-intrusively.

With the following research questions:

1. What vulnerabilities can be detected non-intrusively?
2. What methods can be used to identify these vulnerabilities?
3. Can a scanner be made to detect some of these vulnerabilities non-intrusively at a large scale?

1.3 Overview

Chapter 2 contains definitions, previous work done on the subject, background knowledge, and discusses some vulnerabilities. Chapter 3 discusses known techniques for data collection and vulnerability discovery. Chapter 4 explains the design, implementation, and features of the developed PoC scanner. Chapter 5 discusses some of the discoveries made while running the scanner. Chapter 6 discusses some of the possible tactics for mitigating security risks on the WWW, and the future of the security landscape on the Web. And finally, chapter 7 explains what was not done and summarizes.

Chapter 2

Background

2.1 Definitions

2.1.1 Vulnerability

vulnerability is the existence of a weakness, design, or implementation error that can lead to an unexpected, undesirable event compromising the security of the computer system, network, application, or protocol involved [65].

2.1.2 Vulnerability scanner

A vulnerability scanner is a program that performs the diagnostic phase of vulnerability analysis, also known as vulnerability assessment. Vulnerability analysis defines, identifies, and classifies the security holes (vulnerabilities) in a computer, server, network, or communications infrastructure. Additionally, vulnerability analysis can forecast the effectiveness of proposed countermeasures, and evaluate their effectiveness when put to use. [56] [74]

2.1.3 Vulnerability scanning

Vulnerability scanning is, according to A dictionary of the Internet, "The process of scanning the computers in a network to discover any weak point through which an intruder can pass." [27]

Vulnerability scanning is the process of gathering of information or use of tools like scripts and vulnerability scanners to detect vulnerabilities. Vulnerability scanning is a key component to successful penetration and exploitation of computer systems. It is usually one of the first steps in the process of executing an attack. Vulnerability scanning is generally considered an important part of a vulnerability assessment. [73]

Vulnerability scans can be conducted from inside a network, by setting up a scanner on a local area network (LAN) or from outside the network through the Internet.

System administrators usually scan their network from the inside, using either an active or passive vulnerability scanner (definition below).

Attackers mostly scan the network from outside. If they can get a foothold inside the network, they might try to scan the network from the inside.

2.1.4 Active/passive scanning

In a broad sense, there are two approaches to overseeing security from a system administrator/blue team perspective. These two approaches are active and passive vulnerability assessment or scanning. The active approach encompasses everything a system administrator does to foil system breaches, while the passive (or monitoring) approach entails all the ways the organization oversees system security. Passive scanning can also include passively collecting traffic information within a network. This information can then be used to infer information about systems, versions, vulnerabilities.

Passive vulnerability assessment is a relatively new concept, [52] that might see increasing popularity in the future. The idea behind passive vulnerability assessment is that rather than providing proactive probing of networks by generating test traffic, one infers the vulnerabilities by sniffing regular network traffic. Passive vulnerability scanning or assessment finds information in the course of inspecting packets and can detect anomalies or alert upon known bad signatures.

Active scanning, on the other hand, is the process of actively scanning the network by doing activities like sending “fingerprint” packages to identify operating systems and web server software versions and scanning for open ports. Active scanners send transmissions to the network’s nodes, examining the responses they receive to evaluate whether a specific node represents a weak point within the network. [52] [6].

All actions taken by an attacker are active in some sense. No information flows automatically from victim to attacker without some form of action. From an attacker’s perspective, the concepts are the same, but the methods are different. An attacker can passively collect information, or use active scanning or exploitation to achieve their goal. Passive in this context encompasses every method used to gather information about an entity without making any direct requests to it. Completely passive information gathering is executed via "indirect" means; some examples are domain name system (DNS) lookups and the use of information that has been cached by search engines.

2.1.5 Intrusive vs non-intrusive scanning

A vulnerability scanner can execute intrusive or non-intrusive tests. An intrusive test tries to exercise the vulnerability, which can crash or alter the remote target. A non-intrusive test attempts not to cause any harm to the target system. The test will usually check the remote service version, whether the any vulnerable options are enabled, or if the service is configured insecurely. Intrusive tests are generally a lot more accurate, but cannot be performed legally in a production environment without a contract. In general, a non-intrusive test cannot determine with

complete certainty if a service installed is vulnerable, only if it might be vulnerable. [11, p. 55]

2.2 Related work

2.2.1 Papers

Most papers on vulnerability scanning were done within isolated networks, and there are few open-world vulnerability scanning papers. There is one paper discussing the use of Shodan and a Shodan-based vulnerability assessment tool [23]. This paper is fairly detailed and includes some statistics for the scanner.

Another paper named "How Bad Can It Git? Characterizing Secret Leakage in Public GitHub Repositories", analyzes secret leakage on Github. The researchers found 100 000 repositories that leaked secrets, and that thousands of new, unique secrets were leaked every day. [40]

A paper, named "Understanding the Security Threats of Esoteric Subdomain Takeover and Prevention Scheme" explains subdomain takeover attacks and prevention. [51]

On the topic of security misconfiguration, there are several resources. A paper by Sulatycki and Fernandez seeks to formalize threat patterns by describing their different aspects. The two threat patterns they describe are (1) taking advantage of security misconfiguration and (2) gaining access to sensitive data. [64] These are two broad patterns that can be recognized in chapter 3 and 4. Another paper presents a tool that audits the security configuration for web applications, and can automatically adjust security configuration settings. [19]

2.2.2 Conference presentations

Looking for vulnerabilities non-intrusively was discussed in a talk called “Alexa Top 1 Million Security - Hacking the Big Ones” by David Wind from It.sec [76]. It focused on simple security mistakes with a high-security impact. Some of the vulnerabilities the uncovered were: Subdomain takeover vulnerabilities, exposed credentials, exposed source code, cross-origin resource sharing (CORS) misconfiguration, exposed Amazon Web Services (AWS) S3 buckets. However, the results were largely censored, but some tools and methods were briefly mentioned in the slides. They found several well-known companies like Hp and Lenovo vulnerable to subdomain takeover attacks. Various other sites in the top 1 million were vulnerable to relatively simple attacks or exposed data publicly that were not meant for sharing. The occurrences of these simple mistakes indicate that even the big actors can benefit from focusing on getting the basics of security right.

2.2.3 Blog posts and news sites

There is a high volume of blog posts written by enthusiasts, bounty hunters and penetration testers on the topic of looking for vulnerabilities non-intrusively. The variety of different methods and techniques that can be used makes the demographic looking for vulnerabilities non-intrusively a vibrant community.

A blogger who calls himself xxdesmus, that makes blog posts about data leaks, recently wrote a post on a Thai Database that leaked 8.3 Billion Internet Records. [77] He discovered the exposed ElasticSearch database while browsing BinaryEdge and Shodan. In a news story posted by Hanno Böck, Tiles in Microsoft Windows were subject to a subdomain takeover attack. Microsoft allegedly deleted the nameserver record without providing a comment after being informed of the vulnerability. [10] In a may 2020 blog post by Tillson Galloway, he boasts that he has submitted 30 disclosure reports from Github secret leaks alone. He also published a tool, GitHound, a tool that automates the process of finding secrets across GitHub. [22]

2.3 Relevant concepts and material

2.3.1 The basics of penetration testing

The basics of penetration testing, according to the Pentest standard [42] are:

1. Pre-engagement Interactions
2. Intelligence Gathering
3. Threat Modelling
4. Vulnerability Analysis
5. Exploitation
6. Post Exploitation
7. Reporting

This model gives an idea of where intelligence gathering (2) and vulnerability analysis (4) fit into the penetration testing process. These two concepts are important for the development of the scanner later on.

2.3.2 OWASP TOP 10

The Open Web Application Security Project (OWASP) is an open community dedicated to enabling organizations to develop, purchase, and maintain applications and application programming interfaces (APIs) that can be trusted. The OWASP top 10: The Ten Most Critical Web Application Security Risks[1] has become an authoritative source in the web application

[1] https://owasp.org/www-project-top-ten/

security community. These are only the top ten security risks out there, but they can serve as a start to the discussion on the topic of which vulnerabilities can be detected or exploited non-intrusively. Since they are held in high regard in terms of web application security, some time will be spent walking through them in the following paragraphs. Then a judgment of whether the presence of a security risk can be done non-intrusively or not will be made in table 2.1.

Injection Injection vulnerabilities, such as SQL, NoSQL, OS, and LDAP injection, take place when untrusted data is passed to an interpreter as part of a command or query. The attacker's hostile data, which is usually crafted to trick the system into treating it as instructions, can trick the interpreter into executing unintended commands or accessing data without proper authorization.

The presence of vulnerability to injection attacks is something that would most often be detected intrusively. This is because the attacker is trying to provoke unintended behavior by supplying the system with malign input. Such attacks can lead to database tables being dropped or system crashes.

Broken Authentication Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume the identity of another user temporarily or permanently.

Broken authentication is a category of vulnerability that encompasses various defects within an application's login mechanism, which may enable an attacker to guess weak passwords, conduct a brute-force attack, or bypass the login entirely. Such methods are seen as intrusive in general.

Sensitive Data Exposure Numerous web applications and APIs do not adequately protect sensitive data, such as financial, healthcare, and personally identifiable information (PII). Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised if it is not sufficiently protected, as an example, it might require encryption at rest or in transit, and requires special precautions when exchanged with the browser.

Sensitive data can be exposed in ways that are detectable with non-intrusive methods. A typical case of this happening is when a file is unknowingly saved to a public directory on a web server.

Xml External Entities (XXE) Many older or poorly configured Extensible Markup Language (XML) processors evaluate external entity references within XML documents. External entities can be used to leak internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.

XXE attacks generally fall into the intrusive category.

Broken Access Control Restrictions on what authenticated users are allowed to do are often not adequately enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access to other users' accounts, read sensitive files, modify other users' data, and change access rights.

Violating access control is generally seen as an intrusive action. However, resources are not access controlled at all in some cases (most likely because it was not properly implemented in the first place). In such circumstances, it would generally be seen as a non-intrusive action to access such resources. In that sense, the presence of this vulnerability can be detected both intrusively and non-intrusively, but in general, it can be considered an intrusive action to check for such vulnerabilities.

Security Misconfiguration Security misconfiguration is the most commonly seen issue. It is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information used during development. All operating systems, frameworks, libraries, and applications have to be securely configured, and they must be patched/upgraded in a timely fashion. [45, 61]

Cross-Site Scripting (XSS) XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create Hypertext Markup Language (HTML) or JavaScript. 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.

Testing for XXS vulnerabilities is in general seen as an intrusive action, but some XSS vulnerabilities can be tested locally (and therefore non-intrusively).

Insecure Deserialization Insecure deserialization often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks.

Testing for insecure deserialization is in its very nature intrusive, as it is based on giving malign input similar to injection attacks.

Using Components with Known Vulnerabilities Components like libraries, frameworks, and other software modules, run with the same privileges as the application they belong to. If a vulnerable component is exploited, such an attack can facilitate severe data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.

Detection of vulnerable components can be done non-intrusively. As an example, one can grab the banner of a service containing version

Table 2.1: The OWASP top 10 security risks and their possibility of being detected via intrusive or non-intrusive means.

Risk	non-intrusive	intrusive
A1:2017-Injection		✓
A2:2017-Broken Authentication		✓
A3:2017-Sensitive Data Exposure	✓	
A4:2017-XML External Entities (XXE)		✓
A5:2017-Broken Access Control		✓
A6:2017-Security Misconfiguration	✓	
A7:2017-Cross-Site Scripting (XSS)		✓
A8:2017-Insecure Deserialization		✓
A9:2017-Using Components with Known Vulnerabilities	✓	
A10:2017-Insufficient Logging & Monitoring		

information. Then known vulnerabilities for the given version of the component can be looked up.

Insufficient Logging and Monitoring Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.

Insufficient logging and monitoring is not something an attacker would usually detect via either intrusive or non-intrusive scanning or checking.

As summarized in table 2.1, it should be theoretically possible to exploit or detect the presence of quite a few of these risks passively.

2.3.3 Non-intrusive vulnerability detection

Numerous vulnerabilities can be detected non-intrusively.

Unregistered subdomains

In the DNS hierarchy, a subdomain is a domain that is part of another domain. At the top of the hierarchy are the top level domains (TLDs) which are installed in the DNS root zone. Examples of these are com, org, and no.

Colloquially, one often refers to the `subdomain`-part of `subdomain.domain.com` when speaking of a subdomain. While technically, every domain that is not a top-level domain is a subdomain. So for example: `mn.uio.no` is a subdomain of `uio.no` which again is a subdomain of `no`.

As a company or organization accumulates lots of subdomains, it gets harder to keep track of them. This is the primary reason why many companies are vulnerable to subdomain takeovers today. What happens is that a company utilizes some third-party CMS/Content/Cloud Provider and points their subdomains to these platforms. If they ever forget to configure the third-party service or deregister from that server, an attacker can take over that hostname with the third party.

For example, a company registers an Amazon S3 bucket with the name `testlab.s3.amazonaws.com`. It then has its company's subdomain `testlab.company.com` point to `testlab.s3.amazonaws.com`. A year later, it no longer need the S3 bucket `testlab.s3.amazonaws.com` and deregisters it but forgets the CNAME redirect for `testlab.company.com`. A malicious actor can now go to AWS and set up the previously registered address, `testlab.s3.amazon.com`, and have a valid S3 bucket on the victim's domain [29, p. 40].

Having control over a subdomain allows for various attack scenarios. An attacker can use the same Secure Socket Layer (SSL)-certificates as the base domain, impersonating the victim while, for example, stealing login information that unknowing users type in or serve malicious content. Numerous other attacks are also possible depending on the victim's configuration of cookies, CORS, Oauth whitelisting, CSP, configurations that allow clickjacking, interception of emails, and so on. In the case of cookies: `subdomain.example.com` can modify cookies scoped to `example.com`. This can potentially allow an attacker to hijack the victim's session on the base name. If the base name is vulnerable to session fixation and also uses HTTPOnly cookies, the attacker can set a cookie and when the victim restarts their browser the malicious cookie will take precedence over the newly generated cookie because cookies are sorted by age. [2]

Exposed credentials

Information that is not meant to be public is sometimes exposed by mistake. One common scenario is that developers unknowingly commit secrets into version control system (VCS). Even if this is later discovered, and later removed from VCS the secret is still in the history. Most people do not take the time to rewrite their history to remove any trace of the leak. This can be exploited by attackers who search through the history of open software repositories for such secrets.

Exposed source code

Another case of unintended sharing of information happens when the source code of a website is hosted on its own server. This happens when access to server directories is not restricted correctly. Attackers can use the information found in the source code to steal and impersonate the site, steal intellectual property (IP) from the source code, find vulnerabilities in the application, and more. Depending on the information stored in the source

code, this can also mean leaking database information with customer data and various passwords.

Open cloud storage solutions - AWS S3 buckets

A lot of cloud storage solutions are left open for anyone to access. Perhaps a store is left unauthenticated while testing, and then later forgotten, or maybe the appropriate security controls were never implemented. Attackers can look for such open buckets and, depending on how the bucket is access controlled, read or write to files.

One example of such a storage solution where many buckets have been configured without the proper restrictions is Amazon S3.

Amazon S3 or Amazon Simple Storage Service is a service offered by Amazon Web Services (AWS) that provides object storage through a web service interface. Amazon S3 uses the same scalable storage infrastructure that Amazon.com uses to run its global e-commerce network. Amazon S3 can be employed to store any type of object which allows for uses like storage for Internet applications, backup and recovery, disaster recovery, data archives, data lakes for analytics, and hybrid cloud storage.

One could understand how such a bucket and its contents can be of high value to attackers.

CORS misconfiguration

To be able to securely browse the web with scripts enabled, it is imperative that JavaScript that is running on one domain can only read data from that domain. If this restriction was not enforced, a script running on any page could open a tab or window to the current users' email provider or Facebook profile and steal private information.

This is the very reason JavaScript (and other scripts) are only allowed to send requests and read data from the domain the request originates from. So functions in JavaScript are allowed to make a request to the domain it is hosted on and read the data. Most API calls work like this. This allows data to be processed safely in JavaScript instead of being directly displayed on the webpage.

However, in some cases, we do want to permit sending requests to other domains. For example, there are public APIs that allow anyone to query them. These APIs by necessity also have to allow JavaScript on any domain to send requests to them. In cases such as these, the browser needs to be able to send requests to other domains. cross-origin resource sharing (CORS) solves this. CORS is a header set by the web server. The header regulates precisely which domains are allowed to send requests to the web server.

Unfortunately, it is possible to misconfigure CORS such that its security is undermined. This is most often seen when bypasses of CORS have been made during development, or erroneous checking of the origin is in place, perhaps because of a regular expression that is too permissive when validating the origin. [16]

Vulnerable tech-stacks

A server can run old and/or exploitable technologies. This can be because of outdated software like an outdated operating system, web service stack, dependencies, and imported modules. Their existence on a web server can sometimes be detected with non-intrusive scans, but oftentimes they have to be manually verified.

Vulnerable web applications

The web applications themselves can be vulnerable to various attacks. The OWASP section mentions the ten greatest security risks for web applications. The challenge with a lot of vulnerabilities in web applications is that their presence often only can be detected or proved with intrusive methods. This makes it impossible to scan for some of them with a non-intrusive scanner.

Lack of defence against social engineering

While not a vulnerability, but rather a method of attack, social engineering can be used to topple sophisticated security solutions. With the vast amount of information openly available on the Internet today, an attacker can gain much insight before performing attacks. The collected information can also be leveraged to perform sophisticated social engineering attacks. A sufficiently motivated attacker can take the time to craft convincing spear-phishing emails, especially if he has some relevant information on the victim. Any user can fall prey to a social engineering attack; it is just a matter of timing and having a convincing story. Security policies, technical solutions, and a healthy amount of scepticism can all be used to stop a social engineering attack in its tracks.

Chapter 3

Methods for data collection and vulnerability discovery

This chapter will map out some methods and tools for data collection and vulnerability discovery.

3.1 Port scannning

Port scanning is the process of determining what ports are open on a host machine. It is one of the most basic forms of information gathering. While scanning for open ports is simple, more advanced techniques for fingerprinting can be used to detect the operating system (OS), software, and even software versions running on the different ports. Port scans can be loud on the network, so advanced attackers generally limit the use of such scans to avoid detection.

While there are old and evolving scanners, new ones pop up from time to time as new needs arise.

Nmap ("Network Mapper") Nmap is a free and open-source utility for network discovery and security auditing. It uses raw IP packets in novel ways to determine what hosts are available on the network, what services (application name and version) those hosts are offering, what operating systems (and OS versions) they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics. Nmap is extensible with scripts that provide more advanced service and vulnerability detection, and other features. It was designed to rapidly scan large networks but works fine against single hosts. [44] All of these factors make it an effective tool for discovering vulnerable services.

Various scans can be performed with Nmap. We can perform a default scan. This scans some of the most commonly used ports on the internet.

```
nmap 192.168.0.104
```

We can scan all the ports of a given host.

```
nmap -p1-65535 192.168.0.104
```

Or we can perform OS detection:

```
nmap -O 192.168.0.104
```

Nmap can also be used to gather detailed information. Increased verbosity level in output, enabling aggressive scan options and probe open ports to determine service and version info.

```
nmap -v -A -sV 192.168.0.104
```

Nmap can also scan an input list of hosts and write to an output file.

```
nmap -iL alexa-top-1000.list -oN nmap_output
```

This can be suitable for scanning a large number of targets.

Quick scanning with MASSCAN This is an Internet-scale port scanner. The developers claim that it can scan the entire Internet in under 6 minutes, transmitting 10 million packets per second, from a single machine.

MASSCAN is a fast scanner, but in practice, it seems to be less accurate than Nmap. Masscan's reliability seems to drop when scanning large ranges. It can be used effectively for doing initial reconnaissance, but other tools - like Nmap can be used for detecting changes in a network (also referred to as diffing) [29, p. 25].

MASSCAN can perform quick scans but requires the use of an Intel 10-Gbps adapter and a proprietary driver called "PF_RING ZC" from ntop, which is available for 150 EUR. It allows for easy configuration of scans with its configuration file format:

```
# My Scan (myscan.conf)
rate =  100000.00
output-format = xml
output-status = all
output-filename = scan.xml
ports = 0-65535
range = 0.0.0.0-255.255.255.255
excludefile = exclude.txt
```

It is simple to run using `masscan -c myscan.conf`. A caveat when using MASSCAN is that target addresses must be IP addresses or simple ranges. They cannot be complex subnet ranges like the ones nmap can use, one example being 10.0.0-255.0-255.

3.2 Scanning for web application vulnerabilities

Burp suite Burp Suite is a popular web vulnerability scanner. It is proprietary but has a free "Community"-edition with some of the essential tools. This feature-rich commercial tool comes at a price of 399 USD per user per year. Its benefits come from the add-ons, modular design, and user development base. As a free alternative, OWASP Zed Attack Proxy (ZAP) is an excellent replacement. [29, p. 51-52]

The tools listed below are open-source. Open-source tools have several advantages compared to proprietary or closed source software: They can be easily inspected, modified, and are free. The next paragraphs take a look at some of the top ones for web application vulnerability scanning.

Web Application Attack and Audit Framework (w3af) Web Application Attack and Audit Framework (w3af) is an open-source web application security scanner that helps developers and penetration testers identify and exploit vulnerabilities in their web applications.

The scanner can identify 200+ vulnerabilities, including Cross-Site Scripting, SQL injection, and OS commanding.

OWASP Zed Attack Proxy (ZAP) OWASP ZAP [49] is a web application security scanner similar to Burp, but wholly open-source and free. It is one of the most active OWASP projects. It can be used for vulnerability assessment, penetration testing, runtime testing, and code review. ZAP can also be used within build pipelines to test the security of applications as part of the continuous integration (CI)/continuous deployment (CD) pipeline.

BuiltWith A web site profiler tool. Upon looking up a page, BuiltWith returns all the technologies it can find on the page. BuiltWith's goal is to help developers, researchers and designers find out what technologies pages are using, which may help them to decide what technologies to implement themselves [29, p. 51].

Retire.JS Scan a web app for the use of vulnerable JavaScript libraries. The goal of Retire.js is to help detect versions with known vulnerabilities [29, p. 51].

Nikto Nikto is an open-Source web server scanner, released under GNU General Public License (GPL), which performs comprehensive tests against web servers for multiple items, including over 6700 potentially dangerous files/programs, checks for outdated versions of over 1250 servers, and version specific problems on over 270 servers. It also checks for server configuration items such as the presence of multiple index files, HTTP server options, and attempts to identify installed web servers and software. Scan items and plugins are frequently updated and can be automatically updated.

Nikto is not designed to be stealthy. It tests a web server in the quickest time possible and is evident in log files or to an intrusion prevention system (IPS)/intrusion detection system (IDS). However, it supports anti-IDS methods, though these seem most suited for testing and not for reliable stealth scanning.

Not every check that Nikito performs is a security problem, though most are. Some items are "info only" type checks that look for things that

may not have a security flaw, but the webmaster or security engineer may not know are present on the server. [43]

3.3 Spidering

Web spiders can be seen as some of the most powerful and useful tools developed for both good and bad purposes on the Internet. A spider serves one primary function, Data Mining. A typical spider (like those of Google) works by crawling a web site one page at a time, gathering and storing the relevant information such as email addresses, meta-tags, hidden form data, Uniform Resource Locator (URL) information, links, etc. The spider then crawls all the links on that page, collecting relevant information on each following page, and so on. Since every page can link to a nearly infinite number of other pages, a spider can crawl thousands of links and pages by starting from one page and recursively following links to other pages. The spider gathers information and stores it into a database as it goes along. The web of paths that is followed is where the term 'spider' is derived from. [47]

Many of the advanced vulnerability scanning tools have spidering features. Some simple tools have to spider as their only purpose. Although noisy, they are great for gathering information in a non-intrusive way.

Burp Suite Burp Suite Spidering: In both the free and paid versions, Burp Suite has a great Spidering tool. Content Discovery: when using the paid version of Burp Suite, one of the favorite discovery tools is under Engagement tools, Discover Content [29, p. 52].

Burp is a smart and efficient discovery tool that looks for directories and files. Several different configurations can be specified for the scan. Active Scan: Runs automated vulnerability scanning on all parameters and tests for multiple web vulnerabilities. [12]

OWASP ZAP Has similar discover and active scan features for spidering as Burp [29, p. 53].

Dirbuster Dirbuster is an old tool to discover files/folders of a web application. [29, p. 53]. This tool is made for finding pages and applications inside web servers. Dirbuster is an inactive OWASP project, which essentially has been forked by the OWASP Zap team and is now available as a Zap add-on. It is written as a multithreaded java application, designed to brute force directories and filenames on servers. [58]

GoBuster GoBuster [54] is a very lightweight, fast directory, and subdomain brute force tool. It is a simple console application, written in Go, as the name suggests. It can brute-force URIs (directories and files) and DNS subdomains [29, p. 52].

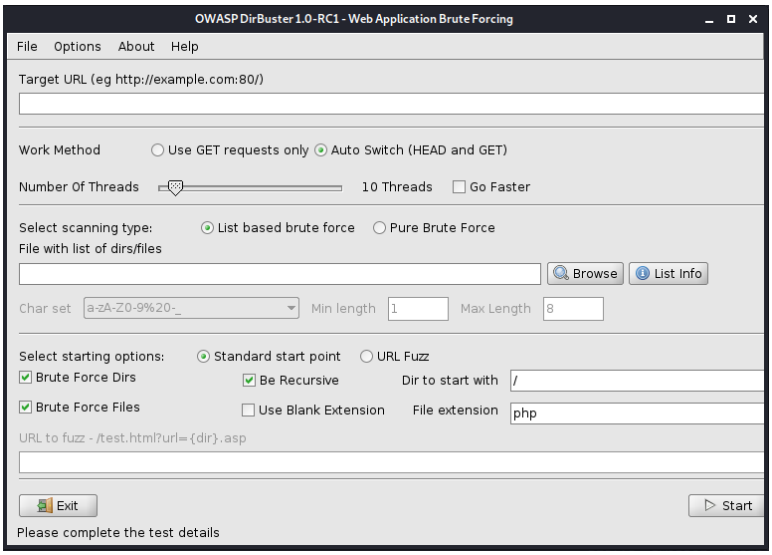

Figure 3.1: Screenshot of the Dirbuster interface

3.4 Vulnerability assessment

There are many ways of doing vulnerability assessments. More often than not, however, some vulnerability assessment tool is going to be used as part of the process. These are very potent and feature-rich tools. They do, however, have some properties that make them less desirable for large scale scanning. The more extensive the scan, the longer it takes to complete - in general. These tools are often made for running long intensive tests that can detect many vulnerabilities. Intensive scans are great for many uses, but the scans take a long time to complete when ran on many hosts. This makes many of these tools "bloated" for the purposes of this thesis since they include much more functionality than is ever going to be used. Additionally, some of them are made to run from within the target network. And most importantly, depending on how they are set up and what scans are used, they can have negative influences on the resources. Vulnerability assessment tools usually exercise the vulnerabilities they scan for. This requires explicit permission from the system owner.

Tenable (Nessus) Nessus by many considered the de-facto industry standard vulnerability assessment solution for security practitioners. [41] This is a rather expensive tool, with a cost of approximately 26 000 NOK for a one-year license.

There are also scripts that help automate Nessus available, like AutoNessus. [53] AutoNessus communicates with the Nessus API to issue various commands to Nessus. It can be set up to run as a scheduled task or cron-job to start or stop scans at the desired time.

Some other popular vulnerability assessment tools are Qualys, Rapid7, Trustwave, IBM, WhiteHat Security, CA Technologies (Veracode) and Checkmarx.

3.5 Scanners on the web

Today, more and more people are moving to cloud- and web-based solutions for performing scans. The hackers and security researchers are moving with them.

3.5.1 Scan databases

Several databases containing information from scans are also available online; these are often called scan-databases. These databases can contain a vast amount of information ranging from the results of simple port scans, or scans for specific vulnerabilities.

Internet-wide scan data repository The Internet-Wide Scan Data Repository[1] is a public archive of research data sets that describe the hosts and sites on the Internet. The repository is hosted by the ZMap Team. [78]

3.5.2 Using search engines

There are numerous search engines and databases containing vulnerability scan information out there.

Google Google can be used to find valuable information about potential targets. Using advanced search terms, one can often find sensitive information. These are often called "Google dorks". An extensive collection of these have been organized into a dictionary, called Google Hacking Database. [35] There are even command line interfaces (CLIs)[2] made for finding vulnerabilities through Google (and other) search engines.

Cons of using the Google search engine:

- The Google Search API was deprecated in 2010. What can be used instead is the standard site search: . But doing a high volume of searches quickly gets one flagged as a bot and would require solving captchas at regular intervals. So for a larger volume of requests, one has to use the Custom Search JSON API.
- The new Custom Search JSON API is only meant to search within a single domain, but it is possible to get around that by using wildcards like "*.com/*". The downside to this is that it is a "hacky" solution, and Google might disallow such configuration of the search API at any time.
- The free Google API limits the user to 100 searches per day, with a maximum of ten results per search. For more searches, one has to go with the paid plan. Signing up for billing on the Google API site nets a user $300 free to spend on API calls for 60 days.

[1] https://scans.io/
[2] https://github.com/utiso/dorkbot

- A file called `robots.txt` can be used by sites to stop Google's crawlers from indexing sites. This can be used to hide information that would otherwise be indexed and appear in searches.

Pros:

- Feature-rich search for finding various information on a webpage.
- Cached sites: Even though the information is removed from a given page, it can still be accessible through Google's own cached version.

3.5.3 Using cloud scanners

Shodan.io Shodan is a search engine that allows a user to query for specific types of devices (webcams, routers, servers, and other devices) connected to the Internet using a variety of filters. Some have (humoristically) described it as the search engine of service banners, which are metadata that the server sends back to the client. The banner can contain information about the server software, what options the service supports, a welcome message or anything else that the client can deduce before starting its interaction with the server. [75]

Shodan gathers data mostly from web servers using HTTP/HTTPS - port 80, 8080, 443, 8443, FTP running on port 21, SSH (on port 22), Telnet (port 23), SNMP (port 161), IMAP (ports 143, or (encrypted) 993), SMTP (port 25), SIP (port 5060), and Real-Time Streaming Protocol (RTSP, port 554). The last example, RTSP, is used for streaming video and contributed to Shodan's relative popularity for searching for web cams. [59]

Censys.io Censys is a search engine for finding servers and devices that are exposed publicly on a network. Censys is based on ZMap; it searches data harvested by ZMap. [14]

S3 bucket scanners With the advent of cloud services like Amazons S3 buckets for storing data online, come new security holes. A lot of S3 buckets have their content openly accessible on the Web or are only protected by weak passwords. Tools like S3Scanner[3] [57] have been developed to find open buckets and dump their contents.

3.6 Methods for subdomain discovery

In terms of identifying IP ranges, one can typically lookup the company from public sources like the American Registry for Internet Numbers (ARIN), AFRINIC (Africa), APNIC (Asia), LACNIC (Latin America), and RIPE NCC (Europe). It is possible to lookup the IP address space belonging to owners, search Networks owned by companies, Autonomous System Numbers of an organization, and more.

[3]https://github.com/sa7mon/S3Scanner

These are all publicly available and listed on their servers. One can lookup any hostname or full qualified domain name (FQDN) [4] to find the owner of that domain through many available public sources, one example being Domain Dossier [18].

Subdomains, on the other hand, are not listed publicly. Subdomain information is stored on the target's DNS server versus registered on some central public registration system. One must know what to search for to find a valid subdomain. [29, p. 32]

Manually parsing SSL-certificates It is common that companies do not realize what they have available on the Internet. Especially with the increase of cloud usage, many companies do not have access control lists (ACLs) correctly implemented. They might not know that their servers are publicly facing. These can include Redis databases, Jenkins servers, Tomcat management, NoSQL databases, and more - many of which have led to remote code execution or loss of PII [29, p. 30].

One tool for scraping hostnames from SSL-certificates is sslScrape. [28] This is one of the many ways to build a list of hostnames.

Discover scripts Discover scripts is a collection of Custom bash scripts used to automate various penetration testing tasks, including recon, scanning, parsing, and creating malicious payloads and listeners with Metasploit. It is made for use with Kali Linux and the Penetration Testers Framework (PTF). [5]

An advanced feature of Discover scripts is that it takes the information it gathers and uses that information in the ongoing search. For example, from searching through the public PGP repository, it might identify emails and then use that information to search Have I Been Pwned (through Recon-NG). That will let us know if any passwords have been found through publicly-released compromises. This is very useful to know for an attacker that might have access to previously leaked information. [29, p. 33].

KNOCK Knockpy is a python tool designed to enumerate subdomains on a target domain through a wordlist. Knock is an excellent subdomain scan tool that takes a list of subdomains and checks it to see if it resolves [29, p. 33].

Sublist3r The challenge with Knock is that it is only as good as the wordlist being used. Some companies have unique subdomains that cannot be found through a standard wordlist. The next best resource to go to is search engines. As sites get spidered, files with links get analyzed

[4] A full qualified domain name (FQDN), sometimes also referred to as an absolute domain name, is a domain name that specifies its exact location in the tree hierarchy of the DNS. So as an example: A device with the hostname `myhost` in the parent domain `example.com` has the FQDN `myhost.example.com`. The FQDN uniquely distinguishes the device from any other hosts called `myhost` in other domains.

and scraped public resources become available, which means one can take advantage of scraped and indexed information for finding subdomains. Sublist3r does this. Sublist3r uses different "google dork" style search queries that can look like a bot. This could get us temporarily blacklisted and require us to fill out a captcha with every request, which may limit the results from the scan. [29, p. 33]

SubBrute SubBrute [63] is a community-driven project with the goal of creating the fastest, and most accurate subdomain enumeration tool. SubBrute uses open resolvers as a kind of proxy to circumvent DNS rate-limiting [70]. This design also provides a layer of anonymity, as SubBrute does not send traffic directly to the target's name servers. Not only is SubBrute extremely fast, it performs a DNS spider feature that crawls enumerated DNS records [29, p. 34].

We can also combine SubBrute with MassDNS [37] to perform very high-performance DNS resolution [29, p. 35].

3.7 Subdomain takeover

One tool to check for vulnerable subdomains is called tko-subs [69]. This tool can be used to check whether any of the previously discovered subdomains are pointing to a CMS provider, such as Heroku, Github, Shopify, Amazon S3, Amazon CloudFront, can be taken over [29, p. 40–41]. If a dangling CNAME is found, something like tko-subs can be used to take over Github Pages and Heroku Apps. Otherwise, we would have to do it manually.

Two other tools that can help with domain takeovers are: HostileSubBruteforcer [26] and autoSubTakeover [4] [29, p. 41].

3.8 Publicly exposed information

Github Penetration testers and Red Teams have been able to get passwords, API keys, old source code, internal hostnames/IPs, and more from Github repositories. These either led to a direct compromise or assisted in another attack. Many developers either push code to the wrong repository (sending it to their public repository instead of their company's private repository) or accidentally push sensitive material (like passwords) and then try to remove it. Github tracks every time code is modified or deleted. That means if sensitive code at one time was pushed to a repository and that sensitive file is deleted, it is still tracked in the code changes. If the repository is public, all these changes can be viewed. For finding specific organizations or entities, one can either use Github search to identify individual hostnames/organizational names or even just use simple Google Dork search, for example, `site:github.com + "orgname"` [29, p. 35].

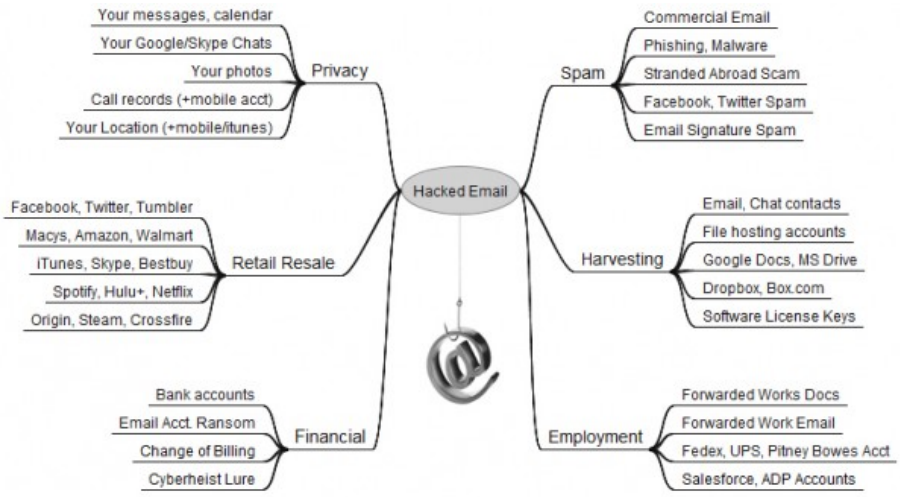

Figure 3.2: Uses of hacked email addresses [31]

Truffle Hog Truffle Hog tool scans different commit histories and branches for high entropy keys and possible secrets and prints them. It is well suited for finding secrets, passwords, keys, and more [29, p. 35].

3.9 Collecting additional information

OSINT open-source intelligence (OSINT) is data collected from publicly available sources to be used in an intelligence context. Where the term "open" refers to overt, publicly available sources. The collection of OSINT can be of great use to attackers, as it can help set up social engineering attacks or otherwise assist in an attack. There are many popular tools for gathering such information out there: Whois, Nslookup, FOCA, theHarvester, Shodan, Maltego, Recon-ng and Censys.

A couple of them explained in more detail as they have functionality that could be of interest to us: Recon-ng is a full-featured reconnaissance framework designed with the goal of providing a powerful environment to conduct open-source web-based reconnaissance quickly and thoroughly. TheHarvester is used to gather OSINT on a company or domain.

Email addresses Email addresses can be used for various purposes. A commonly performed attack is credential stuffing, where known email addresses combined with either passwords from previous breaches, words from dictionaries, or exhaustive search/brute force techniques. Once access is gained, there are a lot of possible uses, as illustrated by figure 3.2.

Knowledge of email addresses can also be used for email bombing. In this type of attack, an attacker tries to hide his true intentions behind massive amounts of spam email. While the user is dealing with an overflowing inbox, the attacker purchases items in their name, and attempts to reset passwords or transfer domains.

Spear-phishing is still one of the more successful avenues of attack. For attackers that do not have any vulnerabilities they can use immediately from the outside, attacking users is the next step. In a spear-phishing attack, the attacker crafts a legitimately looking email and sends it off to the victim(s). The mail often contains malicious attachments or links or tries to trick the victim into doing some action, like sending money. Spear-phishing emails are often spoofed. That means pretending to be from someone else than they genuinely are. Generally, the better the story of a spear-phishing email, the higher the success rate. Sophisticated attackers collect considerable amounts of openly available information and leverage it to create legitimately looking spear-phishing attacks.

To build a list of email addresses, one can scrape information from various websites, web searches, openly available documents, and any other sources that contain useful information. There are already tools that do this; one example is SimplyEmail[5]. The output of this tool provides the email address format of a company or website and a list of valid users [29, p. 41].

Past breaches One of the simplest and most effective ways to get email accounts is to continually monitor and capture past breaches [29, p. 42].

3.10 Unprotected videoconferencing rooms

The 2020 coronavirus pandemic forced people to work from home. As a result, people are using services like Zoom for videoconferencing. On this service, in particular, many users have found themselves Zoombombed: they are either being attended or disrupted by someone who does not belong. Since meetings are assigned a Meeting ID consisting of 9 to 11 digits and are often not password-protected, unwanted parties can join them.

A tool developed by security researchers, called zWarDial, automates the process of finding non-password protected Zoom meetings, its terminal user interface is displayed in figure 3.3. They found a high number of meetings at large corporations to not be password protected. One researcher, Lo, shared the output of one day's worth of zWarDial scanning, which revealed information about nearly 2,400 upcoming or recurring Zoom meetings. That information included the link needed to join each meeting, the date and time of the meeting, the name of the meeting organizer, and any information supplied by the meeting organizer about the topic of the meeting. [32]

[5]https://simplysecurity.github.io/SimplyEmail/

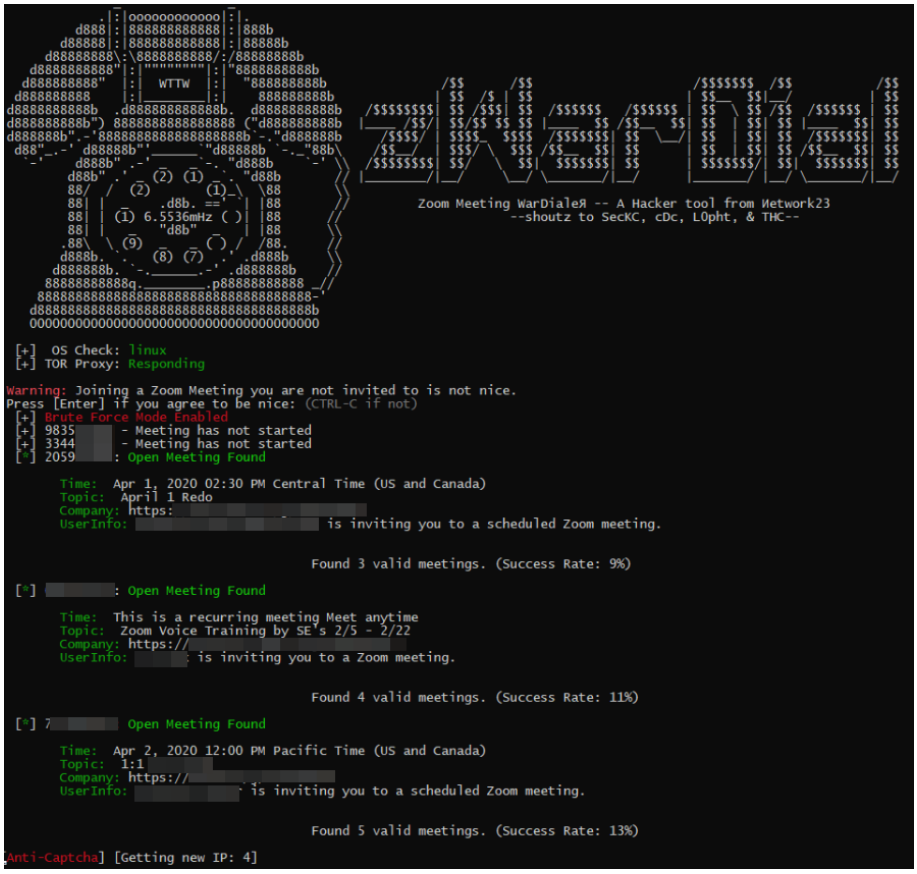

Figure 3.3: A screenshot of zWarDial [32]

Chapter 4

The scanner

4.1 Main focus

4.1.1 Internet-wide non-intrusive vulnerability scanning

The definitions section discussed the differences between active and passive scanning and scanning from the inside vs. outside of a network. Our focus is going to be on passive Internet-wide scanning. This can perhaps more simply be described by table 4.1, and specifically item number four.

Table 4.1: Various scan types

1. Active (noisy) LAN-scan	2. Non-intrusive LAN-scan
3. Active (noisy) Internet-scan	4. Non-intrusive Internet-scan

4.1.2 Making the PoC vulnerability scanner

Making a vulnerability scanner is going to be the primary focus. Some desirable properties for it would be:

1. Be capable of doing large scale scans; be able to scan a potentially long list of hosts.
2. Have the property of non-intrusiveness; Not having any adverse effects on the resources it scans.
3. Be able to detect several vulnerabilities.

The scanner will be focused around scanning web sites. When shaping the scanner, it can be beneficial to think like an attacker. This can make it easier to see what is an attacker interested in, and the potentially quickest and simplest ways to get to it.

A lot of tools exist that are suitable for finding specific vulnerabilities already. There is no need to reinvent the wheel, so any useful existing

programs should be included as separate modules in the scanner. It should also be relatively easy to add or remove modules.

4.2 Preparation

4.2.1 Fetching a list of targets

To begin, one has to get a list of targets to scan. For testing, two datasets will be used. One containing the top-visited no-domains and the Alexa top 1 million most visited websites.

The most popular Norwegian domains We can get this info through Amazon Alexa, but they only provide the top 50 or 100 sites publicly for unregistered visitors. Amazon Alexa provides an extensive list with their Website Analysis Tools through their "Insights" plan, which costs 79 USD / month at the time of writing (September 1. 2019). But they also have a 7-day free trial. It was tried to sign up for this first, but it would only provide the top 500 sites for a country or 1000 globally.

Since many of the top-visited sites in Norway are not no-domains, this proves to be insufficient. We want at least the top 500 visited sites with ".no" domains.

To get all the pages, we need to use the topsites API. This API costs money to use (and has a somewhat esoteric pricing scheme). As the API only returns 100 sites for each request, we need to make ten requests to get the top 1000 sites.

If we test it for just one host:

```
# Get information about the second most visited site
curl -H "x-api-key: jeelechei3ooR4thahzi4Leibee1ute6eebe0oko" \
"https://ats.api.alexa.com/api?Action=Topsites&Count=1\
&CountryCode=NO&ResponseGroup=Country&Start=2"
```

The output is quite verbose and is included in A.3:

To get data for one hundred sites at a time (this is the maximum allowed by the API):

```
curl -H "x-api-key: jeelechei3ooR4thahzi4Leibee1ute6eebe0oko" \
"https://ats.api.alexa.com/api?Action=Topsites&Count=100\
&CountryCode=NO&ResponseGroup=Country&Start=1"
```

The call has to be repeated with `start` query parameter is incremented for each call until the desired number of hosts is reached.

Then the data responses can be collected into a single file:

```
# Having all the files in one folder called responses,
# with an alphabetic ordering of the files starting with "response-"
# to get all the files listed in the correct order:
find responses -type f -name "response-*" | sort
```

```
# And to collect all the files into one:
cat $(find responses -type f -name "response-*" | sort) > result.json

# Count lines
wc -l result.json
> 16480 result.json
```

So this yields a file with over 16000 lines. This file contains a lot of information on the host, but only the hostnames are needed for scanning the targets.

Using the following commands to extract the hostnames:

```
# We can use grep to select only the lines with \acp{URL}
grep "\"DataUrl\":" result.json > grep_result

# And then sed to remove everything except the hostnames
sed -e "s/^\\s*\"DataUrl\"\: \"//" -e "s/\",$//" \
grep_result > hostnames

# Combined into a single "one-liner":
cat $(find responses -type f -name "response-*" | sort) | \
grep "\"DataUrl\":" | \
sed -e "s/^\\s*\"DataUrl\"\: \"//" -e "s/\",$//" > hostnames
```

The Alexa top 1m The second dataset is an already available dataset of the Alexa top 1m. The data is from 2017, so the sites and order might be somewhat outdated compared to what the ranking would have looked like today. This is not a significant concern as we are mainly looking for a list of hosts to scan, and are not so dependant on it being entirely up to date. To fetch the list and format the file appropriately:

```
wget http://s3.amazonaws.com/alexa-static/top-1m.csv.zip
unzip top-1m.csv.zip
cat top-1m.csv | sed 's/^[0-9]*,//' > top-1m
```

4.3 Design

The scanner is designed to do non-intrusive scans on large lists of hosts. It attempts to incorporate known methods for scanning for vulnerabilities in an extensible pipeline. It does not try to "reinvent the wheel" through redeveloping or improving upon already known techniques for vulnerability detection.

The scanner is designed to provide an extensible format where new types of scans are easy to add. Various scans or tasks for the scanner are easy to add in sequence or parallel. This means that the main focus is put on running tasks in parallel, but when required, it can wait for one task to complete before starting one that relies on the result of the previous task. Additionally, the scanner has a configuration file that is loaded upon

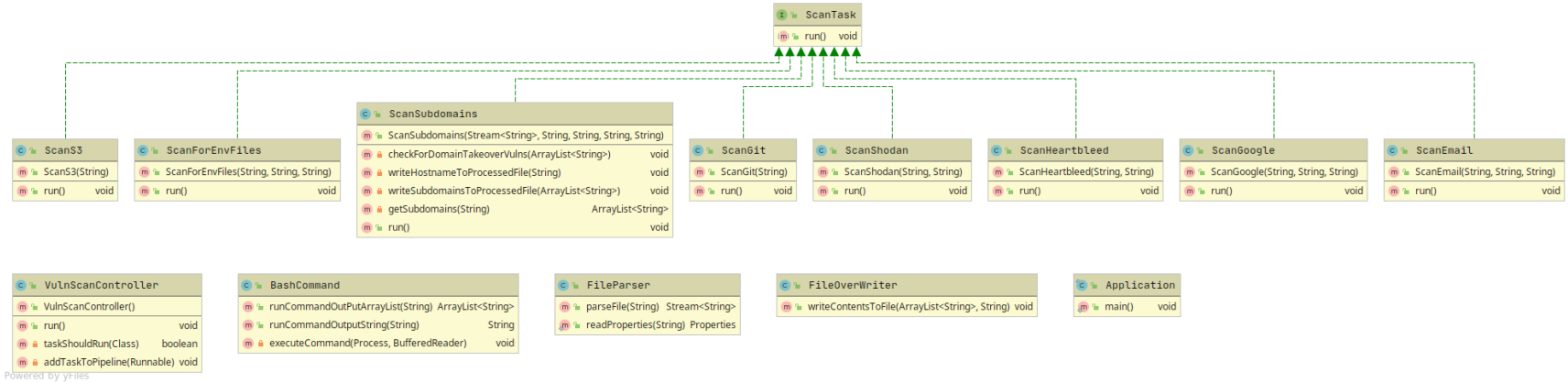

Figure 4.1: UML class diagram of the vulnerability scanner.

runtime, where the user can specify which tasks to run. This makes it easy to tailor the run to the user's needs without the user having to remember any command arguments or flags.

A more detailed and technical description of the scanner and the various scans that are implemented follows in section 4.4.

4.4 Implementation

The scanner is implemented in Java, with the build automation tool Maven for building the application. The choice of language was arbitrary; the scanner could just as well be written in any other language. Some source code from the scanner is included in the appendix, in A.1. The scanner is released under GPL 3.0, and its source code is available on https://github.com/torjuskd/vulnscan.

Each type of scan the scanner performs is called a "scan task," and there is a class for each one. All of these scan tasks are subclasses of the abstract class `ScanTask`. This is illustrated by the UML class diagram in figure 4.1. And figure 4.2 gives a more detailed view of the application.

The `VulnScan`-class contains the logic that defines which scans to run and in what order. This kind of architecture allows for easy parallelization of tasks. We can also run tasks that depend on other tasks sequentially, which is useful for tasks where we need to use previously collected information to run a scan.

Adding a new type of scan is as easy as adding another class that extends the `ScanTask`-class and then adding it to the pipeline in the `VulnScan`-class. To not run a scan type, its flag can be flipped from `true` to `false` in the configuration file (`vulnscan.config`). To remove it permanently from the scanner, one can simply remove the reference to it from the `VulnScan`-class.

The scan tasks that are included in the scanner are:

1. ScanSubdomains
2. ScanForEnvFiles
3. ScanGit
4. ScanS3
5. ScanHeartbleed
6. ScanEmail
7. ScanGoogle
8. ScanShodan

ScanSubdomains (1) enumerates the subdomains for a domain and checks if any of them are vulnerable to takeover attacks. The resulting list of subdomains can be used by other tasks that require knowledge of

subdomains, such as the S3 scan. ScanForEnvFiles (2) scans for `.env`)-files in the root directory of the web root. ScanGit (3) scans for git repositories that are openly hosted in the directory of a website. ScanS3 (4) scans for open Amazon AWS) buckets. ScanHeartbleed (5) performs a scan for the 'Heartbleed' security bug. ScanEmail (6) scans for email addresses associated with a domain. Useful for performing social engineering attacks or information gathering. ScanGoogle (7) scans using some simple google dorks. Or scans for any query that is specified in the configuration file. ScanShodan (8) scans for Norwegian hosts with RDP exposed to the Internet. Custom scans can be configured in the same way as for the google scan.

4.4.1 Build and usage

The application requires Java 11 and Maven to build. Building the application is as simple as:

```
$ mvn clean install
```

With all dependencies installed it can be started from the command line using:

```
$ java -jar vulnscan-1.0.jar
```

The version number depends on what version was built using Maven. What scans to run can be easily configured in the file `vulnscan.config` located in the same directory as the application itself.

The following subsections in this chapter will walk through all the tasks of the scanner.

4.4.2 Subdomain enumeration

The scanner takes a list of domains as input, like the following:

```
$ head -n 5 top-1m-domains
google.com
youtube.com
facebook.com
baidu.com
tmall.com
```

There are several ways of finding subdomains for a target website. The scanner uses the crt.sh database[1] of the Certificate Transparency search engine to look up subdomains. It accomplishes this by using a script from Tlshelpers[2] to look up the subdomains of all the listed domains. This can take a long time when looking up the subdomains for one million websites.

Some dependencies are required to look up subdomains using this technique. If using a Linux distribution with the Advanced Package Tool (APT), download Postgresql and Tlshelpers:

[1] https://crt.sh/

[2] https://github.com/hannob/tlshelpers.git

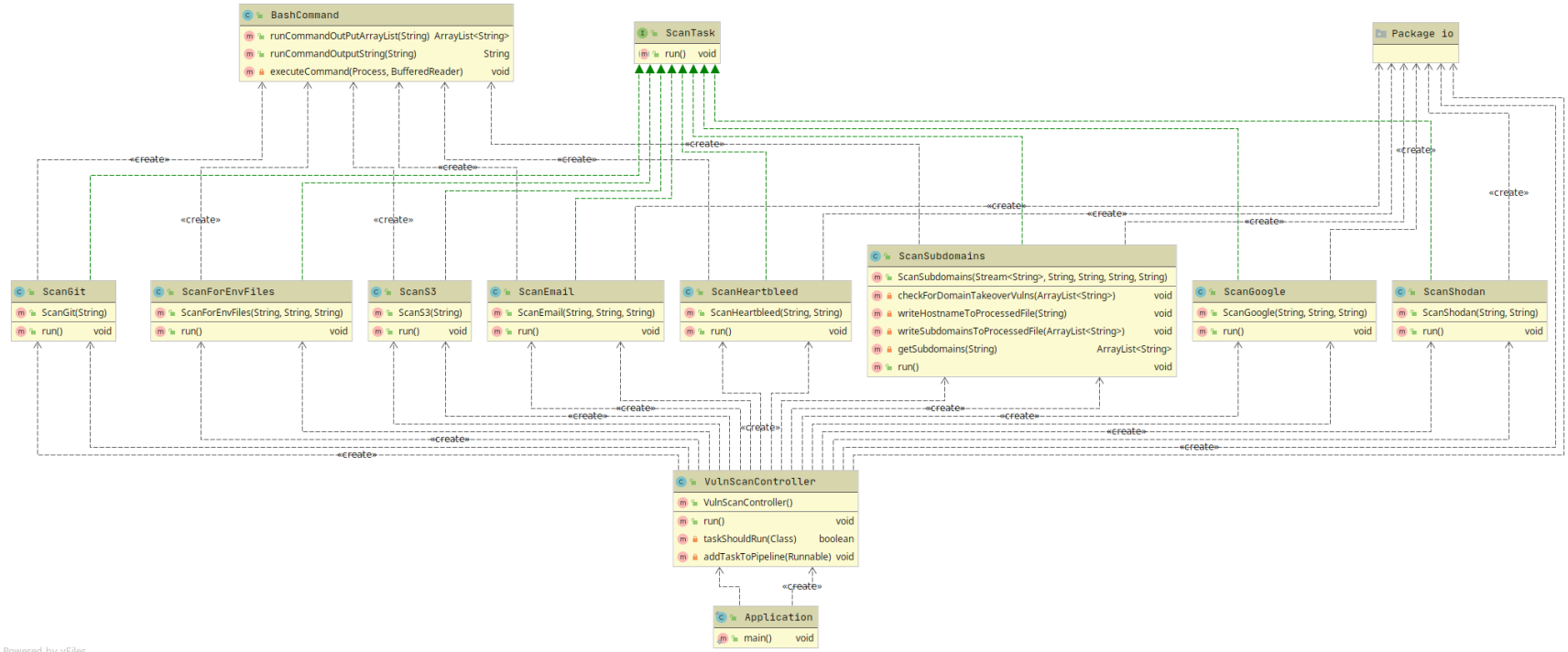

Figure 4.2: Detailed UML class diagram of the vulnerability scanner illustrating program flow through drawn dependencies.

```
sudo apt install postgresql postgresql-contrib
git clone https://github.com/hannob/tlshelpers.git
```

Then to find subdomains for a list of hosts the vulnerability scanner does something equivalent to this:

```
while read in; do bash getsubdomain "$in" >> top-1m-subdomains; done < top-1m-domai

# (Optional:) To monitor results as they appear, we can use tail:
tail -f top-1m-subdomains
```

Looking up the subdomains of the top 1m sites on a standard laptop over a wireless connection took a couple of days when testing.

```
$ wc -l top-1m-subdomains
19015341 top-1m-subdomains
```

We can look up IP addresses from the list of subdomains with dig if we want to.

```
dig +short -f filename
```

4.4.3 Subdomains vulnerable to takeover

With our list of the subdomains of the top 1m sites, we can try to find out if some of them are exploitable. We can look for vulnerable subdomains among the ones we found with subjack[3], a subdomain takeover tool written in Go. It uses a list of "fingerprints;" known responses of hosting and cloud providers that indicate that the domain is available. The list of fingerprints includes Fastly, Github pages, Heroku, Pantheon, Tumblr, Wordpress, and many more. Subjack can also check for subdomains that belong to unregistered domains (NXDOMAIN) and are available for registration. The scanner simply uses the list of previously looked up subdomains and subjack to concurrently check if any of them are vulnerable.

4.4.4 Open cloud storage solutions

Amazon S3 is a very popular cloud storage solution. Using the list of already enumerated subdomains, one can quickly get possible Amazon AWS S3 URLs with grep or ripgrep[4] for even faster results, using a simple regex as an heuristic for selecting domains:

```
rg  '(s3|bucket|aws)' top-1m-subdomains
```

Using the S3 standard subscription, 1000 "LIST requests" costs $0.005 USD[5], in the default region US East (Ohio). This is the request that is used when listing the permissions of a bucket.

[3] https://github.com/haccer/subjack
[4] https://github.com/BurntSushi/ripgrep
[5] https://aws.amazon.com/s3/pricing/

4.4.5 Scanning for vulnerable services

Simple techniques like port scans can detect the presence of vulnerable services.

One downside to scanning a high volume of ports is that it takes a substantial amount of time. For this reason, a module for scanning for only one vulnerability is included in the scanner, to avoid scanning a high volume of ports on each host. This vulnerability is the OpenSSL Heartbleed vulnerability (CVE-2014-0160). A Nmap Scripting Engine (NSE) script, written in Lua by Patrik Karlsson, is used with Nmap to perform the scan (ssl-heartbleed.nse).

The bug itself affects versions 1.0.1 and 1.0.2-beta releases of OpenSSL. The bug permits reading memory from the systems using the vulnerable versions and can compromise encrypted information and the encryption keys themselves.

4.4.6 Environment files

A simple module is added for quickly looking for `.env`-files in the web root. The module can be modified to look for any kind of file inside the web directory of the given host. This module is based on a program called meg[6], made to fetch a lot of files quickly from servers without overloading them.

Most pages will respond with a 404 status code and a body with some text that usually says "Page not found" or similar. Therefore, only responses with the status code 200 are saved. Even these will oftentimes just contain error messages, so the results have to be analyzed/manually verified.

4.4.7 Git directories

The module that looks for git directories hosted in the web root is based on GitTools[7], and simply searches for the presence of git-repositories. URLs for all the found git directories are saved to a file.

4.4.8 Email

The scanner has a module that uses SimplyEmail (mentioned in 3.9) for scanning for email addresses. This tool does a very thorough search, using several search engines, even looking inside documents for email addresses.

4.4.9 Search for internet facing-services

The scanner includes a module that integrates with the Shodan API built on top of an available Shodan java client called jShodan[8]. The query to search for can be configured in the scanner's configuration file.

[6]https://github.com/tomnomnom/meg
[7]https://github.com/internetwache/GitTools
[8]https://github.com/fooock/jshodan

4.4.10 Using search engines to find exposed information

There is an included module that uses the Google search engine to find exposed information. The module includes a simple keyword search that can be configured in the configuration file, here queries can be added to the comma-separated list.

Chapter 5

Results

5.1 Subdomains

The number of subdomains found for the Alexa top 1 million domains was 19 million (the exact number was 19015341).

The number of possibly exploitable subdomains, according to subjack, was about 50 thousand (exact number 51981). This amounted to about 0.27% of all the subdomains.

The subdomains flagged as vulnerable must be manually verified to know exactly how many of them are vulnerable in practice. They ranged from being completely available to using certain providers that make it easier to take over the domain.

Of the top 2000 visited sites in Norway, 606 of them where `.no` domains. The number of subdomains found for these 606 sites was 24083. About 29 of the subdomains where found to be vulnerable with subjack. This equates to 0,12% of the Norwegian subdomains. As the percentage was smaller than for international sites, this can suggest that Norwegian websites are more secure in this respect. But the sample of Norwegian sites is small, so one should be careful drawing any conclusions from it.

5.2 Files exposed in webroot

Several sites were found to have their `.env` files publicly exposed in the webroot directory. Many of them had environment variable names containing the string `PASSWORD`, with some common ones being `MYSQL\_PASSWORD`, `REDIS\_PASSWORD`, `DB\_PASSWORD`, `MAIL\_PASSWORD`.

Some of these passwords are probably for internal services and are not directly exploitable, but many of these `.env`-files were clearly not meant to be public.

A quick way to look for sensitive information in these files is to do a regex search for patterns that target one or more of the sensitive strings. It was hard to come up with any good regexes that didn't require a high degree of manual verification. One that somewhat worked - it did not filter out too many candidates, but had a lot of false positives, was `PASS|password`.

An example of a response with an `.env`-file can be found in the appendix (A.2). The domain name and some other potentially sensitive information (secrets, database names) has been modified to not reveal any sensitive information. From the example file, we can see that some secrets are leaked, in addition to other potentially interesting information for an attacker; the fact that the server seems to be running Laravel in debug mode (`APP_DEBUG=true`), use of Mysql and Redis, the mailprovider and a password and an API key. With this find, an attacker already has a lot of information that can make it easier to attack the application running on this server.

No exposed `.env`-files were found among the top 606 most visited .no-domains.

5.3 Git directories hosted in webroot

In total, 1985 (or about 0.2%) git directories were found to be hosted in the webroot folder of the Alexa top 1 million. Some of these were meant to be hosted publicly, like the Xfce desktop environment source code at xfce.org. And many were absolutely not meant to be public: Several webshops, accounting firms, discussion forums, and many others.

No exposed git directories were found among the top 606 most visited .no-domains.

5.4 Amazon S3 buckets

The scanner is designed to use the already looked up subdomains for scanning for S3 buckets. However, if one were to scan all the subdomains that were looked up, one would have to look up a massive list of subdomains. And only a small percentage of them would be S3 buckets. A simple heuristic to filter out most of the subdomains was therefore used during testing.

With the simple regex mentioned in section 4.4.4; `'(s3|bucket|aws)'`, the total amount of possible S3 subdomains is narrowed down from 19 million to almost eight hundred thousand, or exactly 773 504. With a lookup price of $0.005 per thousand domains, that is a cost of about $4.

The selection heuristic did not end up in a good percentage of found buckets, so the regex `'\.s3-'` was attempted instead. Using this regex as a heuristic, we got a total of 192 possible S3 buckets. 23 of these were found to be real buckets, all with access denied.

It does not seem to be very fruitful to look for open buckets in this manner, at least not for the top most visited sites.

During testing, it seemed more effective to attempt finding buckets by watching certificate transparency logs, using Bucket Stream[1]. Searching in this manner means that the logs are followed as they come in. So the number of buckets that are found depends on the number of events in

[1] https://github.com/eth0izzle/bucket-stream

the certificate stream. This has the disadvantage of being slow, but it is a suitable task for a background job. A handful of test buckets were found with both read- and write privileges to all users during testing. One could set up a script that takes the results from Bucket Stream, then lists and, if possible, dump buckets with s3scanner afterward.

5.5 Emails

Finding email addresses proved to be quite a quite straight forward process. However, it did take a enormous amount of time when scanning with some test domains, so it was not done at scale and the flag for scanning emails was flipped off by default in the final edition of the scanner.

5.6 Searching for hosts with Internet-facing services

Shodan and the Shodan API can be very useful for searching for hosts that have certain properties, such as specific open ports. One such interesting port is 3389, which is most commonly used for RDP (subtracting or adding one to the port number seems to be common as well for the users of this service). Globally, there are 4,591,881 hosts with this port open, and in Norway, there are 5,192. These numbers are high for a service that (as mentioned in chapter 1) should not be exposed to the Internet.

The Shodan API enables searching for and saving information about these hosts. This is information an attacker would use to do credential stuffing or brute force attacks.

Shodan can be used to search for tons of more compelling information and for discovering vulnerable services. The scanner is configured to search for hosts with RDP exposed to the Internet in Norway, but the query can be changed to search Shodan for anything else in the configuration file.

5.7 Searching for exposed information and vulnerable services with google dorks

A simple scan for finding ssh keys is included in the configuration files. This scan was used to find some sites with publicly exposed ssh keys.

5.8 Vulnerable services

The Heartbleed-scan module was never tested on the host lists, because the scan can be loud on the network and targets, and might even be illegal, depending on local laws. [33]

However, a Shodan report from 11-07-2019, claimed that a search for the vulnerability (vuln:cve-2014-0160) returned 91,063 results. With the USA on top of the list with 21,258 hits and China in second place with 8,655. [25]

Chapter 6

Discussion

6.1 Efficiency and evaluation of the scanner

Evaluating the correctness of the results of the scanner is problematic. An automated or manual verification process would be required. An automated process would be just as error-prone as the scanner itself, or in case it was a simple process, it could just be incorporated into the scanner itself, and it would be fault-free in the first place. Manual verification seems then to be the best approach. The problem here is the amount of labor it would take to verify thousands (or millions of results if each subdomain was checked for takeover vulnerabilities).

Neither approach is satisfactory. One is left with using the results of the scanner as a guide or indication, but not as a source of truth.

That being said, the scanner found 0.27% of all subdomains to be vulnerable to takeover, which was close to what was cited (0.5%) in the presentation by David Wind. [76] And the total number of subdomains found was the same (19 million), although the exact number found by It.sec is not quoted in the presentation slides.

Significantly less, 1,985 (or about 0.2%) exposed git directories were found by this scanner, than by It.sec (3,900), but crucially, It.sec scanned all 19 million domains, while the scanner was only used on the Alexa 1 million list. By the percentage (exposed git directories/total domains) was much higher for scanners results using the Alexa 1 million list, 0.2% vs. 0,02%.

The time taken to run the scans was not recorded exactly. This makes it hard to make judgments on efficiency.

Having looked at some ways to detect vulnerabilities, some methods and processes for mitigation is discussed in section 6.2.

6.2 Mitigation

We have so far looked at vulnerabilities and ways to exploit them, now we'll have a look at what can be done to improve the security posture for different entities on the web.

6.2.1 Regulatory, WWW-wide technical and compliance solutions

Proposed solutions to the inherent security risks on the Web vary. Large security companies like McAfee design governance and compliance suites to meet post-9/11 regulations[39], and some, like Finjan have recommended active real-time inspection of programming code and all content regardless of its source. [17] Some have argued that for enterprises to see web security as a business opportunity rather than a cost center, [50] while others call for "ubiquitous, always-on digital rights management" enforced in the infrastructure to replace the hundreds of companies that secure data and networks.[15] Jonathan Zittrain has said users sharing responsibility for computing safety is far preferable to locking down the Internet. [36]

Regulations like the EU General Data Protection Regulation (GDPR) can push service providers to limit the use and collection of user information. It can also make users more aware and in control of what information is being shared and stored, because the user has to accept specific uses of their data explicitly.

6.2.2 Precautions for organizations and companies

Security awareness in organizations Many organizations view information security as a net cost center, as a sort of "necessary evil" that is there only to prevent bad things from happening. The primary objective is to keep costs down as much as possible while still complying with legislation. This is arguably a dangerously myopic view of information security, and lends itself to pitfalls like complying with policy, while not considering numerous vulnerabilities that the policy is not concerned with.

The alternative viewpoint to take is to see information security as an area of opportunity and possibly excellence. Where current practices can be made more secure and continuously improved. The lack of security we see today on the Internet cannot be blamed solely on lack of knowledge and complex technologies; information on how to make more secure solutions is out on the Internet, freely available for anyone who is willing to learn about it. No, the main reason we still see insecurity today is that it just is not that high of a priority many organizations. By taking a proactive stance towards security, money can also be saved in the process. Issuing a security bulletin costs Microsoft at least USD 100 000$ [48, p. 34],and the cost to customers is immeasurable. Data breaches can have grave consequences, both personal and financial. With a proactive approach to security, some of these events could likely have been prevented altogether.

Once a security mindset is in place in an organization, more secure practices will follow. It is hard or impossible to go about it the other way around. When security practices incorporated from the ground up they do not have to be bolted on later.

Sophos suggests developing robust policies and systems to support human decision making and that deals with human failures will be

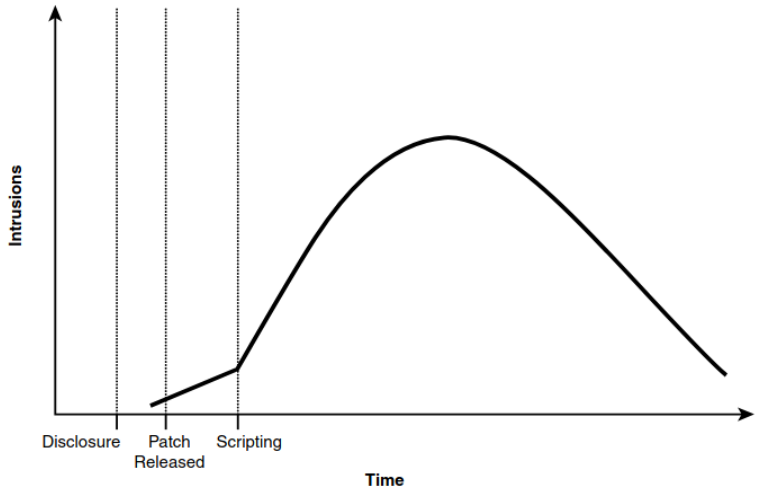

Figure 6.1: The development in number of intrusions permitted (in average) by a security bug during its lifetime [71, p. 17]

required. [61]

Process The old practice of "penetrate and patch" falls short in the current climate of rapid development and integration. [71, p. 15-17] Companies are moving fast and continuously add new servers, cloud solutions, and applications to their portfolio. This requires a continuous reassessment of risks and checking for old and new vulnerabilities. One approach to the problem that is gaining in popularity is called DevSecOps. At its core, DevSecOps is about incorporating security practices into Development and Operations (DevOps). [13] This can make it easier to incorporate security from the start and throughout the development cycle, and take advantage of DevOps principles like test automation. This process can help support suggested countermeasures to attacks, such as repeated security hardening, software updating and patching, and operation of systems to monitor the security of all components in the application stack. [64, p. 3-6]

Web application security The OWASP Testing Framework [48] can be followed to assess the security of a web application. It is a complete framework that can be used to build test programs and qualify test processes. Made to tackle the questions of what, why, when and how to test it takes a holistic approach to security testing. In the framework, web application security testing is broken down into the following major activities:

1. introduction and objectives
2. information gathering
3. identity management testing
4. authentication testing

5. authorization testing
6. session management testing
7. testing for error handling
8. testing for weak cryptography
9. business logic testing

The framework recommends regularly running vulnerability scanners, testing and securing web applications from known risks, and securing communications by enabling Hypertext Transfer Protocol Secure (HTTPS) and HTTP Strict Transport Security (HSTS), among countless of other steps to harden security.

6.2.3 Security experts

There can be many incentives for contributing to the security or "insecurity" of the web for security researchers, bounty hunters and state actors. Vulnerability markets and bug bounty programs can provide economic incentives for disclosure. A public disclosure of a vulnerability also often acts as a status symbol by many security experts. [64, p. 3] Disclosure of a vulnerability to the affected product owner can help initiate the process of patching a vulnerability, while selling vulnerabilities on the black market arguably contributes towards undermining the security of the vulnerable entities.

6.2.4 Security for the end user

From a user perspective, there are things one can do to reduce one's attack surface. One common and effective measure is to disable JavaScript from running in the web browser. The downside to this is that it often breaks much, if not all, of the functionality of many websites and cause others not to display correctly. Other common ways to reduce footprint and increase security are ad-blocking, more strict session handling for example through containerized sessions, blocking specific trackers or rapid deletion of cookies, multi-factor authentication, always using https, disallowing TLS 1.0 and 1.1 and the age-old advice of strong password generation and rapid patching.

The number of security precautions that can be taken as a user is innumerable, and it is often tough or even impossible to assess the effectiveness of the methods that are put to use. In any case, possible security measures can be more easily evaluated when they are dictated by a threat model.

Threat model A threat model is the end result of threat modeling. Threat modeling is a process where potential threats, such as vulnerabilities or the lack of appropriate safeguards, can be identified and enumerated. This

allows mitigations to be prioritized. A threat modeling serves to provide defenders with a systematic analysis of what controls and defenses need to be included, given the system, the probable attacker's profile, the most likely attack vectors, and the most valuable assets for an attacker. Threat modeling answers questions like "Where is an attack most likely to occur?", "Which threats are the most severe?" and "What actions can I take to defend against these threats?". [60, p. 3-28]

operations security (OPSEC) A threat model can be used as part of a more robust approach to security, known as the OPSEC model. Its five key points are:

- Identify the information you need to protect
- Analyze the threats
- Analyze your vulnerabilities
- Assess the risk
- Apply countermeasures

The OPSEC model was developed during the Vietnam War era as a part of military strategies to protect critical information, analyzing vulnerabilities and threats, assessing risks, and applying proper countermeasures. The 5 steps OPSEC model have been functional for the US Army's operational security and used by other NATO members as well. When threats have spread widely in 21 st Century OPSEC began to be used and applied in a more general context in the security industry. Today, in the cybersecurity community, this model serves practically for the protection of critical data and information. [3]

The OPSEC model is an iterative model where information can be attained at any stage and can be applied to the next. For example, after applying countermeasures, one might discover that there were threats that one was not aware of. One can then go back to analyzing threats again, but this time with more insight.

The bottom line is that there is no single solution to all the security and privacy issues on the Web in sight. Stacking several mitigating factors, using layered protection mechanisms seems to be the most sensible approach.

6.3 Internet security in the future

6.3.1 Automation-assisted attacks

According to Sophos, Attackers are using a combination of automated tools and humans to evade security more effectively controls than ever before. In 2019 the MTR Operations Team observed attackers automating the earlier stages of their attacks to gain access and control of the targeted environment and then shift to utilizing patient, methodical means to identify and complete their objective. [61, p. 23]

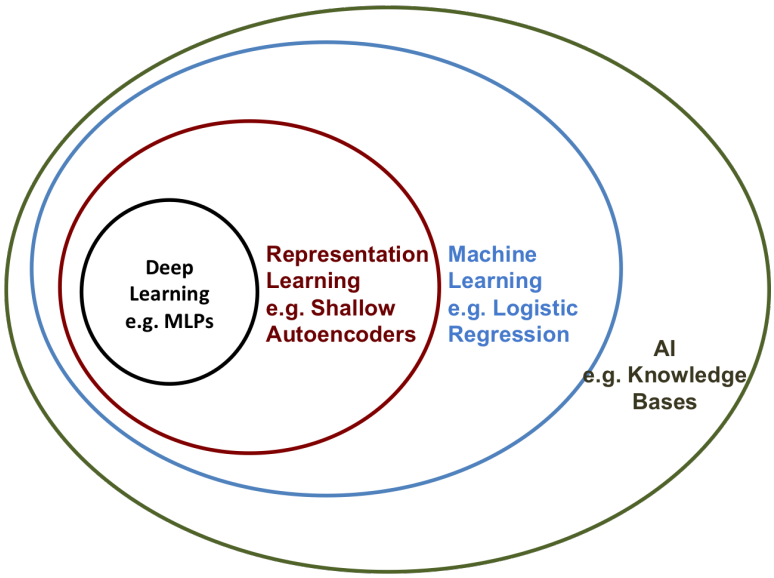

Figure 6.2: The relations between various AI-concepts. [24]

6.3.2 Cloud security

The rapid adaption of cloud solutions has brought with it its own challenges. Since cloud solutions are flexible, they make it easier to make configuration changes on a whim, which also makes it easier to make mistakes. With a single misstep, a system administrator might open the customer database to the Internet.

A recent example of this is the spread of Magecart - a malicious piece of JavaScript that steals credentials or credit card information. The script is embedded into the shopping cart or checkout page of online retailers, where it redirects information to the attacker. This hit big online retailers: Ticketmaster, Cathay Pacific Airways, Newegg, and British Airways. The attacks were only discovered after customer complaints started to come in. [61, p. 20-21]

Visibility into the consequences of configuration changes, and the ability to monitor systems for malicious or suspicious activity has proved the most effective for defending against attacks on cloud solutions. [61, p. 20]

6.3.3 Artificial intelligence (AI)

The use of Machine learning technologies is rapidly increasing. It can be applied for various purposes, both for good and harm. Attackers are already starting to target the machine learning engines. [61, p. 25]

We are not far from a world where machine learning systems are attacking each other and defending themselves. Generative models are models that can be used to generate content to looks human-made, but can be crafted by a machine from scratch. They can be used to generate images, news articles, and videos (often referred to as deep fakes). Such

systems might be used to spread misinformation and assist in phishing attacks in the future. To defend from these attacks, one might employ a machine learning systems and humans to identify fakes, by spotting flaws using methods like pattern recognition and metadata analysis.

Chapter 7

Conclusion and future work

7.1 Conclusion

7.1.1 Vulnerabilities

In chapter 2, it was outlined that many vulnerabilities could be detected non-intrusively. From OWASP top 10, there were: Sensitive data exposure, security misconfiguration, and using components with known vulnerabilities. Vulnerabilities not specific to web applications followed: Unregistered subdomains, exposed credentials, exposed source code, open cloud storage solutions, CORS misconfiguration, vulnerable tech-stacks, and inadequate protection mechanisms against social engineering attacks.

7.1.2 Methods

There are a variety of methods for non-intrusive vulnerability scanning. The methods can be used both by those wishing to attack systems, and those wishing to defend themselves from attacks.

Some of the methods mentioned in chapter 3 are port scanning, scanning for web application vulnerabilities, spidering, using scan databases, search engines, and cloud scanners, various methods for subdomain discovery and takeover, techniques for finding publicly exposed information, gathering public information for use in other attacks (social engineering), and detection of unprotected services. There are probably countless other methods for non-intrusive vulnerability detection, and many are likely yet to be discovered.

7.1.3 The scanner

An important goal was also to create a functional scanner. The goal for making the scanner was for it to have the following capabilities:

1. Be capable of doing large scale scans; Capable of scanning a potentially long list of hosts.

2. Have the property of non-intrusiveness; Not having any adverse effects on the resources it scans.

3. Be able to detect several vulnerabilities.

The PoC scanner ended up attaining all of the mentioned capabilities: The scanner takes a lengthy list of hostnames as input and effectively parallelizes the scan tasks.

Modules were added to the scanner for the following tasks: subdomain enumeration and takeover, scan for open cloud storage solutions, scan for vulnerable services (Heartbleed), a scan for environment files, a scan for detecting exposed Git directories, a module to gather email addresses, a module to search for Internet-facing services using Shodan, and a simple client for searching for exposed information using the Google search engine.

7.2 Future work

Time and financial constraints Some of the scan types were not attempted on the whole dataset of Alexa top 1 million. They were either very time-consuming or could be costly. A prosperous actor with lots of time on their hands could afford to run all the scans on the entire Alexa list.

Warning/reporting to affected parties As email addresses also can be found for the domains, one could also implement an automatic warning mechanism and send an email to addresses like admin@domain.tld or webmaster@domain.tld. But the criteria for what to warn about would have to be well thought out; one organization may want to host their source code publicly while others do not. It would be hard to distinguish between such organizations automatically, so some sort of manual confirmation would probably still be needed.

Extend scanner with Internet resources This scanner focused on using local tools for finding vulnerabilities. More modules that take advantage of data from scan databases and search engines could be added. The scanner could be extended with additional features to become even more powerful and tailored to whichever needs one has.

Assessing the validity of the results Assessing the validity of the results and the effectiveness of the scanner turned out to be a daunting task. Techniques for assessing the validity of results could be incorporated to improve confidence in results, or manual verification can be applied to evaluate validity of the scan results.

Appendices

.1 Acronyms

ACL access control list 22

AI Artificial intelligence iv

API application programming interface 8

AWS Amazon Web Services 7

CD continuous deployment 17

CI continuous integration 17

CLI command line interface 20

CORS cross-origin resource sharing 7

DNS domain name system 6

DevOps Development and Operations 43

FQDN full qualified domain name 22

GDPR General Data Protection Regulation 42

GPL GNU General Public License 17

HSTS HTTP Strict Transport Security 44

HTML Hypertext Markup Language 10

HTTPS Hypertext Transfer Protocol Secure 44

IDS intrusion detection system 17

IPS intrusion prevention system 17

LAN local area network 5

NSE Nmap Scripting Engine 35

OPSEC operations security 45

OSINT open-source intelligence 24

OS operating system 15

OWASP Open Web Application Security Project ix

PII personally identifiable information 9

PoC proof of concept i

RCE remote code execution 2

RDP Remote Desktop Protocol 2

S3 Simple Storage Service iv

SSL Secure Socket Layer 12

TLD top level domain 11

TLS Transport Layer Security 1

UML Unified Modeling Language vii

URI uniform resource identifier 1

URL Uniform Resource Locator 18

UiO the university of Oslo 2

VCS version control system 12

VPN virtual private network 2

WWW World Wide Web iv

XML Extensible Markup Language 9

XSS Cross-Site Scripting 1

XXE Xml External Entities 9

ZAP Zed Attack Proxy 16

w3af Web Application Attack and Audit Framework 17

Appendix A

Appendix

A.1 Scanner source code

Some source code from the scanner is included here. The scanner is released under GPL-3.0, and the source code is available on https://github.com/torjuskd/vulnscan.

A.1.1 The controller class

```
package no.uio.ifi.vulnscan;

import no.uio.ifi.vulnscan.tasks.*;
import no.uio.ifi.vulnscan.util.io.FileParser;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.util.*;
import java.util.concurrent.CompletableFuture;

/**
 * The "controller"-class of non-intrusive large scale vulnerability scanner
 * Handles properties, constants and the execution of all the tasks of the scanner
 */
public class VulnScanController {
    private final Logger log = LoggerFactory.getLogger(VulnScanController.class);
    private final String processedHostsFilename = "processed_hosts";
    private final String processedSubdomainsFilename = "processed_subdomains";
    private final String subdomainsSubjackResultsFile = "subdomain_subjack_results";
    private final String subdomainsTempFileName = "subdomains_temp";
    private final String heartbleedFilename = "heartbleed_script_output";
    private final String megPathsFilename = "meg_paths";
    private final String emailResultFolderName = "email_output";
    private final String propertiesFileName = "vulnscan.config";
    private final String megHostnamesWithProtocolFilename = "meg_hostnames_with_protocol";
    private final String simplyEmailDir;
    private final String hostsToScan;
    private final String googleApiKey;
    private final String googleSearchQuery;
    private final String googleSearchEngine;
    private final String shodanApiKey;
    private final String shodanSearchQuery;
    private final Properties properties;
    private final List<CompletableFuture<Void>> scanTasks;

    /**
     * Sets the following
     * filename        the path of the file containing the hosts you want to scan
     *
     */
    public VulnScanController() {
        scanTasks = new ArrayList<>();

        properties = FileParser.readProperties(propertiesFileName);

        hostsToScan = properties.getProperty("HOST_LIST_TO_SCAN_FILENAME");
        simplyEmailDir = properties.getProperty("SIMPLY_EMAIL_DIR");
        googleApiKey = properties.getProperty("GOOGLE_API_KEY");
        googleSearchQuery = properties.getProperty("GOOGLE_SEARCH_QUERY");
        googleSearchEngine = properties.getProperty("GOOGLE_SEARCH_ENGINE");
```

```
        shodanApiKey = properties.getProperty("SHODAN_API_KEY");
        shodanSearchQuery = properties.getProperty("SHODAN_SEARCH_QUERY");
    }

    /**
     * Run the various tasks of the scanner
     * <p>
     * This is where scan tasks are executed. Add new scan tasks here if wanted.
     * New scan tasks have to implement the ScanTask-interface, see {@link ScanTask}.
     * <p>
     * Tasks can run in parallel or sequentially. This is accomplished
     * through the use of CompletableFutures.
     */
    public void run() {
        if (taskShouldRun(ScanForEnvFiles.class)) {
            addTaskToPipeline(new ScanForEnvFiles(megPathsFilename,
                                                  hostsToScan,
                                                  megHostnamesWithProtocolFilename));
        }

        if (taskShouldRun(ScanGit.class)) {
            addTaskToPipeline(new ScanGit(hostsToScan));
        }

        /*
         * Only runs s3 scan after subdomain enumeration.
         */
        if (taskShouldRun(ScanSubdomains.class) && taskShouldRun(ScanS3.class)) {
            scanTasks.add(CompletableFuture.runAsync(
                    new ScanSubdomains(new FileParser().parseFile(hostsToScan),
                                       subdomainsTempFileName,
                                       subdomainsSubjackResultsFile,
                                       processedHostsFilename,
                                       processedSubdomainsFilename))
                            .thenRun(new ScanS3(processedSubdomainsFilename)));
        } else if (taskShouldRun(ScanSubdomains.class)) {
            addTaskToPipeline(new ScanSubdomains(new FileParser().parseFile(hostsToScan),
                                                 subdomainsTempFileName,
                                                 subdomainsSubjackResultsFile,
                                                 processedHostsFilename,
                                                 processedSubdomainsFilename));
        }

        if (taskShouldRun(ScanHeartbleed.class)) {
            addTaskToPipeline(new ScanHeartbleed(hostsToScan,
                                                 heartbleedFilename));
        }

        if (taskShouldRun(ScanEmail.class)) {
            addTaskToPipeline(new ScanEmail(hostsToScan,
                                            emailResultFolderName,
                                            simplyEmailDir));
        }

        if (taskShouldRun(ScanShodan.class)) {
            addTaskToPipeline(new ScanShodan(shodanApiKey,
                                             shodanSearchQuery));
        }

        if (taskShouldRun(ScanGoogle.class)) {
            addTaskToPipeline(new ScanGoogle(googleApiKey,
                                             googleSearchQuery,
                                             googleSearchEngine));
        }

        log.info("Scan starting, processing domains.");
        scanTasks.forEach(CompletableFuture::join);
        log.info("All hosts processed, Finished.");
    }

    /**
     * Checks if a task should be run, based on parameters in the config-file.
     * classname=true to run a task or
     * classname=false to not run a task
     *
     * @param clazz the class of the scan-task
     * @return true if task should run
     */
    private boolean taskShouldRun(final Class clazz) {
        return Boolean.parseBoolean(properties.getProperty(clazz.getSimpleName()));
    }

    /**
     * Adds task to pipeline.
     * All the tasks that are added this way will be executed in parallel.
     *
     * @param runnable task to run
     */
    private void addTaskToPipeline(final Runnable runnable) {
        scanTasks.add(CompletableFuture.runAsync(runnable));
    }
}
```

A.1.2 Example of a scan task: Env file scan

```
package no.uio.ifi.vulnscan.tasks;

import no.uio.ifi.vulnscan.util.BashCommand;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class ScanForEnvFiles implements ScanTask {
    private static final Logger log = LoggerFactory.getLogger(ScanForEnvFiles.class);
    final String megPathsFilename;
    private final String actualHostsToScanFileName;
    private final String megHostnamesWithProtocolFilename;

    public ScanForEnvFiles(final String megPathsFilename,
                           final String actualHostsToScanFileName,
                           final String megHostnamesWithProtocolFilename) {
        this.megPathsFilename = megPathsFilename;
        this.actualHostsToScanFileName = actualHostsToScanFileName;
        this.megHostnamesWithProtocolFilename = megHostnamesWithProtocolFilename;
    }

    @Override
    public void run() {

        log.info("running meg to look for files in webroot");

        new BashCommand().runCommandOutputString("sed -e 's/^/https:\\/\\//' " +
                                                 actualHostsToScanFileName + " > " +
                                                 megHostnamesWithProtocolFilename);
        // create paths file with /.env if it does not exist
        new BashCommand().runCommandOutputString(
                "[ ! -f " + megPathsFilename + " ] && echo \"/.env\" >> " + megPathsFilename);

        new BashCommand().runCommandOutputString(
                "meg --savestatus 200 " + megPathsFilename + " " +
                megHostnamesWithProtocolFilename);

        log.info("meg finished");
    }
}
```

A.2 Exposed .env response example

```
http://example.com/.env

> GET /.env HTTP/1.1
> Host: example.com
> User-Agent: Mozilla/5.0 (compatible; meg/0.2; ...

< HTTP/1.1 200 OK
< Etag: "5c61c1fb-2fd"
< Accept-Ranges: bytes
< Server: nginx/1.16.1
< Date: Tue, 24 Sep 2019 16:32:30 GMT
< Content-Type: application/octet-stream
< Content-Length: 765
< Last-Modified: Mon, 11 Feb 2019 18:42:03 GMT

APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:cGFzc3dvcmRwYXNzd29yZHBhc3N3b3JkCg==
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack
```

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=mydb
DB_USERNAME=mydb
DB_PASSWORD=secret

BROADCAST_DRIVER=log
CACHE_DRIVER=file
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_DRIVER=smtp
MAIL_HOST=smtp.mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null

PUSHER_APP_ID=
PUSHER_APP_KEY=
PUSHER_APP_SECRET=
PUSHER_APP_CLUSTER=mt1

MIX_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
MIX_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"

REDIRECT_HTTPS=1
```

A.3 Amazon Alexa top sites API JSON


```
{
    "Ats": {
        "OperationRequest": {
            "RequestId": "868a0fa1-c3c0-46f0-b10c-c9419518b9dd"
        },
        "Results": {
            "Result": {
                "Alexa": {
                    "Request": {
                        "Arguments": {
                            "Argument": [
```

```
                        {
                            "Name": "countrycode",
                            "Value": "NO"
                        },
                        {
                            "Name": "start",
                            "Value": "2"
                        },
                        {
                            "Name": "count",
                            "Value": "1"
                        },
                        {
                            "Name": "responsegroup",
                            "Value": "Country"
                        }
                    ]
                }
            },
            "TopSites": {
                "Country": {
                    "CountryName": "Norway",
                    "CountryCode": "NO",
                    "TotalSites": "2521",
                    "Sites": {
                        "Site": {
                            "DataUrl": "youtube.com",
                            "Country": {
                                "Rank": "2",
                                "Reach": {
                                    "PerMillion": "530000"
                                },
                                "PageViews": {
                                    "PerMillion": "66700",
                                    "PerUser": "3.74"
                                }
                            },
                            "Global": {
                                "Rank": "2"
                            }
                        }
                    }
                }
            }
        },
        "ResponseStatus": {
            "StatusCode": "200"
```

```
                }
            }
        }
    }
```

Bibliography

[1] *2012 Global Losses From Phishing Estimated At $1.5 Bn - Firstbiz*. [Online; accessed 5. Apr. 2020]. Feb. 2013. URL: https://web.archive.org/web/20141221122958/http://firstbiz.firstpost.com/biztech/2012-global-losses-from-phishing-estimated-at-1-5-bn-16850.html.

[2] *A Guide To Subdomain Takeovers*. [Online; accessed 11. Apr. 2020]. Apr. 2020. URL: https://www.hackerone.com/blog/Guide-Subdomain-Takeovers.

[3] M. Kubilay Akman. 'OPSEC Model and Applications'. In: *Security Dimensions. International and National Studies* 25 (2018), pp. 60–81. ISSN: 2353-7000. URL: https://www.ceeol.com/search/article-detail?id=723776#tableOfContents.

[4] *autoSubTakeover*. URL: https://github.com/JordyZomer/autoSubTakeover. (visited on 19/05/2019).

[5] L. Baird. *Discover scripts*. URL: https://github.com/leebaird/discover (visited on 15/05/2019).

[6] Genevieve Bartlett, John Heidemann and Christos Papadopoulos. 'Understanding Passive and Active Service Discovery'. In: *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*. IMC '07. San Diego, California, USA: ACM, 2007, pp. 57–70. ISBN: 978-1-59593-908-1. DOI: 10.1145/1298306.1298314. URL: http://doi.acm.org/10.1145/1298306.1298314.

[7] *BBC NEWS | Technology | Google searches web's dark side*. [Online; accessed 29. Mar. 2020]. Mar. 2020. URL: http://news.bbc.co.uk/2/hi/technology/6645895.stm.

[8] Yuval Ben-Itzhak. 'Infosecurity 2008 - New defence strategy in battle against e-crime'. In: *ComputerWeekly.com* (Apr. 2008). URL: https://www.computerweekly.com/opinion/Infosecurity-2008-New-defence-strategy-in-battle-against-e-crime.

[9] Scott Berinato. 'Software Vulnerability Disclosure: The Chilling Effect'. In: *CSO Online* (Jan. 2007). URL: https://www.csoonline.com/article/2121727/software-vulnerability-disclosure--the-chilling-effect.html?page=2.

[10] Hanno Böck. 'Subdomain Takeover: Microsoft loses control over Windows Tiles - Golem.de'. In: *Golem.de* (Apr. 2019). URL: https://www.golem.de/news/subdomain-takeover-microsoft-loses-control-over-windows-tiles-1904-140717.html.

[11] Bryan Burns et al. *Security Power Tools*. Sebastopol, CA: O'Reilly, 2007. URL: https://cds.cern.ch/record/1095693.

[12] *Burp Suite Scanner*. URL: https://portswigger.net/burp (visited on 14/05/2019).

[13] K. Carter. 'Francois Raynaud on DevSecOps'. In: *IEEE Software* 34.5 (2017), pp. 93–96.

[14] *Censys - mission*. URL: https://censys.io/about (visited on 13/05/2019).

[15] Thomas Claburn. *RSA's Coviello Predicts Security Consolidation – Security – InformationWeek*. [Online; accessed 5. Apr. 2020]. Feb. 2007. URL: https://web.archive.org/web/20090207091418/http://www.informationweek.com/news/security/showArticle.jhtml?articleID=197003826.

[16] *CORS Misconfigurations Explained*. [Online; accessed 9. Mar. 2020]. Apr. 2018. URL: https://blog.detectify.com/2018/04/26/cors-misconfigurations-explained.

[17] *CWE - Vulnerability Type Distributions in CVE*. [Online; accessed 29. Mar. 2020]. Mar. 2020. URL: https://cwe.mitre.org/documents/vuln-trends/index.html.

[18] *Domain Dossier*. URL: https://centralops.net/co/domaindossier.aspx (visited on 13/05/2019).

[19] B. Eshete, A. Villafiorita and K. Weldemariam. 'Early Detection of Security Misconfiguration Vulnerabilities in Web Applications'. In: *2011 Sixth International Conference on Availability, Reliability and Security*. 2011, pp. 169–174.

[20] *Firefox 74.0, See All New Features, Updates and Fixes*. [Online; accessed 6. Apr. 2020]. Apr. 2020. URL: https://www.mozilla.org/en-US/firefox/74.0/releasenotes.

[21] Seth Fogie et al. *XSS Attacks: Cross Site Scripting Exploits and Defense*. Syngress, May 2007. ISBN: 978-159749154-9. URL: https://www.amazon.com/XSS-Attacks-Scripting-Exploits-Defense/dp/1597491543.

[22] Tillson Galloway. *How I made 10K in bug bounties from GitHub secret leaks*. May 2020. URL: https://tillsongalloway.com/finding-sensitive-information-on-github/index.html.

[23] Béla Genge and Călin Enăchescu. 'ShoVAT: Shodan-based vulnerability assessment tool for Internet-facing services'. In: *Security and Communication Networks* 9.15 (2016), pp. 2696–2714. DOI: 10.1002/sec.1262. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sec.1262. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/sec.1262.

[24] Ian Goodfellow, Yoshua Bengio and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[25] *Heartbleed Report - Shodan*. [Online; accessed 20. Mar. 2020]. Mar. 2020. URL: https://www.shodan.io/report/0Wew7Zq7.

[26] *HostileSubBruteforcer*. URL: https://github.com/nahamsec/HostileSubBruteforcer. (visited on 19/05/2019).

[27] Darrel Ince. *vulnerability scanning*. 2019. URL: https://www.oxfordreference.com/view/10.1093/acref/9780191884276.001.0001/acref-9780191884276-e-4402.

[28] P. Kim. *sslScrape,* URL: https://github.com/cheetz/sslScrape (visited on 13/05/2019).

[29] P. Kim. *The Hacker Playbook 3: Practical Guide To Penetration Testing*. Kindle Edition: Secure Planet, 2018.

[30] Brian Krebs. *The Scrap Value of a Hacked PC, Revisited*. [Online; accessed 3. Jun. 2020]. June 2020. URL: https://krebsonsecurity.com/2012/10/the-scrap-value-of-a-hacked-pc-revisited.

[31] Brian Krebs. *The Value of a Hacked Email Account*. [Online; accessed 12. Apr. 2020]. Apr. 2020. URL: https://krebsonsecurity.com/2013/06/the-value-of-a-hacked-email-account.

[32] Brian Krebs. *'War Dialing' Tool Exposes Zoom's Password Problems*. [Online; accessed 2. Apr. 2020]. Apr. 2020. URL: https://krebsonsecurity.com/2020/04/war-dialing-tool-exposes-zooms-password-problems/#more-51159.

[33] *Legal Issues - Nmap Network Scanning*. [Online; accessed 27. Apr 2020]. Apr. 2020. URL: https://nmap.org/book/legal-issues.html.

[34] Kristina Libby. 'How Safe Is Zoom?' In: *Popular Mechanics* (Apr. 2020). URL: https://www.popularmechanics.com/technology/security/a31982009/is-zoom-safe.

[35] J. Long. *Google Hacking Database (GHDB) in 2004*. 10th May 2004. URL: https://web.archive.org/web/ (visited on 09/05/2019).

[36] Carolyn Duffy Marsan. *How the iPhone is killing the 'Net - Network World*. [Online; accessed 5. Apr. 2020]. Apr. 2008. URL: https://web.archive.org/web/20080414043829/http://www.networkworld.com/news/2008/040908-zittrain.html.

[37] *MassDNS 0.3*. URL: https://github.com/blechschmidt/massdns (visited on 19/05/2019).

[38] John Matherly. *Trends in Internet Exposure*. [Online; accessed 6. Apr. 2020]. Mar. 2020. URL: https://blog.shodan.io/trends-in-internet-exposure.

[39] *McAfee Governance, Risk and Compliance Business Unit*. [Online; accessed 5. Apr. 2020]. Apr. 2008. URL: https://www.eweek.com/security/mcafee-governance-risk-and-compliance-business-unit.

[40] Michael Meli, Mattew R. McNiece and Bradley Reaves. 'How Bad Can It Git? Characterizing Secret Leakage in Public GitHub Repositories – NDSS Symposium'. In: *The Network and Distributed System Security Symposium (NDSS)* (Feb. 2019). URL: https://www.ndss-symposium.org/ndss-paper/how-bad-can-it-git-characterizing-secret-leakage-in-public-github-repositories.

[41] *Nessus 3 documentation*. URL: http://www.nessus.org/documentation/index.php?doc=nessus3 (visited on 22/04/2019).

[42] C. Nickerson et al. *The penetration testing execution standard*, URL: http://www.pentest-standard.org (visited on 16/03/2019).

[43] *Nikto*. Mar. 2020. URL: https://tools.kali.org/information-gathering/nikto (visited on 18/03/2019).

[44] *Nmap*. [Online; accessed 10. Apr. 2020]. Mar. 2020. URL: https://nmap.org.

[45] OWASP. *Category: OWASP Top Ten Project*. URL: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project (visited on 23/02/2019).

[46] OWASP. *OWASP Top 10-2017: The Ten Most Critical Web Application Security Risks*. URL: https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf (visited on 23/02/2019).

[47] O.W.A.S.P. *Testing: Spidering and googling*, URL: https://www.owasp.org/index.php/Testing (visited on 14/05/2019).

[48] *OWASP Web Security Testing Guide v4.1*. [Online; accessed 12. Apr. 2020]. Apr. 2020. URL: https://owasp.org/www-project-web-security-testing-guide.

[49] O.W.A.S.P. Zed. *Attack Proxy Project*. URL: https://www.owasp.org/index.php/OWASP%5C_Zed%5C_Attack%5C_Proxy%5C_Project. (visited on 14/05/2019).

[50] Rob Preston. *Down To Business: It's Past Time To Elevate The Infosec Conversation*. [Online; accessed 5. Apr. 2020]. Apr. 2008. URL: https://web.archive.org/web/20080414031843/http://www.informationweek.com/news/security/client/showArticle.jhtml?articleID=207100989.

[51] S. M. Z. U. Rashid, M. I. Kamrul and A. Islam. 'Understanding the Security Threats of Esoteric Subdomain Takeover and Prevention Scheme'. In: *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*. 2019, pp. 1–4.

[52] J. Reavis. *Trend to ponder: Passive vulnerability assessment*, 2003. URL: https://searchsecurity.techtarget.com/tip/Trend-to-ponder-Passive-vulnerability-assessment (visited on 21/02/2019).

[53] Redteamsecurity. *AutoNessus, Github*. URL: https://github.com/redteamsecurity/AutoNessus. (visited on 09/05/2019).

[54] O. Reeves. *Gobuster*. URL: https://github.com/OJ/gobuster. (visited on 14/05/2019).

[55] Paul Ritchie. 'The security risks of AJAX/web 2.0 applications'. In: *Network Security* 2007.3 (2007), pp. 4–8. ISSN: 1353-4858. DOI: https://doi.org/10.1016/S1353-4858(07)70025-9. URL: http://www.sciencedirect.com/science/article/pii/S1353485807700259.

[56] M. Rouse. *Vulnerability scanner*. July 2006. URL: https://searchsoftwarequality.techtarget.com/definition/vulnerability-scanner (visited on 23/02/2019).

[57] D. Salmon. *Github - S3Scanner,* URL: https://github.com/sa7mon/S3Scanner. (visited on 13/05/2019).

[58] Offensive Security. *DirBuster Package Description*. URL: https://tools.kali.org/web-applications/dirbuster (visited on 19/05/2019).

[59] *SHODAN FAQ*. URL: http://www.shodanhq.com/help/faq (visited on 15/05/2019).

[60] Adam Shostack. *Threat Modeling: Designing for Security*. Wiley, Feb. 2014. ISBN: 978-1-118-80999-0. URL: https://www.wiley.com/en-us/Threat+Modeling%3A+Designing+for+Security-p-9781118809990.

[61] SophosLabs research team. *Security Threat Report for 2020.* [Online; accessed 5. Apr. 2020]. Jan. 2020. URL: https://www.sophos.com/en-us/medialibrary/pdfs/technical-papers/sophoslabs-uncut-2020-threat-report.pdf.

[62] Kevin Stankiewicz. ''The Hot Zone' author warns the next pandemic could 'balloon faster' than the coronavirus'. In: *CNBC* (Apr. 2020). URL: https://www.cnbc.com/2020/04/03/the-hot-zone-author-next-pandemic-can-be-worse-than-the-coronavirus.html.

[63] *Subdomain-bruteforcer (SubBrute)*. URL: https://github.com/TheRook/subbrute (visited on 04/05/2019).

[64] Rohini Sulatycki and Eduardo B. Fernandez. 'Two Threat Patterns That Exploit "Security Misconfiguration" and "Sensitive Data Exposure" Vulnerabilities'. In: *Proceedings of the 20th European Conference on Pattern Languages of Programs*. EuroPLoP '15. Kaufbeuren, Germany: Association for Computing Machinery, 2015. ISBN: 9781450338479. DOI: 10.1145/2855321.2855368. URL: https://doi.org/10.1145/2855321.2855368.

[65] The European Union Agency for Network and Information Security (ENISA). *Glossary - ENISA*. URL: https://www.enisa.europa.eu/topics/threat-risk-management/risk-management/current-risk/risk-management-inventory/glossary (visited on 21/02/2019).

[66] *The Old and New: Current Trends in Web-based Threats*. [Online; accessed 5. Apr. 2020]. June 2018. URL: https://unit42.paloaltonetworks.com/unit42-the-old-and-new-current-trends-in-web-based-threats.

[67] the University of Oslo. *Zoom: Videomøter og digital undervisning - Universitetet i Oslo*. [Online; accessed 6. Apr. 2020]. Apr. 2020. URL: https://www.uio.no/tjenester/it/telefoni-sanntid/videokonf/zoom.

[68] *The Web Application Security Consortium / Web Application Security Statistics*. URL: http://projects.webappsec.org/w/page/13246989/Web-Application-Security-Statistics/#APPENDIX2ADDITIONALVULNERABILITYCLASSIFICATION (visited on 21/02/2019).

[69] *tko-subs*. URL: https://github.com/anshumanbh/tko-subs (visited on 19/05/2019).

[70] U.S.-C.E.R.T. *DNS Amplification Attacks*. 19th Oct. 2016. URL: https://www.us-cert.gov/ncas/alerts/TA13-088A (visited on 04/05/2019).

[71] John Viega and Gary McGraw. *Building Secure Software: How to Avoid Security Problems the Right Way (Addison-wesley Professional Computing Series)*. Addison-Wesley Professional, Oct. 2001. ISBN: 978-0321774958. URL: https://www.amazon.com/Building-Secure-Software-Addison-wesley-Professional/dp/0321774957.

[72] Brandon Vigliarolo. 'Who has banned Zoom? Google, NASA, and more'. In: *TechRepublic* (Apr. 2020). URL: https://www.techrepublic.com/article/who-has-banned-zoom-google-nasa-and-more.

[73] *Vulnerability Scan vs Vulnerability Assessment: What's the Difference?* [Online; accessed 30. May 2020]. May 2018. URL: https://www.hitachi-systems-security.com/blog/vulnerability-scan-vs-vulnerability-assessment/#targetText=Pentesting%3A%20A%20Vulnerability%20Assessment%20with,you%20are%20trying%20to%20protect.

[74] *Vulnerability Scanning*. URL: https://www.techopedia.com/definition/4160/vulnerability-scanning. (visited on 02/04/2019).

[75] *What is Shodan?* URL: https://help.shodan.io/the-basics/what-is-shodan (visited on 13/05/2019).

[76] David Wind. 'Alexa Top 1 Million Security - Hacking the Big Ones'. In: *Proceedings of IT-SECX 2018*. Matthias Corvinus-Straße 15, 3100 St. Pölten Austria, 2018. URL: https://slashcrypto.org/data/itsecx2018.pdf.

[77] xxdesmus. 'Thai Database Leaks 8.3 Billion Internet Records'. In: *Rainbowtabl.es* (May 2020). URL: https://rainbowtabl.es/2020/05/25/thai-database-leaks-internet-records.

[78] ZMap. *Internet-Wide Scan Data Repository*. URL: https://scans.io/ (visited on 09/05/2019).