Bachelor's Thesis Computing Science

Stronghold: Automating corporate security.

A novel approach at pipelining the entire security improvement cycle for Microsoft and Google environments.

Joost Grunwald
s1057493

May 11, 2023

First supervisor/assessor:
dr. Simona Samardjiska

Second assessor:
dr. ir. Erik Poll

Radboud University

Abstract

This paper outlines a novel approach to the ongoing security improvement cycle for both Microsoft and Google environments. The aim is to develop tools for Active Directory, Azure Active Directory and Google Workspace. In addition to this, we introduce Besieger for auditing websites. We integrate Nessus into our pipeline for internet network auditing. We also introduce phishing simulations specifically finetuned for Google and Microsoft environments to do aimed phishing simulations. The proposed pipeline involves a series of steps for the assessment, planning, deployment, and monitoring of security solutions to strengthen and protect corporate environments. We call the entire system Stronghold, a solution which aim is to automate the security improvement cycle. It aims to go from vulnerability detection to remediation within one single encapsulated environment. Stronghold manages to get State of the Art results in vulnerability detection and remediation while also introducing a new security remediation pipeline called FRIS. The goal of this pipeline is to offer information, further research, and even direct solutions and scripts for found vulnerabilities. We develop one environment aimed at both State of the Art results and State of the art usability and speed. This is to combat the high-cost problem that often arises in corporate environments where high costs or minimum budget leads to ignorance of security problems.

Contents

1 Introduction **3**

2 Preliminaries **5**
2.1 Active Directory 5
2.1.1 Services 5
2.1.2 Group policies 6
2.1.3 Domain name services 6
2.1.4 Kerberos 6
2.2 Vulnerabilities inside Active Directory 7
2.3 Azure Directory Passwords 7
2.3.1 Password policy 7
2.4 Azure Active Directory 7
2.4.1 MFA 8
2.4.2 Conditional access 8
2.4.3 Compliance policies and bitlocker 8
2.5 Google (Cloud) Workspace 9
2.6 Phishing and Spam in corporate environments 10

3 (A)AD Vulnerability Cycling **12**
3.1 FRIS in action 13

4 SOTA AD Pentesting **17**
4.1 Methodology 17
4.2 Test Environment 17
4.3 Overlap between SOTA 18
4.4 SOTA AD vulnerability finding 20

5 Google Workspace auditing **21**
5.1 Methodology 21
5.2 Vulnerability Assessment 22

6 Corporate Phishing Simulation **24**
6.1 Methodology 25
6.2 Iterative Improvement 26

6.3 Sophistication . . . 28
6.3.1 Attack 1 . . . 28
6.3.2 Attack 2 . . . 30
6.3.3 Attack 3 . . . 30
6.3.4 Uniquely identifying every user . . . 31
6.4 Quiz . . . 31

7 Besieger: Automatic web pentesting **32**
7.1 Surface level vulnerabilities . . . 33
7.1.1 HTTP(s) (headers) . . . 33
7.1.2 Cookie vulnerabilities . . . 35
7.2 Version identification and exploitation . . . 36
7.3 Fuzzing and crawling . . . 36
7.4 DNS and subdomain enumeration . . . 37
7.5 Web Application Firewalls (WAF) . . . 37
7.6 SSL and ciphers . . . 37
7.7 Other attacks . . . 37

8 Scan results on real-life environments **38**
8.1 Active Directory . . . 38
8.1.1 Main Report . . . 39
8.1.2 Management summary . . . 39

9 Related Work **44**

10 Discussions **45**
10.1 Discussion points . . . 45
10.1.1 Interpretation of results . . . 45
10.1.2 Comparison with previous research . . . 45
10.1.3 Limitations and future research . . . 46
10.1.4 Implications . . . 46
10.2 Future work . . . 46

11 Conclusions **48**

Chapter 1

Introduction

Corporate environments are organizations or businesses that operate with the goal of generating profits. These environments often handle sensitive information such as financial data, intellectual property, and personal information of employees and customers. Because of this, security is extremely important in corporate environments [5].

There are several reasons why security is often disregarded in corporate environments. One reason is that companies may prioritize cost-cutting measures over security, leading them to neglect investing in strong security measures [1]. Another reason is that some companies may not have a clear understanding of the risks and vulnerabilities that exist within their systems, leading them to underestimate the importance of security [9]. Additionally, some companies may have a culture that does not prioritize security, which can lead to a lack of attention to security protocols and practices [13].

It is common for corporate environments to use solutions from Microsoft or Google for their domains. These companies offer a range of products and services such as email, cloud storage, and productivity tools that are widely used by businesses [43, 59]. Most often used are Google Workspace and Microsoft (Azure) Active Directory. Using solutions from these companies can help organizations streamline their operations and improve productivity, but it is important for companies to also ensure that they have adequate security measures in place to protect their sensitive data [19].

Overall, security is crucial in corporate environments as it helps to protect sensitive information and prevent unauthorized access or breaches [64]. It is important for companies to prioritize security and invest in strong security measures to protect their assets and maintain the trust of their employees and customers [5].

This paper presents a novel approach to Google Workspace and (Azure) AD security improvement, which is an automated pipeline that encapsulates the entire security improvement cycle, with the goal of maximum security improvements with minimum effort. In order to facilitate these systems, we present our novel security system called Stronghold. This system aims to:

- Find as many vulnerabilities in the Google/(A)AD environment as possible.
- Offer information about all found vulnerabilities.
- Offer links and further reading for all found vulnerabilities.
- Offer impact analysis for possible patches to make sure the system keeps working properly after patching.
- Offer solutions for all vulnerabilities, completing the pipeline.

This system fits our aims because it offers a complete step-based pattern to quickly 1 find the problem 2 understand the problem 3 possibly read deeper into the problem 4 analyze the impact of the problem and 5 solve the problem. The user quickly embarks on a guided journey that allows him to improve his security without ever having to really leave our assessment ecosystem. The ease of usability is further underlined by offering complete web pages and reports. The urgency gets underlined by a set of security scores that are generated based on our findings. Our work for Microsoft is somewhat related to the Purple Knight [11] and the PingCastle [41] tools, both are security frameworks based upon finding vulnerabilities in (Azure) Active Directory. We differ from these systems by finding more vulnerabilities and by going further than just finding problems, introducing our FRIS pipeline.

Our work for Google is novel, as far as we know of, there are no automatic tools to enumerate weaknesses in Google workspace settings, which was confirmed by a google employee after a support call from our end [17].

Our first chapter is about the preliminaries of this research, diving into the Google Workspace and (Azure) Active Directory infrastructures and about the possible sources of vulnerabilities. We then provide a scientific hypothesis and explain the requirements of our proof and provide the research we did. We then dive into related work and the current state of the art in vulnerability frameworks for Google and (Azure) Active Directory. We then conclude with our results and a direct comparison between our framework and existing systems.

Chapter 2

Preliminaries

This chapter aims to introduce the concepts that are important in order to fully understand the depth of the work and the concepts that are discussed in later chapters. It aims to give an extensive overview of the underlying structures that form the foundation of this work. In order to fully understand vulnerabilities it is quite essential to grasp the structures that facilitate them.

2.1 Active Directory

Active Directory is a Microsoft system that functions as a directory service for the Windows domain [12]. It is composed of a set of processes and services, the primary one being the Domain Service Role, or Domain Controller. This server authenticates, authorizes, enforces policies, stores information, and provides rights and roles to all members of the Domain. The active directory environment is comprised of multiple components with individual functions and vulnerabilities.

2.1.1 Services

Active Directory (AD) services provide a centralized, secure, and globally managed repository of user and resource data [47]. The main services are authentication, authorization, user management, and information storage. Authentication verifies the identity of a user in the network, while authorization grants access to resources or services. User management tasks include creating, managing, and deleting user accounts. Information storage enables users to store and manage data in a secure and reliable manner. Additionally, AD services provide single sign-on (SSO) capabilities, allowing users to access multiple systems and applications with a single login [3]. AD also offers directory synchronization, which allows organizations to keep multiple directories in sync across multiple computers and locations. Overall, Active

Directory services provide a robust and secure platform for managing user and resource data.

2.1.2 Group policies

Group policies enable administrators to centrally manage and control the configuration of users and computers in a domain [58]. Group policies are applied to computers or users in a specific organizational unit (OU) which are placed in a hierarchical structure to divide and manage objects. Group policies can be used to configure user settings and control computer settings such as software installations, system updates, network access, security settings and user rights [58]. In addition to controlling the configuration of objects, group policies are also used to deploy software and scripts across the network. Group policies enable administrators to manage and control the configuration of users and computers and deploy applications and scripts quickly and easily.

2.1.3 Domain name services

Domain name services (DNS) provide a way to locate resources and services on a network [42]. DNS is used to translate human interpretable names (e.g. www.example.com) to IP addresses. DNS resolves the named host or service to its IP address, thus allowing other hosts to connect to it. DNS also provides a way to organize and manage information in the domain. All domain computers are registered with a DNS record, which can also include information about services offered by the host [42]. DNS enables users to access resources quickly and easily by providing an easy way to locate resources and services in the domain.

2.1.4 Kerberos

Kerberos is a network authentication protocol used to authenticate users and services in a secure manner [49]. Kerberos uses tickets to authenticate users and provides strong cryptography to protect the authentication process. Tickets are obtained from a Kerberos server and the client must present the ticket to the service to prove their identity [49]. Kerberos is a trusted third-party authentication protocol and is used by many organizations to authenticate users and services in the domain. The use of Kerberos authentication provides an extra layer of security to the domain and ensures that user credentials are kept safe and secure.

2.2 Vulnerabilities inside Active Directory

Active Directory is a powerful directory solution and is used in many organizations for authentication, authorization, and user management [12]. However, with the increased use of AD, the threats to its security have increased as well. Some of the common vulnerabilities of AD include weak passwords, incorrect permissions settings, inadequate patch management, and lack of auditing procedures [57]. Additionally, there are security vulnerabilities that exist within the AD framework itself, such as privilege escalation, replication issues, and denial of service attacks [57]. As with any system, vulnerabilities exist even if the system is secure in itself. It is important to identify and address these vulnerabilities to maintain a secure environment.

2.3 Azure Directory Passwords

Azure Directory, like almost every system nowadays, uses passwords for authentication [33]. Passwords in active directory are of extreme value, as there is no such thing as Multi-Factor Authentication present in these environments. Hence knowing a password gives you full access to an account. Therefore using tools to examine password strength and choice is a big part of the work we have to do in the assessment of an active directory server.

2.3.1 Password policy

Active Directory uses a global password policy to set certain requirements for passwords [3]. It allows IT administrators to set a minimum password length. It also allows them to require complexity, meaning that of lowercase letters, uppercase letters, numbers, and other symbols, three of these four have to be used in a password. In addition to this the password policy offers options for password history, disallowing previous passwords and an option for password lockout, to specify after how many attempts a user is locked out and how long it takes before they can log in after a lockout again [58]. Notice that this kind of policy allows for a huge amount of problems to quickly arise. A to short minimum password length is an enormous problem because it makes life way too easy for hash crackers. A lockout amount of 0 allows for unlimited login possibilities and password spraying/brute-forcing attack. Therefore carefully setting up this policy is often one of the most important responsibilities of the AD administrator.

2.4 Azure Active Directory

Azure Active Directory is the cloud-based alternative for Azure Directory [33]. While this version does not have the amount of options and depth as

Azure Directory, it has a load of different features and defense mechanisms. A substantial example of this being MFA. The system is a little less prone to vulnerabilities due to old mechanisms and systems because it is hosted on Microsoft's end. It is also newer, meaning fewer vulnerabilities due to old age occur.

2.4.1 MFA

MFA, short for Multi-Factor Authentication, is the new and upcoming standard for authentication [38]. This form of authentication adds a second layer of defense to authentication as it requires users to verify their authentication requires using a second factor. In practice, this is often by using the Microsoft authenticator app on their phone. This makes it much harder for hackers to simply crack a hash and use a password, because now they also need access to a phone. While MFA is a deal changer in the scene, there are also some weaknesses known.

1. SMS spoofing, to know a possible sms message [**?**].
2. MFA often is required only once in x days, so you can still use a colleagues laptop and use it to bypass MFA.
3. Stolen session cookies can be authenticated for a certain period before asking for MFA again [38].

2.4.2 Conditional access

In azure active directory, one of your first lines of defense becomes conditional access [33]. Conditional access, as given away by the name, lets you block traffic or enforce requirements on it based on certain conditions that you can set up. In practice, this is used to block legacy authentication (non-MFA authentication). But also to require MFA, to block persistent browser sessions, noncompliant devices, etc. It can also be used to block certain countries or exclude certain IP ranges like the company office from MFA [33].

2.4.3 Compliance policies and bitlocker

Two powerful features which we also think are important security mechanisms inside the AAD environment are compliance policies and a BitLocker policy [34]. With compliance policies, we can require devices in our network to be of a certain safe windows version, in addition to this we can require secure boot to be on and we can also set requirements in how good their computer pin/password should be [34]. Therefore we can require some basic safety precautions for devices that want to be on our network. With a BitLocker policy, we can automatically roll out BitLocker on devices, securing their hard disk and making sure that the data on it is properly protected in case of theft [34].

2.5 Google (Cloud) Workspace

The Google Workspace (formerly known as G Suite) is a collection of productivity and collaboration tools offered by Google [28]. It includes a range of services such as Gmail, Google Drive, Google Calendar, and Google Docs, which allow users to communicate, store and access files, schedule events, and create and edit documents online. Some of the key security options within Google Workspace include:

1. Encryption: Google Workspace uses encryption to protect the confidentiality of users' data while it is in transit and at rest [27]. This includes the use of Secure Sockets Layer (SSL) and Transport Layer Security (TLS) for data transmission, as well as AES and SSL encryption for data storage.

2. Two-factor authentication: This is an additional layer of security that requires users to provide a second form of authentication, such as a code sent to their phone, in order to access their account [22]. This helps to prevent unauthorized access to users' accounts.

3. Access controls: Google Workspace allows administrators to set up user accounts and permissions, which can be used to control access to different tools and services within the suite [30]. This includes setting up roles and permissions for different users, as well as specifying the types of actions that users are allowed to take within the tools.

4. Data loss prevention: Google Workspace includes tools for detecting and preventing the accidental or unauthorized sharing of sensitive data [26]. This includes the ability to set up data loss prevention policies to block the sharing of specific types of data, as well as the ability to track and audit data sharing activities.

5. Password security: Google Workspace includes tools for managing password security, including the ability to require strong passwords and to enforce password expiration policies [31]. Users can also use tools such as Google's Password Checkup to check the strength and security of their passwords.

6. Session length: Google Workspace allows administrators to set the length of time that user sessions will remain active, which can help to prevent unauthorized access to accounts if a device is left unattended [32]. Administrators can also set up inactivity timeout policies to automatically log users out after a certain period of inactivity.

7. Context-aware access: Google Workspace includes context-aware access controls, which allow administrators to set up policies that grant or restrict access to certain tools and services based on the context in which they are being used [24]. For example, administrators can set up policies that only allow access to certain tools from specific locations, devices, or networks.

8. Phishing and spam protection: Google Workspace includes tools for detecting and blocking phishing and spam emails [29]. These tools use a combination of machine learning algorithms and user feedback to identify and block malicious emails. Users can also report suspicious emails to help improve the effectiveness of the spam filters.

9. SPF, DKIM, and DMARC: Google Workspace supports the use of SPF (Sender Policy Framework), DKIM (DomainKeys Identified Mail), and DMARC (Domain-based Message Authentication, Reporting, and Conformance) to help protect against email spoofing and phishing attacks [23]. These protocols allow administrators to verify the authenticity of emails that are sent from their domain, and to block emails that fail these checks.

10. External users and external sharing: Google Workspace allows users to share files and collaborate with people outside their organization [25]. However, it is important to be cautious when sharing data with external users, as this can increase the risk of data leaks and other security vulnerabilities. Google Workspace provides tools for managing external sharing, including the ability to set up policies that control who can access shared data and how it can be shared. It is also important to note that mail forwarding to external addresses is disabled by default in Google Workspace to prevent data leaks [25].

2.6 Phishing and Spam in corporate environments

Spam and phishing are two forms of online communication that can pose significant risks to corporate environments [61]. Spam refers to unsolicited electronic messages, often sent in large volumes, with the intent of promoting a product or service. These messages can clog up email inboxes and distract employees from more important tasks. Phishing, on the other hand, involves the use of fraudulent emails or websites to trick individuals into revealing sensitive information, such as login credentials or financial information [6]. In a corporate environment, spam and phishing can pose a number of risks. These include:

- Loss of productivity: As mentioned, spam emails can distract employees and reduce their productivity. Phishing attacks can also require significant time and resources to investigate and mitigate.

- Data breaches: Phishing attacks can lead to the compromise of sensitive corporate information, such as intellectual property or customer data [35].
- Financial losses: Phishing attacks can result in the theft of financial information or the unauthorized transfer of funds [18].

To mitigate these risks, it is important for corporate environments to implement robust spam and phishing prevention measures. One effective approach is the use of phishing simulations, which involve the creation and distribution of simulated phishing attacks to employees [21]. These simulations can help educate employees about the risks of phishing and test their ability to identify and report suspicious emails. Other important background concepts related to phishing simulations include:

- Social engineering: Phishing attacks often rely on social engineering techniques to manipulate individuals into revealing sensitive information or performing certain actions [48].
- Email security: Measures such as authentication and encryption can help protect against the interception of emails and the compromise of sensitive information [20].
- User education: Providing employees with training on how to identify and report suspicious emails can help mitigate the risk of successful phishing attacks [36].

Chapter 3

(A)AD Vulnerability Cycling

We introduce a novel methodology for the vulnerability solving process called FRIS. This methodology expands the normal pipeline used by a security officer to encapsulate the information finding and problem solving part into the automated and offered pipeline. You can see the original pipeline and our novel extension of it in the figure below:

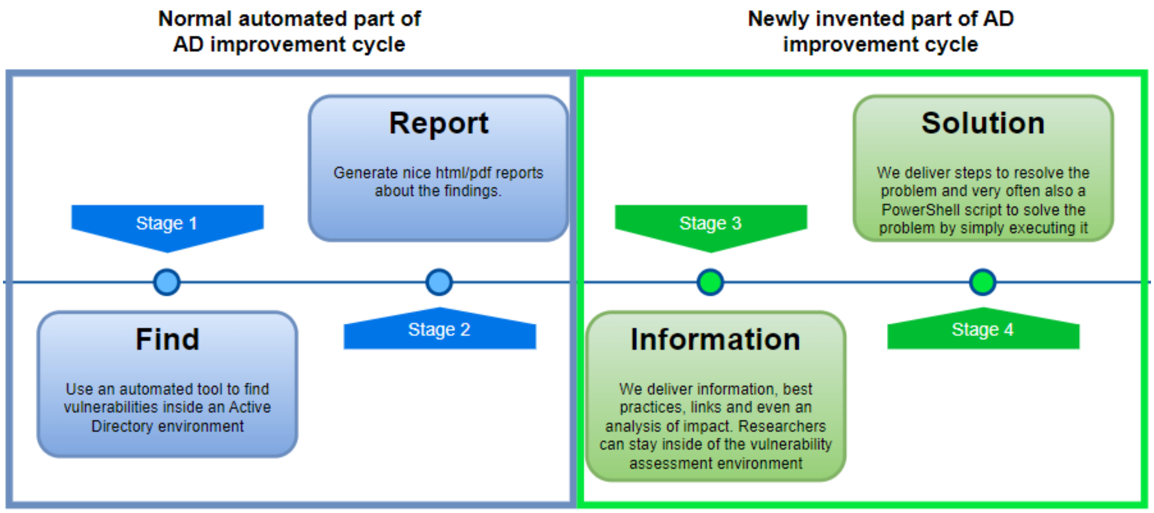

Figure 3.1: The old security improvement cycle vs the new one

Let us go more in depth about the FRIS cycle, FRIS stands for Find, Report, Inform, Solve. A pipeline we will explain a little bit more in depth:

1. Find, we find a vulnerability and confirm that it is present automatically
2. Report, we generate HTML documentation about the found vulnerabilities but also an HTML page per single vulnerability. In addition to

this, we use our novel scoring system to score the target on multiple categories and to give each vulnerability its own unique score.

3. Inform, inside our report, we offer information, so that the security researcher using Stronghold is immediately capable to continue his journey inside the vulnerability. Using summaries we wrote, but also links towards articles and official documentation, we allow the researcher to catch up on the topic and go more in-depth in a very short amount of time. Ensuring that all needed resources are already in hand reach by providing them to the security researcher through links. We also provide an impact analysis to go in-depth about possible interference's of solutions with the system.

4. Solve, we offer steps that can be followed to solve the problem and often offer a power shell script to immediately patch the vulnerability.

In a normal vulnerability scanning environment the Find step could be done by a vulnerability scanner, after which vulnerabilities and returned/reported and the Security Officer has to find information about the vulnerability/context himself, our aim is to cut time and resources and offer this information immediately. Making sure to also add links to guides and official documentation to further cut time. In addition to this, we offer the solution as well, trying to make this pipeline of solving vulnerabilities as fast and encapsulated as possible.

Note that we not only aim to extend this pipeline towards usability and solving but that we also aim to get a new SOTA in the part of the vulnerability assessment pipeline that is already present, so finding vulnerabilities.

3.1 FRIS in action

To showcase FRIS and its ease, we showcase it in a rather small and simple AD environment. Let us showcase the size of this environment with direct output of our Stronghold scan:

Figure 3.2: Information generated inside a Stronghold report

We already see some interesting things, the environment is rather small (63) users, but the admin percentage is way too big for what you would expect in a properly secured environment. The amount of vulnerabilities found is also quite high. This is a real-life business using Active Directory on which we ran a Stronghold scan. The scan uses its own scoring system to score the environment.

Figure 3.3: Scoring an actual corporate environment on security

The scores are very low, let us take a further look at the environment and showcase our new FRIS system. For each vulnerability that Stronghold detects, it creates its own database entry using the FRIS methodology, meaning that we can look at one of the critical vulnerabilities and use Stronghold to enforce our security.

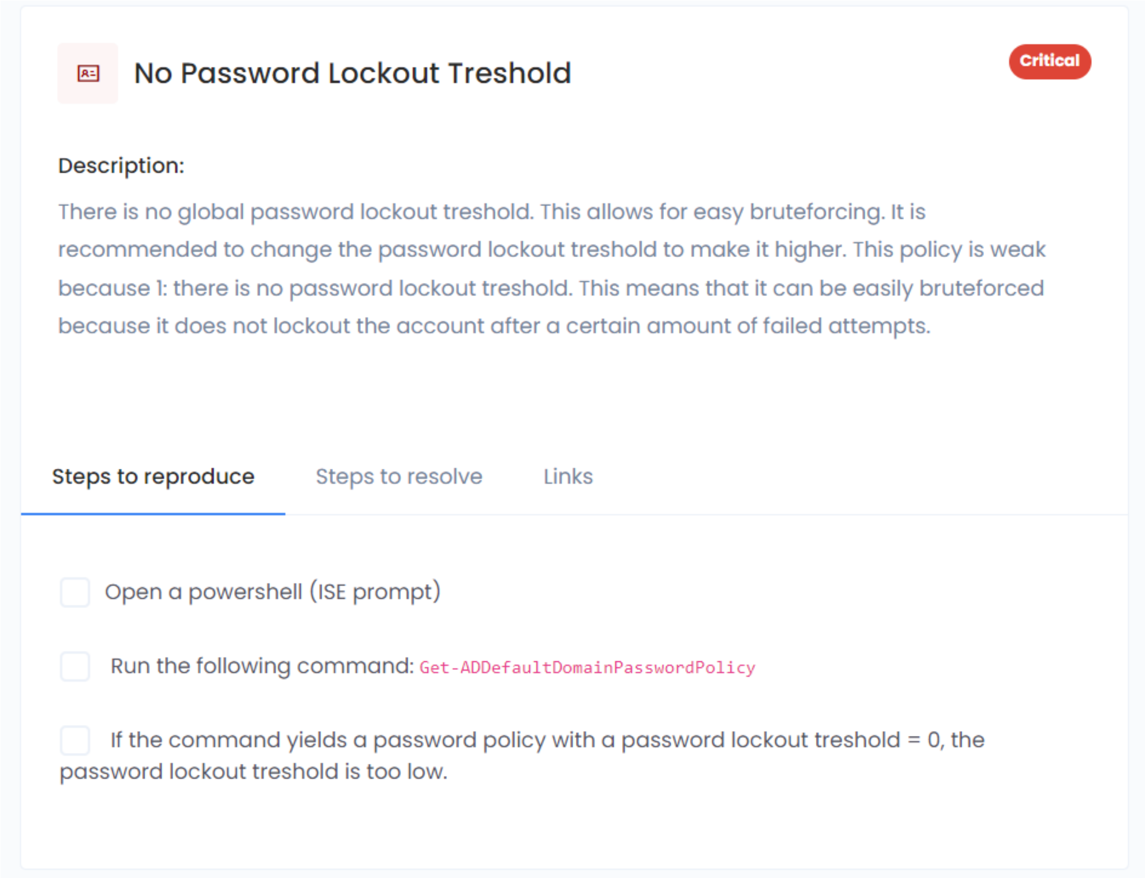

Figure 3.4: Examining a critical security flaw found by Stronghold

So there we have it, information about our critical vulnerability. This particular vulnerability makes it easy to just brute-force passwords without ever being locked out, hence why it is indeed quite critical. Now as you can see we can reproduce the presence of the vulnerability by simply following the provided steps, we can also embark further onto our FRIS journey by looking at a link to the official Microsoft password guidelines that is provided:

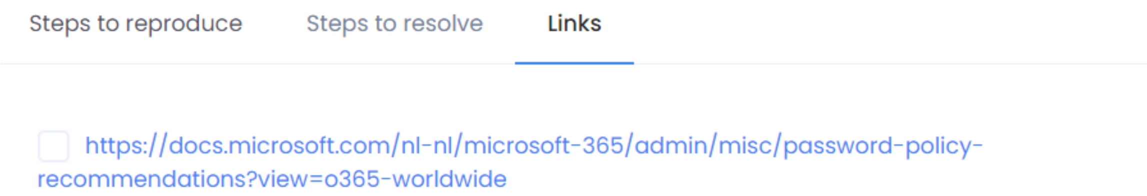

Figure 3.5: The inform step of FRIS further expanded

Then, for the last step of FRIS, we also still have to solve the actual vulnerability, let us embark on the last step in this journey:

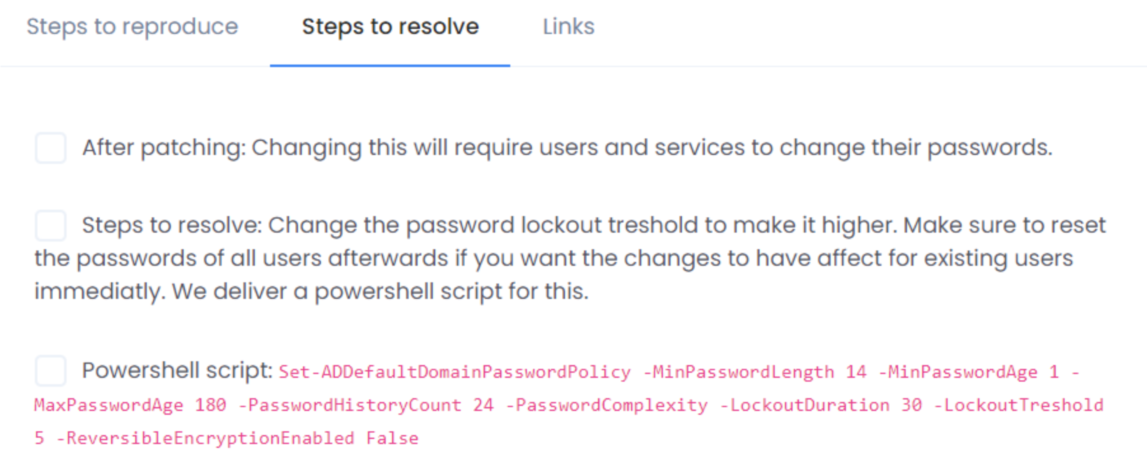

Figure 3.6: The solve step of FRIS

Perfect, we have got some user communication to do, the impact analysis of FRIS is showcased as well here. The PowerShell script provided actually updates the entire password policy to be conform Microsoft's best standards.

Chapter 4

SOTA AD Pentesting

In this chapter, we compare our results with the current State Of The Art in AD vulnerability finding. In order to do this we compare our results with Purple Knight [11] by Semperis and with Pingcastle [41]. The known SOTA products in the market for AD vulnerability finding.

4.1 Methodology

For these tests we only consider serious vulnerabilities, that is, vulnerabilities that are considered by Pingcastle [?]as scoring points for the security score. For Purple knight, we only consider vulnerabilities that are considered non baseline. (Warnings and criticals). For our own tool Stronghold, we are even harsher, we only consider vulnerabilities that are either medium (score > 6), high or critical. Therefore not counting low-scored medium, low, and baseline vulnerabilities. Note that Stronghold, during the research conducted found around 90-120 vulnerabilities per Active Directory it scanned. Filtering on serious vulnerabilities therefore partially cripples our system. We found it important to not only compare the number of serious vulnerabilities found but to also test how many of the vulnerabilities that other tools found, were found by Stronghold. We therefore directly compare the vulnerabilities found to find any vulnerabilities that Stronghold misses.

4.2 Test Environment

Thanks to Fourtop ICT, I had the possibility to test Stronghold inside real systems. Fourtop ICT is an MSP company that has multiple clients which leverage Active Directory systems. Most of the time these clients use all kinds of different AD infrastructures. I have worked with Microsoft server 2008, but I also found a server using Microsoft server 2022. This makes this environment perfect for testing and developing Stronghold. Note that all of these environments are used daily by companies that differ in size, I tested at companies with 8 users but also at companies with more than 200 users. It is important to me that we catch different Active Directory setups with different amounts of security measures and different amounts of changed settings over the years. The fact that we expose some very serious security flaws in these corporate environments underlines how helpful this branch of tools can be.

4.3 Overlap between SOTA

When compared to the vulnerabilities that SOTA tools PingCastle [41] and Purple Knight [11] find, Stronghold scores very well, it is able to find almost all vulnerabilities that these tools find, let us first compare our tool with Pingcastle [41] in 6 different corporate environments:

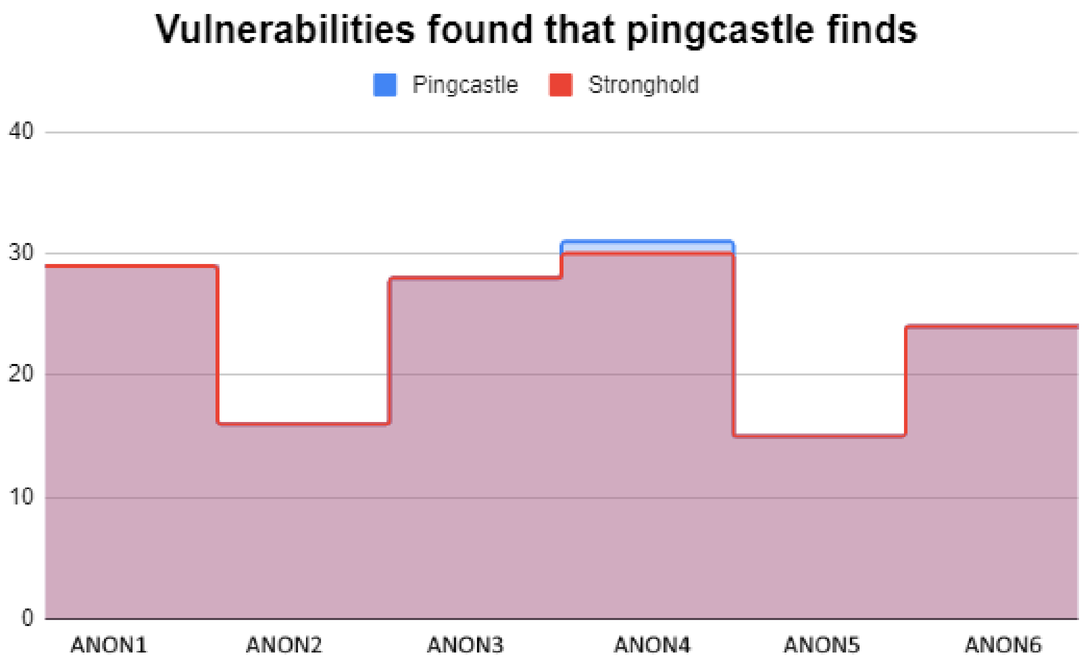

Figure 4.1: Missed vulnerabilities compared with PingCastle

Note that we only consider serious vulnerabilities for this graph, as discussed above. If we look at the results we see that we find almost all vulnerabilities that Pingcastle [?]does. This includes some of the rarer vulnerabilities that we found like WSUS being run over an insecure HTTP protection, weak certificates, or exchange permissions leading to privilege escalation. The missing Vulnerability at ANON4 has to do with incomplete subnets. We have written a script to find these as well, but at the time of testing, this was something that didn't work properly for ANON4. Let us now take a look at the vulnerabilities that we find from the ones purple knight found:

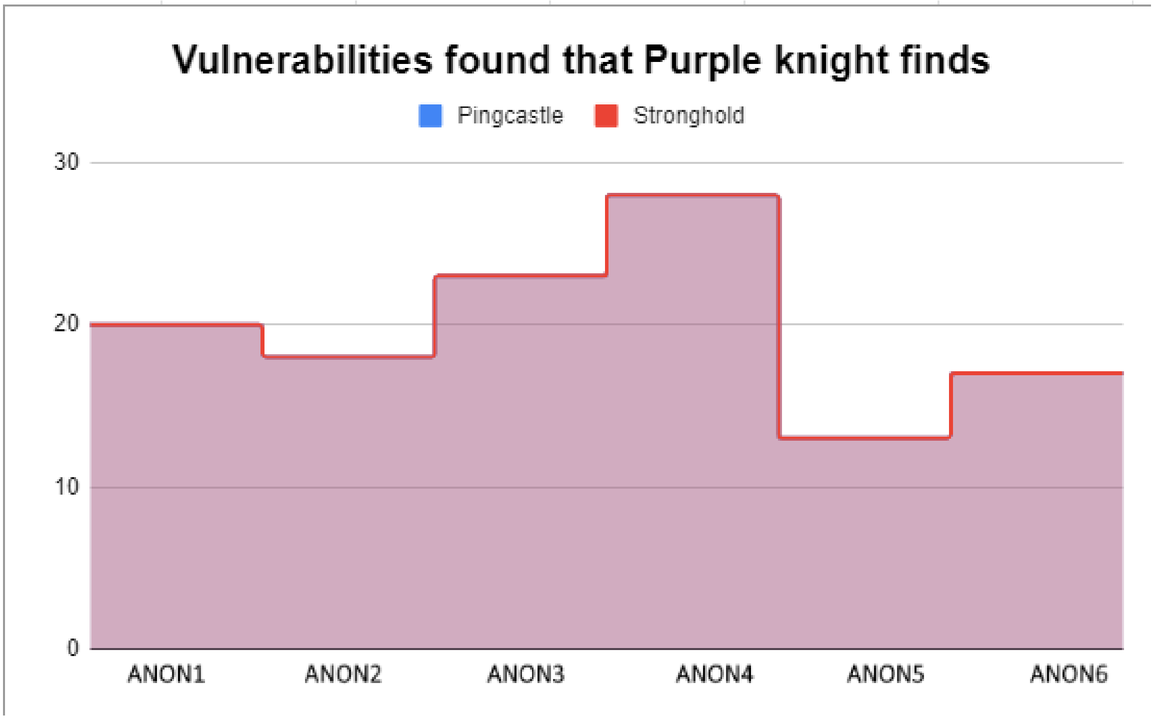

Figure 4.2: Missed vulnerabilities compared with Purple Knight

We see that we can find every vulnerability within our test environments that Purple knight can find. Purple knight goes a little bit less into depth and more into the more general checks like properties, DCsync rights, and excessive rights. Some of the vulnerabilities in Purple Knight [11] were not exactly found in the same way. Purple Knight, for instance, finds that the exchange server has a property that allows it to have admin rights. Our system Stronghold finds that exchange has been given many rights and should have its rights reduced. Because our solution fixes the problem that Purple Knight found and because we have found the vulnerability but labeled it otherwise, we find that this is not a vulnerability we missed.

4.4 SOTA AD vulnerability finding

In order to fully qualify Stronghold as the new state-of-the-art tool in AD security assessment, we want to directly compare it with its competitors in different environments. The aim is to show that we can consistently outperform the current state of the art to set a new standard in AD pen testing:

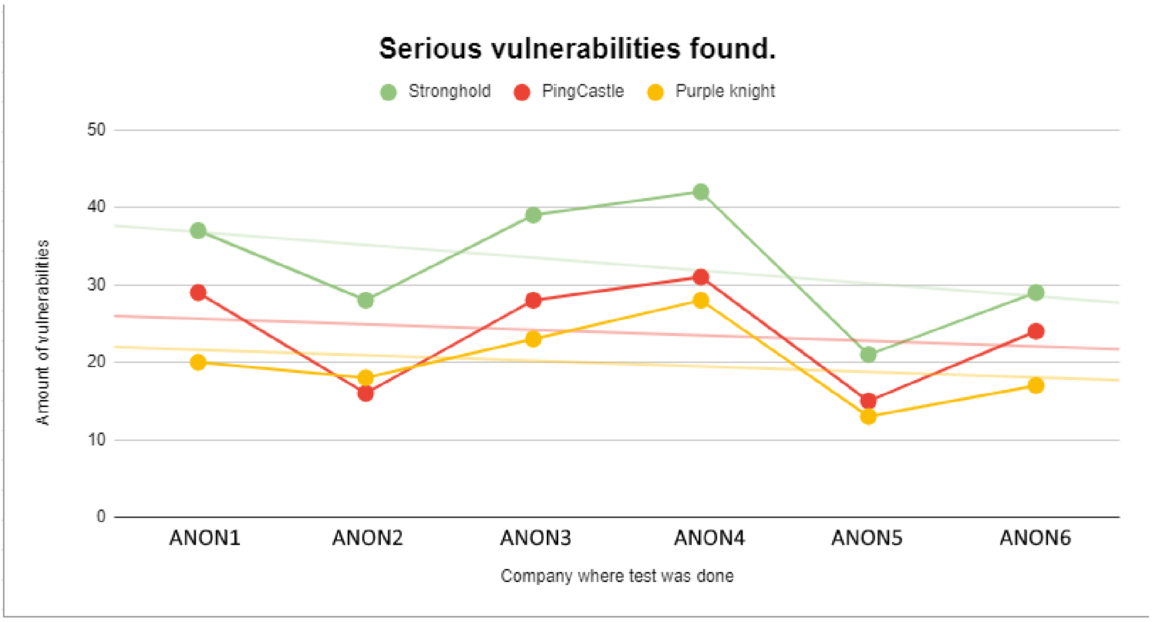

Figure 4.3: Stronghold compared with similar Tools

Chapter 5

Google Workspace auditing

In this chapter, we introduce a novel system that we created to scan Google Workspace for security misconfigurations and related vulnerabilities. After discussing with google support we made sure that to the best of our knowledge there is no tool at the moment that is able to enumerate Google Workspace for vulnerabilities.

5.1 Methodology

For these tests we look at all kinds of vulnerabilities, that is vulnerabilities Stronghold marks as baseline, low, medium, high, or as critical. We will add information about the vulnerability distribution to make sure that we properly show the findings we did. Note that Stronghold, during the research conducted found around x vulnerabilities per Google Workspace it scanned.

Thanks to stichting Nuwelijn/Muzerij, I had the possibility to test Stronghold inside real systems. Nuwelijn/Muzerij is an education foundation that utilizes a Google Workspace environment with 200 staff/teachers and a lot of students. The fact that we expose some very serious security flaws in this corporate environment underlines how helpful this branch of tools can be.

To properly audit the Google Workspace environment, we use selenium, which is a browser automation tool, to log in to the Google admin console (even with MFA by reproducing a token with the secret key). After logging in our tool automatically traverses sub-sections of the admin console and gets values of settings, parameters, and users. Based on these values and our vulnerability database, we generate a report filled with vulnerabilities and misconfigurations.

5.2 Vulnerability Assessment

At the moment, 29 vulnerabilities can be automatically found withing Google Workspace environments. This should be extended in the future to over 100 vulnerabilities and misconfigurations that are possible to be found within Google Workspace environments. The FRIS pipeline, leveraging solutions and dictionary objects for full vulnerability embeddings is maintained in our work for Google Workspace. An example of such an embedding can be found below.

```
# if contains "Share all information, and allow managing"
if "Share all information, and allow managing" in value:
    # vuln0019
    vuln0019 = Vulnerability
    (
        "0019",  # ID
        "Calendar sharing options are set to 'Share all information, and allow managing of calendars'",  # Name
        "The calendar sharing options are set to 'Share all information, and allow managing of calendars', this me
        "High"  # Severity
        "Go to https://admin.google.com/ac/managedsettings/435070579839/sharing, then disable the 'Share all infor
        "Go to https://admin.google.com/ac/managedsettings/435070579839/sharing, then check the 'Share all informa
        "7.2"  # grade
        "Services"  # category
        "NVT",  # powershellscript
        "https://support.google.com/a/answer/60765?"  # link1
        "https://webapps.stackexchange.com/questions/38995/cannot-share-more-than-free-busy-outside-of-organizatio
        "https://www.oreilly.com/library/view/hands-on-g-suite/9781789613018/5655910f-b4b7-48db-9489-0dda17451a2c.
        "External users will no longer be able to adjust agenda's, notify employees of this change and check who h
        "1"  # count
    )

    # add to vuln_list
    vuln_list.append(vuln0019)

    # do print
    print("Calendar sharing options are set to 'Share all information, and allow managing of calendars'")
```

Figure 5.1: Example of vulnerability

The tool then traverses the google admin console and some of the possible misconfigurations and prints matches. In the future this should be linked to our reporting environment.

```
▯[31m         Y8888888b     |  |         .+      d8888\  ` .'  `Y888888P'▯[0m
▯[31m         `88888888b    J  |      .-'     .od888888\.-'▯[0m
▯[31m          Y88888888b    \ |  .-'     d888888888P'▯[0m
▯[31m          `888888888b   \|-'       d888888888P▯[0m
▯[31m           `Y88888888b            d8888888P'▯[0m
▯[31m             Y88888888bo.      .od88888888▯[0m
▯[31m             `8888888888888888888888888888▯[0m
▯[31m              Y88888888888888888888888888P▯[0m
▯[31m               `Y8888888888888888888888P'▯[0m
▯[31m                 `Y8888888888888P'▯[0m
▯[31m                      `Y88888P'▯[0m

▯[33m                          STRONGHOLD GOOGLE EDITION              ▯[0m
▯[33m                                Version 0.1.4                          ▯[0m
▯[33mFinding vulnerabilities inside google workspace...▯[0m
Password strength is not enforced
Password reuse is allowed
Minimum password length is <= 8
```

Figure 5.2: Example of output

Chapter 6

Corporate Phishing Simulation

In this chapter, we introduce a novel system that does complex phishing simulations on Google and Microsoft-based environments. Our novel system called GateKnocker is designed solely for phishing attacks on these two corporate environments. There are many known phishing tools and phishing simulations, however, we wanted to build a system that can function inside our current FRIS environment, while also having the usability, sophistication, and specialization options we require. We created the following set of requirements to adhere to:

- GateKnocker should be fully automated and very easy to use.
- GateKnocker should be diverse in its attacks and amount of sophistication used.
- Gateknocker should be able to use sophisticated techniques like email spoofing or link masking techniques.
- Gateknocker should be able to use fully customized HTML templates for Microsoft and Google environments.
- Gateknocker should be able to identify mistakes by employees on a per-user basis.
- Gateknocker should include pre-written awareness emails and customized quizzes for Microsoft and Google environments.
- Gateknocker should be able to generate HTML documentation within our FRIS ecosystem.

The two key components to focus on are sophistication and usability. We aim to get State of the art results in usability to significantly reduce the costs of phishing simulation. On the other hand, we believe that we need State of the art sophistication to be a proper simulation of real-life events.

6.1 Methodology

Our methodology is developed based on maximum performance increase for the spotting of phishing by employees. Therefore the goal is not only to assess the current state of phishing identification within an environment but also to maximize the skill set of identifying phishing for the employees concerned. Research from [10] proved that repetition is vital for all learning. In addition, this research from [40] proves that making mistakes is an essential factor in learning as it helps us adjust wrong behavior. Taking this research into account, it is important that our system uses a systematic approach to repeat certain information in different ways, while also making sure said challenges are not too easy, so employees become aware of the mistakes they make while identifying phishing. We introduce our novel system called AT IT:

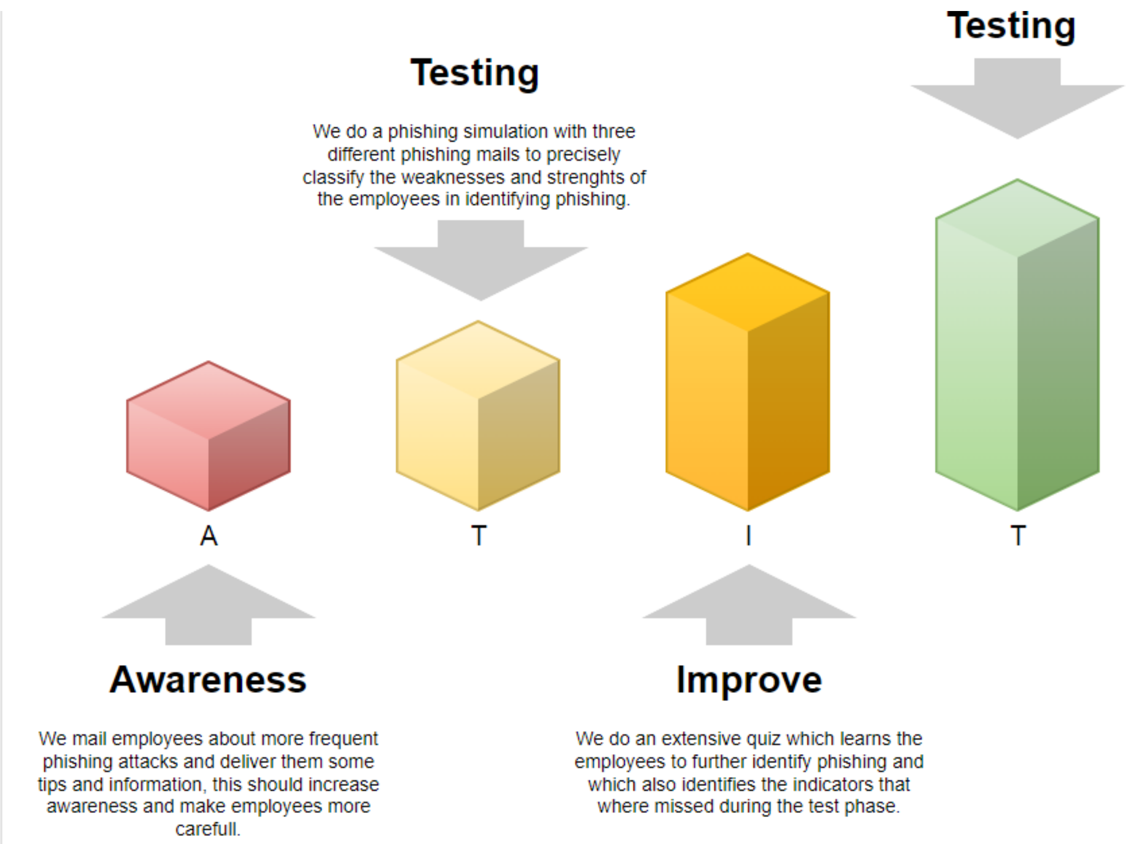

Figure 6.1: Our novel phishing learning system

ATIT, which stands for Alert, Test, Improve Test, is a system that is based on the way complex language models learn. Like language models, employees start with a set of knowledge about phishing and how to detect it, we amplify this knowledge by improving their awareness. Our initial Alert Step. We then Test to get an initial view of employee performance. We follow up with our Improve step: a lengthy quiz aimed at identifying indicators of phishing and improving from past mistakes. We finish with a reTest inside a new simulation to make a precise measure of made progress. This leaves us with a set of employees that are fine-tuned on the task of phishing identification. [15] While this also makes it possible to statistically grasp the improvements we made.

We have 4-fold repetition and a 3-fold ability to learn from made mistakes, adhering to the methodology we developed. We get a 3-fold chance to properly document and report our results within our FRIS ecosystem. We generate a report of the first simulation, where key awareness is pointed toward the mistakes that are most often made and the percentage of mistakes made. We specify our biggest area of growth here and use our quiz and tips to further train the employees. After the quiz, we generate a report of the results from the quiz and compare this with previous iterations to be able to report on the abilities of this particular batch of employees. We then retest and generate a third report on the performance of the employees after fine-tuning and on the improvements that were made using our AT IT methodology.

The novelty of the above system is its encapsulation, while there are tools for sending phishing emails or for doing quizzes, we offer the entire pipeline encapsulated, while also automated and reported automatically. The users are actively confronted with phishing and how to find it three times and are more informed or protected than after a regular single phishing simulation.

6.2 Iterative Improvement

In machine learning, iterative improvement is the act of approximating the ideal result in small steps. [45] In phishing environments, awareness created by phishing simulations and education becomes less over time. Research from [65] proves that the percentage of clicks in phishing simulations continues to lower over time, even after 20 simulation campaigns were run. We believe that in phishing, the employees that have the hardest time identifying phishing are the weakest link. As they are, by far, the biggest probable way of entry a phishing campaign could find into your environment. Hence, taking into account the algorithm for iterative improvement and the ongoing improvements shown when simulating for a longer period of time. We try to implement our own idea of Iterative Improvement within our AT IT ecosystem: From our simulations so far, we see that there is an approximate conversion rate of between 10-20 percent in a properly set up environment. We hence take the users that were tricked by our test, together with the 10 percent of worst-scoring employees on our quiz, and apply a follow-up learning step and simulation to reiterate their improvement cycle. We do this after a period of one month has passed. We also advise companies to redo phishing simulations every x year, to make sure they stay up to date with the latest sophisticated attacks discovered in the world of phishing.

6.3 Sophistication

We subdivide the number of users inside an organization into three groups, with three different types of attacks and three different phishing emails. This approach is aimed to test the environment for multiple different sources of attack and sophistication. In addition, the attacks are highly customized for the environment they try to penetrate.

6.3.1 Attack 1

The first attack is not aimed to surpass anti-spam (if present), as many phishing attacks will get detected and classified as spam. Let us show our HTML example for a Google environment:

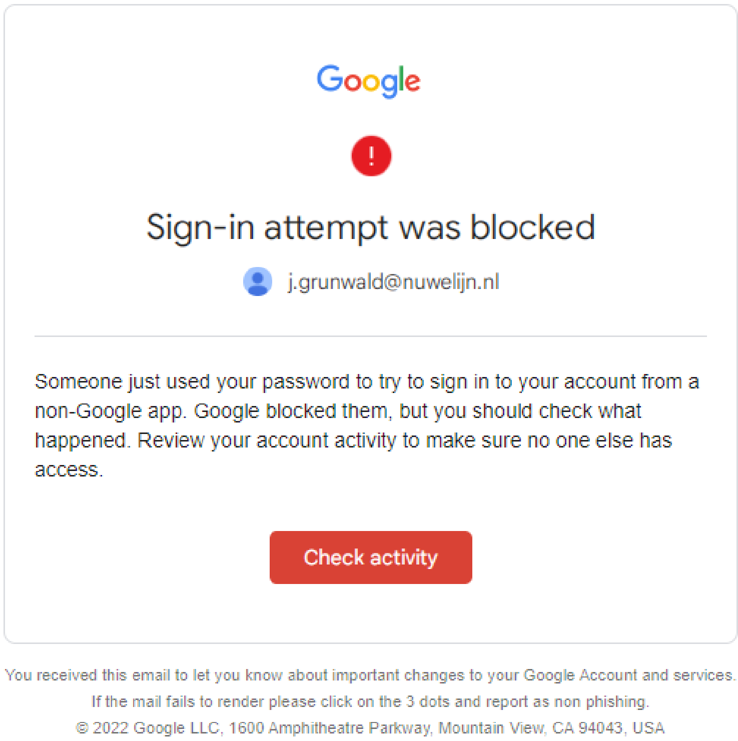

Figure 6.2: GateKnocker first attack

This attack is designed to be sophisticated and to get people to recognize the familiar Google check activity dashboard. It tests how well people spot this sophisticated attack and how well people check links and their spam sections. We can execute the attack with or without a handy piece of software called maskurl. This means that we can either use a bit.ly link or a https:www.google.com/security-measures@ZDFA kinda link. The HTML uses personal information (the victim's email) and looks realistic. We have a similar-looking example for Microsoft businesses:

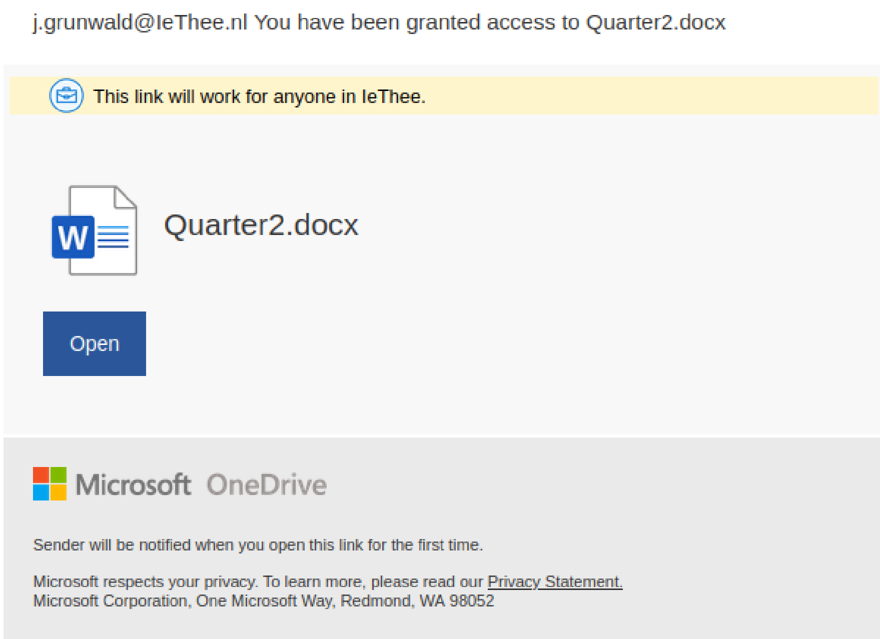

Figure 6.3: GateKnocker first attack

Note that while these attacks are sophisticated, they are also realistic and have even some build-in flaws we hope employees can use to recognize phishing/spam and which we can point out after a successful phishing simulation. The flaws here are:

- Send from a Gmail/outlook account closely related to the organization (for example for IeThee the mail comes from IeThee@gmail.com). The fact that the company domain is not used is an indication of possible phishing and should also mark emails as external when an environment is properly set up.
- Sophisticated emails like these should be marked as spam, which is a huge indicator that there could be something wrong with this mail.
- The name of the sender is Googl Support or Microsof Support. Notice the missing letter that should be an indicator of something being off.
- The link is either a bit.ly link or a google @ZDfA link, both are hidden but when hovering over the link should be an indicator of phishing.

6.3.2 Attack 2

The second attack will use email spoofing as its primary weapon, email spoofing is an attack in which we use a mail tool to send the email as if it comes from someone inside the organization. In this example, we use the fact that we send out phishing awareness emails earlier and fake a mail from internal IT staff that asks you to enable enhanced phishing protection settings and provides a link. We spoof the email address of someone working in IT and copy their mail signature to the end of our phishing mail, we then provide a link that is either a bit.ly link or a google @ZDfA link that directs to accounts recovery settings. We test credibility, email spoofing detection, and the use of credible information in phishing. We make sure there are also some weaknesses in this attack. The first one is a banner/spam section which should be attached by properly configured anti-spam.

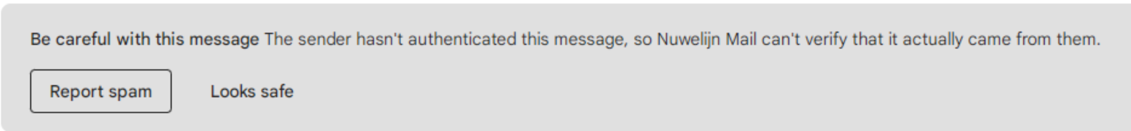

Figure 6.4: Email spoofing warning banner

Other weaknesses are:

- The link can be spotted again.
- The sender has no profile picture in spoofed mail while he has one in non-spoofed mail.
- The email should enter the spam section if spam settings are configured well enough.

6.3.3 Attack 3

The third attack is the most simple but also the most effective, as in our experience, it seems to bypass the spam settings. We do not use links to track who failed this attack. We create a simple email message from google support that asks a user if they can ask a single local IT personnel to disable the MFA of the victim. Leaving him/her vulnerable to attacks, we then monitor who of the victims actually mails our IT personnel asking him to disable the Multi-Factor Authentication. The email tests common sense and user reaction when not helped with banners/spam filters. There are some weaknesses in this mail again:

- Send from a Gmail/outlook account closely related to the organization (for example for IeThee the mail comes from IeThee@gmail.com). The fact that the company domain is not used is an indication of possible phishing and should also mark emails as external when an environment is properly set up.

- The name of the sender is Googl Support or Microsof Support. Notice the missing letter that should be an indicator of something being off.
- The mail asks to disable security settings, which should be a red flag without any other indicators already.

6.3.4 Uniquely identifying every user

We use the website wasitviewed.com and python to create custom emails for every user we mail to. We swap in their details and also create a unique link on wasitviewed. Wasitviewed mails us whenever one of these links is clicked, sending us the unique identifier as well and excluding traffic from bots and from our IP. This way our phishing bot can automatically register how many users fell for our traps and also exactly which users. Which is very handy information to register and to use to automate our reporting step.

6.4 Quiz

A phishing quiz is a tool that presents users with a series of simulated phishing attempts and asks them to identify the signs of a phish. This type of quiz can be an effective way to educate users about the types of tactics that attackers may use and help them become more vigilant in recognizing phishing attacks. We use our quiz to make sure our employees are trained to recognize the key patterns of phishing attacks, with a focus on the grammar of phishing emails, email addresses where mails are from, emails trying to be someone they are not, phishing and spam banners, email spoofing, and link authentication. With these examples and tips, we hope to use iterative improvement to take small steps in correcting the mistakes employees make in identifying phishing. We also introduce the examples employees were tricked by in our phishing simulation, allowing them to see the exact indicators they missed and providing a sense of urgency to our quiz. We specify these indicators and ask users to identify all indicators in various emails, training them on this and also telling them how to identify the ones they missed by automatic grading and tips provided in the Google forms/Microsoft forms environments.

Chapter 7

Besieger: Automatic web pentesting

In this chapter, we introduce a novel system that does complex web server audits. Our system called Besieger is designed solely for automated penetration tests of web related environments. There are many known web vulnerability scanners for this, however, we wanted a free tool that is able to get STOTA results in multiple web categories. We desire to focus on real-life environments, specializing on realistic real-life vulnerabilities and problems such as outdated software and probable mistakes. We utilise the power of open source for this and try to improve on state of the art tools such as Nessus, Acunetix, Invicti and Burp Suite Professional. We created the following set of requirements to adhere to:

- Besieger should be fully automated and very easy to use.
- Besieger should have multiple speed options and should consist of a high amount of different modular sub scanners that are select-able and combine-able.
- Besieger should be fully based on free and open source software.
- Besieger should be a very complete web attack tool, taking into account some OSINT attack vectors, some google dorking related attack vectors and version related attack vectors.
- Besieger should be able to chain attacks and use results of earlier modules in later modules.
- Besieger should be able to outperform the current state of the art DAST tools in real-life environments. It is a clear goal to perform better on this kind of environments, we do not care about performing better in test environments or in purposely vulnerable web applications.

- Besieger should be able to have a nice and easy console interface and coloring.
- Besieger should be focused on being a complete tool that is usable for Penetration Tests and reports of web applications.

Besieger is a command line tool, the only input it needs is a single URL. After a quick check to ensure the URL is actually reachable and correct, the scan automatically starts its magic.

Figure 7.1: A first glance at the CLI

7.1 Surface level vulnerabilities

When developing a brand new DAST tool, it is important to implement the basic but still often very handy sub-tools. We find it important to review websites based on HTTP settings, headers and cookie security. These are all fundamental and easy checkable elements that can still lead to some serious vulnerabilities when improperly implemented.

7.1.1 HTTP(s) (headers)

Besieger is able to find the following vulnerabilities:

- Clickjacking
- Missing security headers
- Incorrect security headers

- Content Security Policy bypasses
- HSTS and SRI checks
- Redirection and STS checks

It uses a combination of customly written code and external tools for this, such as nuclei and the Mozilla observatory. All findings are reported with our custom highlighting and printing system, that is partially inspired by the way nuclei prints it output. Let us show an example from running besiegers http module on the website of the Radboud University (https://www.ru.nl):

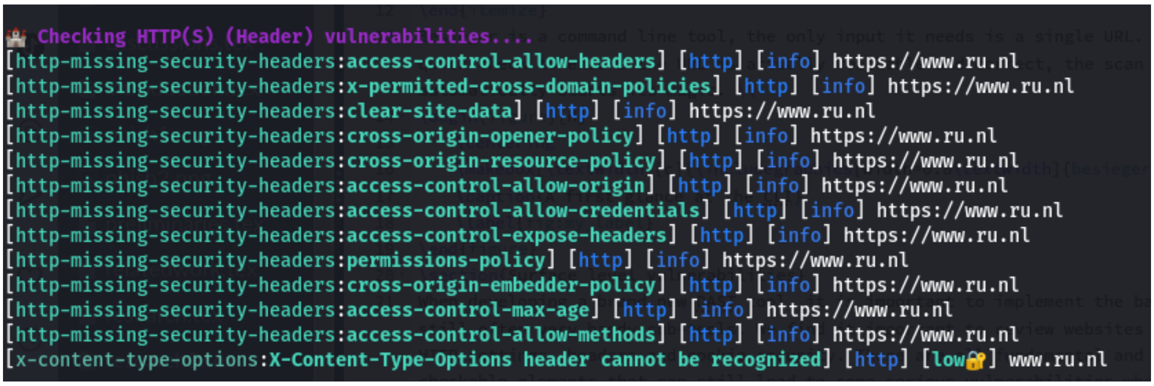

Figure 7.2: Missing HTTP(s) headers

We see some missing headers, of which none are really dangerous, the most important headers (same-origin-policy, x-frame-options, content-source-policy and anti-xss header) seem to all be implemented, as can be expected from an organisation like Radboud. This makes it even nicer that we have found vulnerabilities using Besieger, there is a low vulnerability inside the X-Content-Type-Options header because the value is set to an in-legitimate value. Our second module also finds a low vulnerability:

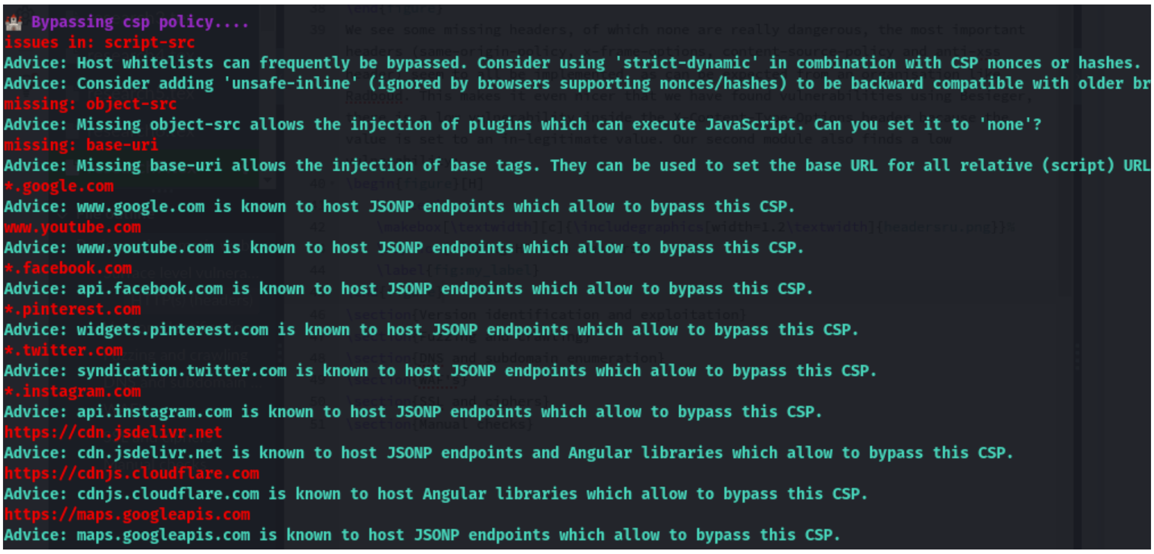

Figure 7.3: CSP can be bypassed

The Content Source Policy (CSP) of the Radboud can be bypassed in quite a few ways! It contains a few links not set strict enough, allowing bypassing via JSONP endpoints and it also misses an object-src and base-uri.

7.1.2 Cookie vulnerabilities

Besieger currently contains two cookie audit modules, one is manual and external and the other one is automated. Our first module is part of our Manual check library, which we will discuss in depth later. It automatically shows results from some websites that audit the input website on cookie privacy and compliance laws. The second module was custom coded and checks for the presence of the secure flag, the samesite flag and the httponly flag. We also check for the presence of session cookies and session ids, and adjust the height of the vulnerability based on this information. Let us show some of the results of this module on the website of a Brasilian University: ufsc.br.

Figure 7.4: Insecure session cookies

This example website uses highly insecure session cookies, we use http-cookiejar to detect cookies and session cookies. In this case the name also gives away that this is ID for a PHP session. This particular cookie has none of the flags set that should be used to properly protect the cookie.

7.2 Version identification and exploitation

The process of detecting software versions and identifying potential vulnerabilities requires the use of multiple tools and techniques [53]. One such tool is Wappalyzer, which specializes in fingerprinting technologies and employs a combination of regular expressions and dynamic detections to accurately identify software components and their versions [63]. Another tool, Nuclei, is a fast and customizable vulnerability scanner that uses templates to identify software versions and potential vulnerabilities [51]. Additionally, Nmap, a widely-used network scanning tool, is employed for port fingerprinting, which can help uncover vulnerable services running on target systems [44]. Web scraping techniques are also used on websites like webtechsurvey.com, employing the Selenium framework to gather information about software components [56].
By combining the fingerprints obtained from these tools, it is possible to cross-reference them against known vulnerability databases, such as the National Vulnerability Database (NVD) [50], searchsploit [55], and vulners [62]. This process often uncovers numerous vulnerabilities that might be missed by other scanners, increasing the chances of identifying exploitable software components [53].

7.3 Fuzzing and crawling

Fuzzing and crawling are essential techniques for identifying security vulnerabilities in web applications and services [60]. Tools like Katana [37], Arjun [7], and dirsearch [14] are employed to identify and fuzz endpoints. These fuzzed tools are then used to check for various security issues, such as

SQL injection, cross-site scripting (XSS), and path traversal [52]. Moreover, this process can help identify potentially sensitive or configuration files that might be exposed [60].

7.4 DNS and subdomain enumeration

Automated investigation of DNS records is conducted to check for blacklists, misconfigurations, and settings such as SPF, DKIM, and DMARC [2]. If an open SMTP service is detected by Nmap, a submodule is used to audit it [44]. For subdomain enumeration, multiple tools like Amass [4] are combined to generate passive lists of subdomains, followed by active tools and wordlists to generate possible subdomains [16].
The results are filtered based on their HTTP response codes, and further checks for subdomain takeover and identification of secrets on all domains are performed [2]. These domains can also be used for further manual labor if necessary.

7.5 Web Application Firewalls (WAF)

To identify WAFs and attempt bypasses, multiple tools are used [**?**]. Techniques for bypassing WAFs include searching DNS history for the original IP behind the WAF and checking if the WAF does not support certain offered ciphers [8]. A benchmark of the WAF's performance is compared to a median to evaluate how effectively it stops malicious queries [39].

7.6 SSL and ciphers

SSL and cipher configurations are inspected for potential misconfigurations and weaknesses [54]. This includes checking for outdated protocols like SSL 2.0/3.0 and TLS 1.0/1.1, as well as weak ciphers, server bugs, and lack of forward secrecy [46]. Additionally, normal checks for TLS/SSL configurations are performed to ensure the security and robustness of the encrypted communication [54].

7.7 Other attacks

Besieger uses a wide range of other attacks, such as a manual tour around Censys, Shodan and various google dorks. It also has specific attacks on Drupal and Wordpress, checks for secrets, tries http request smuggling, deserialisation attacks, header poisoning, CSRF and more. However for brevity not all features are included.

Chapter 8

Scan results on real-life environments

In this chapter, we apply some of the systems and methodologies that were designed onto real life organisations that have allowed us to scan their environment and to take part in this research as anonymous companies. We are going to find vulnerabilities and apply our vulnerability cycle onto the found vulnerabilities in this chapter. The companies have been chosen on size, thinking it would be beneficial for tests to scan bigger organisations, because that is more likely the place where the tools are going to be used. The companies have been chosen without prior knowledge of the security standard of the environment concerned.

8.1 Active Directory

We have scanned the Active Directory environment of a company that has more then 900 users, it has 54 domain admins and has been used for several years. Apart from the number of users, we didn't know anything about this before scanning. Due to RSAT not being available, we might have missed some certificate based vulnerabilities. There where also minor bugs due to not being able to run directly on a DC, but nothing that should have impacted scanning to much.

8.1.1 Main Report

After running our vulnerability scan system on one of the AD servers in the network we extracted the generated files and inserted them inside of their css/js container. We obtained the following Fortifier report:

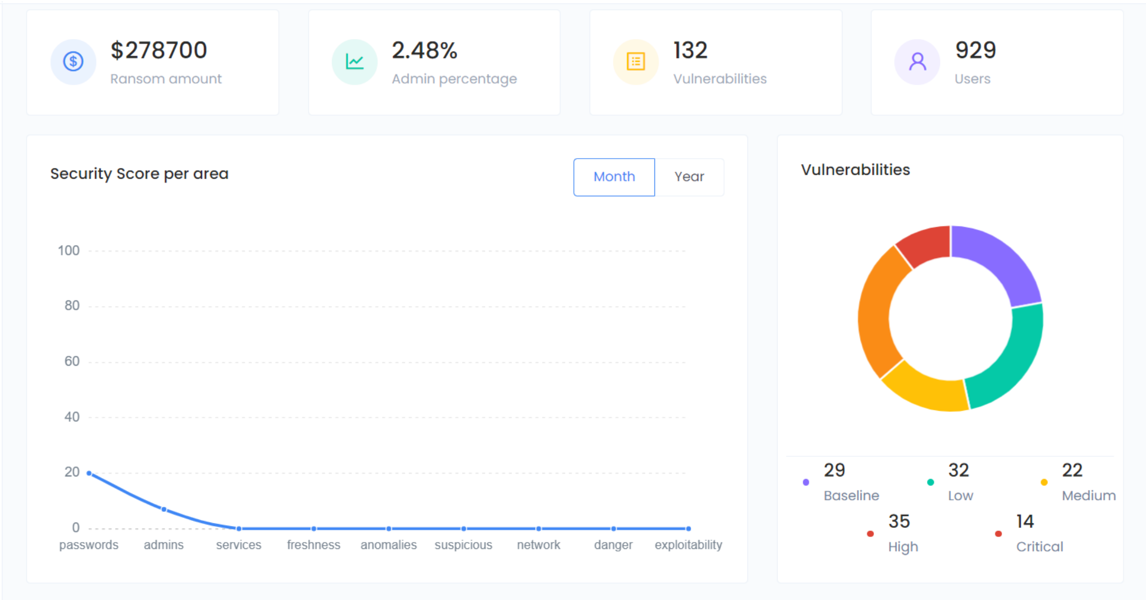

Figure 8.1: Fortifier AD Report

Results show that we have found 132 vulnerabilities, of which 14 are critical and 35 are high. The organisation scores 0/100 in almost every category we grade. The exception is password security, as a okayish password policy with minimum length = 8, complexity required and a lockout threshold of 5 was implemented. The score for password security is still only 20 due to some critical vulnerabilities we discuss later on. Admin security scores 7 out of a possible 100.

8.1.2 Management summary

We will take a look at some of the various vulnerabilities presented to give a complete view of the output fortifier generates, the emphasis will be on the higher vulnerabilities, as they often have more impact and usability, but we will also display some of the lower rated vulnerabilities identified as Fortifier aims to be fairly complete.

Let us start by taking a look at the top 7 vulnerabilities identified:

Vulnerability	Occurences	Category	Severity
There are windows servers 2003 on this network	9	Freshness	100
There are windows servers 2000 on this network	5	Freshness	100
Accounts with no password set	1	Anomalies	99
ESC6 - EDITF_ATTRIBUTESUBJECTALTNAME2	1	Anomalies	97
There are windows servers 2008 on this network	30	Freshness	96
Windows has not been updated in more than 120 days	1	Services	95
There are windows xp computers on this network	1	Freshness	95

Figure 8.2: Fortifier top 7

Some serious vulnerabilities can be found. Windows server 2008 stopped being securely supported by Microsoft in 2020, which means it has been pretty insecure for approximately 3 years at the time of writing. According to Nessus it is already a 10/10 vulnerability on the CVSS3 scale, windows server 2000 and 2003 are even worse. In addition to this, approximately 30 accounts were found that had no password set. There also was a certificate that had a very bad parameter that allows arbitrary users to upgrade themselves to domain admin via the creation of malicious certificates. There were also windows XP computers found that were doing critical tasks, namely being an essential factor of an entry pass generation system. In addition, various servers and domain controllers missed for 167 days of security patches by Microsoft. The vulnerabilities found point the overview of an environment that once was way bigger then 900 users, with the amount of servers on the network.

Now let us introduce another 7 very highly rated vulnerabilities:

Accounts vulnerable to the Kerberoasting attack	1	Anomalies	92
54 domain admins found	54	Admins	91
46% of domain admins is inactive	1	Admins	90
Accounts with passwords in dictionary	1	Anomalies	90
SSL 3.0 used for internal servers	1	Anomalies	86
Domain admins have delegation rights	42	Admins	85
Check the use of Kerberos with weak encryption (DES algorithm)	1	Freshness	85

Figure 8.3: 7 more high rated vulnerabilities

The first vulnerability found is about kerberroasting, which allows attackers to try and get the cleartext passwords of privileged domain users. Because a certain value is set, the account is not protected by the lockout password policy and hence the password can be bruteforced on the AD and therefore found if relatively weak. There were 4 admin accounts that were vulnerable to this, with three of them having a weak password, reflecting the critical value the vulnerability got assigned.
The second vulnerability is about the amount of domain admins, every domain admins (as demonstrated by the kerberroasting above) is a way to pwn the entire domain, therefore there never should exist so many admins.
The third vulnerability is the high amount of inactive admins, admin accounts that are inactive for a long period of time form an unneeded risk, amplified by the fact that they often belong to users who have left an organisation.
There were also passwords of users found in online lookups and dictionary, 42 users had their password leaked, including 3 administrators.
SSl 3.0 was used for internal communication by servers, as SSL 3.0 is long outdated and very vulnerable, this was also reported as high vulnerability.

Let us introduce a few more of the high vulnerabilities that were found:

There are 2 admins that have not logged in for 3 years	2	Freshness	83
There are windows windows 7 computers on this network	7	Freshness	80
Accounts with missing AES keys	1	Anomalies	80
LM hashes present	32	Freshness	76
Admins not in protected users group	39	Admins	75
Anonymous Access Enabled	1	Anomalies	75
Local admin password is very old	1	Admins	74

Figure 8.4: 7 more high rated vulnerabilities

Administrators with passwords older than 6 years	1	Freshness	73
A user can add devices to the domain	1	Anomalies	71
There are 9 admins that have not logged in for 180 days	9	Freshness	70
NLTMAuthentication Not Disabled	1	Passwords	70
Tombstone lifetime is too small	1	Anomalies	70
LAPS not installed	1	Service	740
Insecure DNS Zone [redacted]	1	Service	70

Figure 8.5: 7 more high rated vulnerabilities

Let us also show some lower vulnerabilities that are still highly usable for remediation:

Users with DCSync rights	2	Services	69
Pre 2000 group is not empty	1	Services	66
Accounts with the same password	1	Anomalies	65
There are more than 30% users that have not logged in for 180 days	38.4284176533907	Freshness	65
WSUS configuration using HTTP instead of HTTPS	10	Network	65
Administrators with passwords older than 3 years	6	Freshness	65
Exchange can control DNSAdmins	1	Anomalies	61

Figure 8.6: 7 randomly chosen medium vulnerabilities

Domain Admins group is not empty	42	Admins	35
DCS have differing operating systems	2	Freshness	30
There are users with passwords never set	18	Freshness	30
Disabled admins not in protected users group	3	Admins	25
Client failback is disabled for the Netlogon and SYSVOL folders on a domain controller.	1	Services	22

Figure 8.7: Some low/baseline vulnerabilities

Chapter 9

Related Work

Currently, there are a number of approaches to improve the security of AD. These include manual processes such as setting up Group Policies and Security Policies, as well as various tools, such as Microsoft's Security Compliance Manager and Microsoft's Advanced Group Policy Management. These tools can be used to detect and address security vulnerabilities in AD, but they are labor-intensive and require extensive expertise.

In addition, there are a number of open-source and third-party solutions available for AD security, such as Splunk, Rapid7, AlienVault, and Qualys. These solutions provide automated vulnerability scanning and reporting, but they do not provide a comprehensive solution for the entire security improvement cycle.

The current accessible state of the art tools are PingCastle and Purple Knight. These tools offer a wide range of features, such as detailed reporting and analysis, but they are limited to manually-defined security policies and do not provide a comprehensive solution for the entire security improvement cycle. Most tools get stuck on the first two steps of the FRIS system. They only Find and Report vulnerabilities. These tools, however, are excellent for finding a big amount of vulnerabilities in a short amount of time. We improve their State of the art approach and find even more vulnerabilities.

Chapter 10

Discussions

10.1 Discussion points

In this study, we proposed the Stronghold system, a novel approach to automating the security improvement cycle for both Microsoft and Google environments. Our primary aim was to develop tools that could effectively address vulnerabilities in Active Directory, Azure Active Directory, Google Workspace, and Google Cloud while also providing automatic auditing, phishing simulations, and remediation capabilities. Although we had ambitious goals, we were able to achieve significant success in several aspects of our research.

10.1.1 Interpretation of results

The Stronghold system demonstrated state-of-the-art results in vulnerability detection and remediation. Furthermore, we introduced the FRIS pipeline, which provided valuable information, further research, and direct solutions for addressing the discovered vulnerabilities. These results highlight the potential of the Stronghold system to effectively combat security threats in corporate environments, particularly in scenarios where budget constraints often lead to the negligence of security issues.

10.1.2 Comparison with previous research

Compared to existing solutions, the Stronghold system offers a more comprehensive approach to security improvement. By targeting both Microsoft and Google environments and incorporating a seamless process from vulnerability detection to remediation, our system offers a unique solution to address the ever-evolving cybersecurity landscape. Additionally, the focus on usability and speed sets our system apart from other state-of-the-art solutions, making it more accessible and cost-effective for organizations to implement and maintain.

10.1.3 Limitations and future research

Despite our successes, there were limitations to our study. We acknowledge that improvements can be made in the Web, Azure, and Google components of the Stronghold system. Moreover, we aim to surpass the current state-of-the-art in even more benchmarks in future research. It is also essential to continually update and refine our system to keep up with the rapidly changing cybersecurity landscape.

10.1.4 Implications

The Stronghold system has significant implications for the security of corporate environments utilizing Microsoft and Google platforms. By providing an effective, user-friendly, and cost-efficient solution to identify and address vulnerabilities, organizations can better protect their digital assets and reduce the risk of cyberattacks. Furthermore, the development of the FRIS pipeline offers valuable resources for IT professionals to research and remediate detected vulnerabilities, further strengthening the overall security posture of their organizations.

10.2 Future work

- Create toolset for automatic wifi pentesting to be included in our internal network scan.
- WhistleBlower, automated scanning, automated mail notifications, automated discovery of new subdomains. Add compability to make Fortifier a passive scanning tool that keeps evaluating for possible security holes.
- During this research we created a basic Azure scanner which didn't yet scan enough to be labeled STOTA, we plan to extend this and set a new STOTA in Azure vulnerability scanning.
- Automatic generation of 3D networks with all the sub-fields we audit for inside the main section of our automatic generated report. This would allow for more comprehensive and detailed visualization of the networks being tested and the sub-scores of departments.
- A segmentation graph generator that generates graphs of networks audited and the segmentation in them based on simple inputs. This would provide a clear overview of the different segments and sub-networks within a network.
- Beekeeper: a custom system to add honeypots to environments and websites and to trigger alerts from these. This would allow for the

detection of potential attacks and provide valuable information about the techniques and methods used by attackers.

- Librarian: an automatic toolset for source code analysis. This would enable automated scanning of source code for vulnerabilities and potential security issues, making it easier to identify and fix any issues before deployment.

- Rumourwatcher: an employee OSINT model that gets info about employees from a number of sources and generates password lists from these.

- Physical pentesting possibilities and tools. This would involve testing the physical security of a facility, such as access controls, cameras, and alarms, to identify any vulnerabilities and to ensure the safety of employees and assets.

- Pentesting with reinforcement learning. This would involve using machine learning algorithms to improve the efficiency and effectiveness of pentesting by automating the process and learning from previous pentesting results.

- Language model trained on pentesting/hacking books and info. This would be a language model that has been trained on large amounts of pentesting and hacking information to assist in identifying vulnerabilities, generating attack scenarios and more.

Chapter 11

Conclusions

In conclusion, our research demonstrates the potential of the Stronghold system to address the ongoing security improvement cycle in both Microsoft and Google environments. By automating vulnerability detection and remediation and providing a comprehensive solution for IT professionals, the Stronghold system offers a promising approach to enhancing cybersecurity in corporate settings. Future research should focus on refining the system's components and surpassing current state-of-the-art benchmarks to ensure that the Stronghold system remains effective and relevant in the ever-evolving world of cybersecurity. It should extend and improve to set a benchmark that surpasses the STOTA performance in all of its categories.

Bibliography

[1] A. Acquisti and J. Grossklags. *Security and Decision Making*, pages 10–17. IGI Global, 2012.

[2] Mohammad Al-Duwailah, Ahmad Al-Hammouri, and Khaled Taha. Comprehensive dns-based enumeration and analysis of domains. In *2018 IEEE Symposium on Computers and Communications (ISCC)*, pages 00134–00139. IEEE, 2018.

[3] R. Allen. *Windows Server 2003 Active Directory Design and Implementation: Creating, Migrating, and Merging Networks.* Packt Publishing Ltd., 2006.

[4] Amass. Owasp amass, 2021.

[5] J. Andress. *The basics of information security: Understanding the fundamentals of InfoSec in theory and practice.* Syngress, 2014.

[6] APWG. Phishing activity trends report, 2018.

[7] Arjun. Arjun: Http parameter discovery suite, 2021.

[8] Mario Heiderich Barnett, Tobias Holz, Pravir Chandra Mehta, Aashish Nukala, and Carlos Rios. *Web Application Obfuscation.* Syngress, 2011.

[9] R. Baskerville. Information security governance: Toward a framework for action. *Information Systems Management*, 21(3):36–47, 2004.

[10] Robert F. Bruner. Repetition is the first principle of all learning, 2001.

[11] Semperis B.V. Purple castle, 2022. Tool to improve AD security `https://www.purple-knight.com/`.

[12] B. Desmond, J. Richards, R. Allen, and A.G. Lowe-Norris. *Active Directory: Designing, Deploying, and Running Active Directory.* O'Reilly Media, Inc., 2008.

[13] G. Dhillon. *Principles of information systems security: text and cases.* John Wiley Sons, 2006.

[14] Dirsearch. Dirsearch: Web path scanner, 2021.

[15] Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. feb 2020. Submitted on 15 Feb 2020.

[16] Zakir Durumeric, David Adrian, Ariana Mirian, and J Alex Halderman. A search engine backed by internet-wide scanning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 542–553, 2015.

[17] Google Employee. Personal communication, 2021.

[18] FBI. 2017 internet crime report, 2017.

[19] R. Gandhi, A. Sharma, W. Mahoney, W. Sousan, Q. Zhu, and P. A. Laplante. Dimensions of cyber-attacks: Cultural, social, economic, and political. *IEEE Technology and Society Magazine*, 33(1):28–38, 2014.

[20] S. Garfinkel. Email-based identification and authentication: An alternative to pki? *IEEE Security Privacy*, 3:20–26, 2005.

[21] Gartner. Market guide for security awareness computer-based training, 2018.

[22] Google. 2-step verification, n.d.

[23] Google. Authenticate email with spf, dkim, and dmarc, n.d.

[24] Google. Context-aware access, n.d.

[25] Google. Control who can access google services, n.d.

[26] Google. Data loss prevention (dlp) for google workspace, n.d.

[27] Google. Encryption in transit in google cloud, n.d.

[28] Google. Google workspace (formerly g suite), n.d.

[29] Google. Machine learning in gmail to block sneaky spam, n.d.

[30] Google. Manage user access, n.d.

[31] Google. Password security, n.d.

[32] Google. Set session lengths for google services, n.d.

[33] R. Grimes. Mastering identity with azure active directory. In *Proceedings of the 2016 IEEE Systems and Information Engineering Design Symposium (SIEDS)*, pages 1–6, 2016.

[34] D. Horn. Protecting your enterprise with azure active directory and windows 10. In *Proceedings of the 2017 IEEE Military Communications Conference (MILCOM)*, pages 497–502, 2017.

[35] Ponemon Institute. 2018 cost of a data breach study: Global overview, 2018.

[36] M.E. Kabay. *Security Awareness: Concepts and Practices.* John Wiley Sons, 2001.

[37] Katana. Katana: A python tool for google hacking, 2021.

[38] A. Kok. Multi-factor authentication: A survey. In *Proceedings of the 2018 IEEE Conference on Dependable and Secure Computing (DSC)*, pages 1–8, 2018.

[39] Ronald L Krutz and Russell Dean Vines. *Cloud Security: A Comprehensive Guide to Secure Cloud Computing.* Wiley Publishing, 2010.

[40] Lendita Kryeziu. Learning from errors. *ILIRIA International Review*, 5(1):393, 2015.

[41] Vincent le Toux. Ping castle, 2022. Tool to improve AD security `https://github.com/vletoux/pingcastle`.

[42] C. Liu and P. Albitz. *DNS and BIND.* O'Reilly Media, Inc., 2006.

[43] J. Lowensohn. Microsoft office 365, google apps go head to head. `https://www.cnet.com/news/microsoft-office-365-google-apps-go-head-to-head/`, 2011.

[44] Gordon Fyodor Lyon. *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning.* Nmap Project, 2009.

[45] Rebecca A. Maynard, Rebecca N. Baelen, Phomdaen Souvanna, et al. Using iterative experimentation to accelerate program improvement: A case example. *Educational Evaluation and Policy Analysis*, 46(5), 2020. First published online May 28, 2020.

[46] Christopher Meyer and Jörg Schwenk. Sok: Lessons learned from ssl/tls attacks. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 189–203. Springer, 2013.

[47] M. Minasi and C. Rice. *Mastering Windows Server 2003.* Sybex, 2004.

[48] K.D. Mitnick and W.L. Simon. *The Art of Deception: Controlling the Human Element of Security.* John Wiley Sons, 2002.

[49] C. Neuman and T. Ts'o. Kerberos: An authentication service for computer networks. *IEEE Communications Magazine*, 32:33–38, 2011.

[50] NIST. National vulnerability database, 2021.

[51] Nuclei. Nuclei - fast and customizable vulnerability scanner, 2021.

[52] OWASP. Owasp top ten project, 2021.

[53] Andriy Panchenko, Fabian Lanze, and Igor Ponce-Alcaide. Website fingerprinting at internet scale. In *Proceedings of the 23rd Annual Network and Distributed System Security Symposium (NDSS)*, 2016.

[54] Ivan Ristic. *Bulletproof SSL and TLS: Understanding and Deploying SSL/TLS and PKI to Secure Servers and Web Applications.* Feisty Duck, 2013.

[55] Offensive Security. Searchsploit manual, 2021.

[56] Selenium. Selenium webdriver, 2021.

[57] R. Sidelnikov. Active directory security: Best practices and vulnerabilities. In *Proceedings of the 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, pages 1040–1045, 2018.

[58] W.R. Stanek. *Windows Group Policy: Windows Server 2008 and Windows Vista Resource Kit.* Microsoft Press, 2009.

[59] D. Sullivan. Google apps for your domain: The good & the bad. `https://searchengineland.com/google-apps-for-your-domain-the-good-the-bad-12063`, 2007.

[60] Michael Sutton, Adam Greene, and Pedram Amini. *Fuzzing: Brute Force Vulnerability Discovery.* Addison-Wesley Professional, 2007.

[61] Symantec. 2018 internet security threat report, 2018.

[62] Vulners. Vulners: Vulnerability data base, 2021.

[63] Wappalyzer. Wappalyzer: Identify technologies on websites, 2021.

[64] M. E. Whitman and H. J. Mattord. *Principles of information security.* Cengage Learning, 2011.

[65] Emma J. Williams, Joanne Hinds, and Adam N. Joinson. Exploring susceptibility to phishing in the workplace. *International Journal of Human-Computer Studies*, 120:1–13, 2018.