Design and develop automated detection of three common broken authentication vulnerabilities

Submitted by

S.M. Towhidul Islam

181-35-2435

Department of Software Engineering

Daffodil International University

Supervised by

Mr. Md. Maruf Hassan

Associate Professor

Department of Software Engineering

Daffodil International University

This Project report has been submitted in fulfilment of the requirements for the Degree of Bachelor of Science in Software Engineering.

APPROVAL (Room- 610)

This thesis titled on "**Design and Develop Automated Detection of Three Common Broken Authentication Vulnerabilities**", submitted by **S. M. Towhidul Islam (ID: 181-35-2435)** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering and approval as to its style and contents.

BOARD OF EXAMINERS

---------- **Chairman**

Dr. Imran Mahmud
Head and Associate Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

---------- **Internal Examiner 1**

Md. Shohel Arman
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

---------- **Internal Examiner 2**

Khalid Been Badruzzaman Biplob
Lecturer (Senior)
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

---------- **External Examiner**

Md. Tanvir Quader
Senior Software Engineer
Technology Team

DECLARATION

It's been declared that this thesis including all the research-based experimental works has been completed by me under the supervision of Md. Maruf Hassan (Associate Professor). Department of Software Engineering of Daffodil International University.

I also declare that neither this thesis nor any part of this whole research-based experiment has been submitted elsewhere for the award of any degree.

S.M. Towhidul Islam

181-35-2435

Department of Software Engineering

Daffodil International University

MD. Maruf Hassan

Associate Professor

Department of Software Engineering

Daffodil International University

ACKNOWLEDGEMENT

We are very much grateful to our Daffodil International University helping us to complete this Bachelor of Science study with this thesis. We are very grateful to our supervisor Mr. Md. Maruf Hassan, Associate Professor, Department of Software Engineering, Daffodil International University. As well as we would like to thank Dr. Imran Mahmud, Associate Professor & Head in Charge of the Department of Software Engineering, Daffodil International University for motivating us for quality research, also we want to thank our teachers who has been continuously supporting us throughout this undergrad. Behind the thesis, we had to deal with implementation issues and the encouragement and guidance from our supervising teacher at these times was truly unparalleled. His excellent scientific advice and encouragement, as well as his enthusiastic supervision, made the completion of this thesis possible. This would not have been possible without his unfailing collaboration. Many thanks to my classmates who supported and encouraged me throughout this work. Surely, and above all, our hearts are grateful for this blessing from Almighty Allah. And a lot of respect to our parents for their sincere support.

Abstract

The purpose of this thesis is to present a new tool for detecting common web attacks that lead to web application information disclosure, primarily through improper authentication and session management. It provides a flexible URL search engine that scans HTTP requests and responses during web page delivery and records the necessary data without impacting web server performance. New tools can detect attacks using HTTP responses such as Post and Get methods. And by investigating all factors, we are looking for satisfactory results. The new tools are highly extensible, allowing for future work. Web applications are consistently used on a consistent schedule. Web applications are experiencing security risks and breaches these days. Security analysts, companies, and organizations are working together to stop, or at least mitigate, these attacks and dangers. The Open Web Application Security Project (OWASP) is a non-profit security association that discovered and ordered ten attacks against vulnerabilities affecting her web applications today. Suspended reviews and executive weakness attacks are the next top attacks in the report listed in OWASP. This white thesis describes flawed authentication and session management exploits and their detection process. We also propose an automated system to detect vulnerability attacks such as brute force, session ID rotation after successful login, and session ID disclosure in URLs by exposing all the facts.

Keywords: Broken Authentication, Brute force, Session ID, Rotation, Log in, Exposes URL.

Table of Contents

Table of Figures

CHAPTER 1
INTRODUCTION

1.1 Background

Fundamentally, flawed authentication refers to weaknesses or flaws inherent in web-based platforms or applications that allow programmers to bypass login security and get close enough to each of the accolades claimed by hacked clients. to Authentication ensures that validated clients can access data and respect web applications (S. Hossain & Mahmud, 2018). If an attacker bypasses the cycle and impersonates the client in your application, you are "broken". The inherent flaws mentioned above can be broadly classified into two classes. Specifically, unfortunate session management and unfortunate Credential management (Alahmad et al., 2022). The shortcomings of session management should be recognized by examining how online authentication and browsing generally works. All communications a customer has with an organization, through virtual entertainment sites or web-based betting gateways, are recorded and saved for web sessions that can be tracked in the web application used (Ami & Malav, 2013). A web application provides a session ID to the client for each visit. This identity is essential so that the application can communicate with the client and respond to requests. The Open Web Application Security Task (OWASP) proposal for flawed authentication seems to state that the session ID given to a logged-in client can be easily compared to the client's unique login credentials. Additionally, it can be used to impersonate the client on your application without too much effort (S. Hossain & Mahmud, 2018). Such session IDs should be carefully expired like this. Defects and escape clauses may be controlled by the programmer (S. Hossain & Mahmud, 2018).

Essential customer certifications may also be obtained or forfeited upon application. Credential management is therefore paramount to cybersecurity. As an exception, web applications should not allow regular or simple passwords such as "password1" or "pass123". Allowing the use of such passwords weakens authentication controls. A failure of a web application to protect its clients from programmers who force training with stolen or hacked passwords is a form of authentication failure.

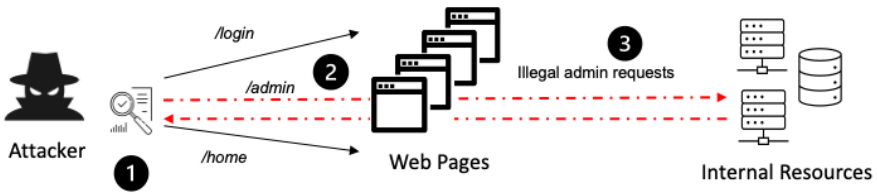

Figure 1: Broken Access Control Example

Fragmented verification attacks have accounted for a fair share of the most horrific information breaches of recent times, and security experts agree with their warnings about this underestimated danger. OWASP remembers it in its outline of the "Main 10" of the biggest web application security possibilities around 2017. By 2020, Broken Authentication had risen to his No. 2 spot. In this thesis, we analyzed some of broken authentication and session management vulnerability attack. Exams cover types, models, and how to detect what's broken authentication and session management attacks in automated controls. We are proposing a system of detecting these given attacks:

1.1.1 Brute Force Attack:

A brute force attack is an experimentation method used to get to the bottom of sensitive information. The maximum broadly diagnosed packages for brute force attacks are breaking passwords and breaking encryption keys (keep to peruse to look at encryption keys). Other ordinary focuses for animal electricity attacks are API keys and SSH logins (Najafabadi et al., 2015). Brute force mystery key attacks are regularly finished via way of means of contents or bots that target a site`s login page (Najafabadi et al., 2014).

What separates brute force attacks from different breaking strategies is that beast pressure attacks do not make use of a scholarly system (Hofstede et al., 2017); they simply have a cross at utilizing diverse mixes of characters till the proper combo is found. This is just like a crook trying to interrupt right into a mixture included via way of means of endeavoring every plausible blend of numbers till the included opens.

1.1.2 Doesn't Rotate Session ID After Log in:

Once the user is "logged in", the application should transform the session ID in a way that an attacker can do. If we knew or set the user's session id (Dabirsiaghi et al., 2009), we don't know the new session id at all, so we have no resources Introduce ourselves to the application her server as a stakeholder. Execution of "Rotate" We should invalidate the old session and run another session with the goal of not containing any explicit information about the victim Each session except a new secure session that runs after we answer the prompt Contains valid certification (Dabirsiaghi et al., 2009). This feature requires a way for the user to inspect certain activities as security restrictions. Very Usually this is in the setup record where the activity is flagged. such a review It may allow the application to perform other security activities.

1.1.3 Session ID disclosure in URLs:

Sensitive data in URLs can be signed in many places, including client programs, web servers, and transfer or intermediary servers between two endpoints. The URL can also be displayed on screen, bookmarked, or sent from the client. If offsite participation is pursued, they may be revealed to outsiders via the referrer header. Injecting a session token into a URL increases the risk of being caught by an attacker (Session token in URL, n.d.).

1.2 Motivation of research

OWASP is a non-profit organization aiming to contribute to programming security. Stay safe from the top 10 web application security threats. According to OWASP, Web applications have been tested for some kind of broken access control, that is why Detection of this is very important.

This vulnerability can definitely be detected using white box detection. white box code Explode to see if there are direct references to internal elements in the framework. On the other hand, in applications it is very interesting to distinguish this type of weakness without any code. Therefore, we should examine the response of every application request. Sent from the analyzer. this is a tedious effort.

There are many detection methods of these vulnerabilities. But aren't any singular method of detecting them together at a time. The goal of the thesis is to build a tool which will detect these vulnerabilities automatically in a certain time and it is also a black box testing tool.

1.3 Problem Statement

It's normal to receive an email from a professional co-op notifying you that someone from an illegal area has entered your file. By the time this happens, you may have fallen prey to a brutal power attack. In such a situation, it is advisable to change the private key immediately. In case you fall victim to a brutal undetected or unreported force attack, you can try to keep your private key in your secret records changing. If you are an executive in an organization, site security and customers should be extra vigilant for signs of Beast Force attacks, especially high volume attacks (Najafabadi et al., 2014). Many failed logins may have come from distracted clients, but your site could take a hit (Hofstede et al., 2017). The session class session ID infrastructure was specifically designed to be generic for session ID pivots. This means that if there are multiple simultaneous posts like log in successfully to a similar session and then changes ID after log in, the previous session will be lost. We would expect long response times in wearable environments using layer engineering (Dabirsiaghi et al., 2009). Setting it to disabled resets it to the default to another session ID. It's okay to extend the session pivot time (Dabirsiaghi et al., 2009). The default settings are usually fine. Coupled with encryption and other security measures, session bonuses are well protected against session hijacking. Fundamentally, session identity pivots under stateless multi-access conditions is always a problem and basically impossible to avoid. Sessions imply values stored on the server that are well-defined for individual clients of the application (Dabirsiaghi et al., 2009). This is important for two reasons. First, HTTP is a stateless convention. Each request is discreet and contains no information about previous or subsequent requests. Meetings help the server track the sender of a request. Otherwise you'll have to log in every time you tap a button or connect. The second reason for the sessions is user approval. A session ID can be used to recognize a specific client with explicit privileges within the framework. The application knows who the person is and what they are allowed to do. Session He consists of two parts. A server-side information store stores meeting IDs and passes them data about customers, such as: B. His customer ID or track data. A similar meeting ID will be sent as a reward from the program. Treats are placed in the customer's frame by the program. A client passes a treat with each request to let the server know that this request is from a similar client (Session token in URL, n.d.). Most applications use meetings to track customers before and after reviews. Proper session of the panel is essential to the security of the application. A legitimate meeting ID has a similar level of trust to a username/secret word or second factor verification token. There are many research works available of these vulnerabilities individually, which are a bit complex and also time consuming, because of implementing non-automated methods. And This thesis proposes a system that will detect these vulnerabilities in one place with a time saving way also in an easy to understanding way.

1.4 Research Question

How to detect brute Force?

How to detect if session ID is getting exposed in URL?

How to detect if session ID is rotated after log in or not?

How to detect log in page via crawling?

1.5 Research Objectives

The intent and goals of this thesis are:

- To reduce the chances of broken access control breaches by detecting brute force, session ID exposed in URL, Session ID rotation After Successful Log in by.
- To Creating a singular framework to detect all three automatically.
- To Save time when scanning.

1.6 Research Scope

Brute Force, Session ID Revealed in URLs, Session ID Revolution After a valid login, these three vulnerabilities are still a significant threat to web applications. Timely detection of these vulnerabilities is critical, according to OWASP. Otherwise, it can lead to serious problems. It's disturbing that 94% of web applications are always trying to find some kind of broken access control vulnerability. The moderate incidence of this admission control violation is 3.81% (OWASP Top 10). These three vulnerabilities have become perhaps the biggest threat to web applications. This attack is also part of an access control breach. There are not many techniques accessible on the site, but there are no programmed instruments. For this reason, this localization technique is the most requested means of testing web applications for vulnerabilities.

1.7 Thesis Organization

This thesis is adjusted as follows:

Part 1 presents the rationale for the proposal, the inspiration for the consideration, the problem statement, the research question, the research goal, the extent of the research, and the hypothesis linkages, while the second part provides a point-by-point overview from previous research. and divided by spotlight and synopsis. Section 3 is divided into deployment sketching, requesting URLs, compiling information, extracting information, brute force checking, finding session IDs in URLs, and session ID confusion. After a valid login, also refer to Stream Burn to calculate the strategy. In Part 4, we divided execution sketches, data collection, attack creation modules, reporting modules, results, and test environments. In Part 5 you can see the contribution and limitations.

CHAPTER 2

LITERATURE REVIEW

2.1 Background

In fact, the program saves the client's credentials in the authentication handle, so the session continues to work even after it expires by sending all the data to the server side. Similarly, the specific duration of the session between the client and her web application his server should be determined by the framework chair. Session and certificate management are considered broken authentication because attackers can obtain session IDs or hijack client credentials. Authentication is broken when an attacker tries to contemplate the client's private key (Ami & Malav, 2013), the session token, and the client's identity in her data. Once the customer's entitlements have been matched and verified against the records in the record, a session is established between the customer with a particular special identity and her web application (Alahmad et al., 2022).

web crawlers gather and interaction the whole items in the Web in a unified area, so that it tends to be filed ahead of time to have the option to answer numerous client inquiries (Jose et al., 2016). In the beginning phase when the Web is still not extremely enormous, basic or arbitrary creeping strategy was enough to record the entire web. Be that as it may, after the Web has developed exceptionally enormous, a crawler can have huge inclusion in any case, seldom revive its slithers (Dhenakaran & Sambanthan, 2011), or a crawler can have great inclusion and quick revive rates however not have great positioning works or backing progressed question capacities that need seriously handling power (T. V. Udapure et al., 2014). Consequently, more development creeping techniques are required because of the restricted assets like time and organization transfer speed (Giles et al., 2010).

Brute-force was actually a game for PC released in 2000 (Dave, 2013). Brute Force was a third-person shooter and consisted of several characters. Each with their own strengths and capabilities. The aim was to find several other characters who were reliable to the union (Najafabadi et al., 2015). A team was formed named "brute-force team" to answer the union. Mission of the team was to find and fought with the aliens and armed force (Najafabadi et al., 2014). Whenever any of these aliens are seen, the members of the Brute force team battled with them. And if the attacker slightly modifies the dictionary words and perform attack is known as "Hybrid Brute-force attack" (Koch & Rodosek, n.d.). The attack is an attempt to determine a password by analytically trying every possible combination of letters, numbers, and symbols etc. If the attacker attacks on the basis of exact dictionary words, then it is known as "Dictionary attack" (Jablon, 2002). Many people wish to choose a meaningful dictionary word rather than selecting a random password.

HTTP cookies, which generally contain one or a small number of short identifier strings allowing a server to associate seemingly unrelated requests, rapidly became the dominant mechanism for web session management (R. Kumar & Goel, 2014). However, because these tokens are static and transmitted "in the clear", an adversary able to intercept them can use these cookies to gain unauthorized access to a user's session. During this secure session, the server generates cookies that the user can later employ as lightweight authentication tokens. Unfortunately, the use of cookies introduces a number of security risks, especially when they are employed as session authentication tokens (Modi, 2020).

Groups were your computer whenever you browse the Internet Supports many switches and server's world. The web can be used for many purposes, including: informal communication venues, online exchanges, the Internet shopping etc. So we have a consistent deal Data on the web means they are at risk Weaknesses too (M. Johns et al., 2011). then encouraged expansion digital fraud. Programmers are getting better and better I recently break into a framework. there are different varieties an attack carried out over the Internet by a programmer or attacker. Enemy seeks unauthorized control cause confusion, provide false information, coerce or access your personal data. high risk, the most common cybercriminal attack is session capture. Session commander ring is also called man-in-the-middle attack (Hans et al., 2013). A session commandeering attack is known as capture (Pan et al., 2016). Through his TCP/IP communication session without them Consent or Information. OTC creates single-use Authentication token with modified hash chain development. These tokens, if validated by the web Application, not reusable. Also, each OTC A certification is attached to a specific bid for an asset. i.e. enemies cannot be caught and reused Because you redirected the session illegally. HTTP is stateless by convention, requests to web servers are handled as follows (Hans et al.,

2013): Free exchange without contacting each other. This plan is simple and adaptable, but we will create for applications that require different relationships Exchange with a single user fair problem on site. Pervasive HTTP handling Allow one or several short identifier strings A server that quickly forwards seemingly unrelated requests Became the main system for web sessions management. Unfortunately, the use of treats presents itself Various security opportunities. Used as a session authentication token. As model, many sites have great strengths HTTPS-like systems Validate the client first. during this protected session Waiter creates treats for customers to use later (Bansal et al., 2013) as a lightweight authentication token. Anyway, given that These tokens are static and sent "for free" Enemies ready to catch them can use these treats Gain unauthorized access to a user's session (Bansal et al., 2013).

2.2 Summary

There are some research thesiss which has demonstrated these attacks, and some of them even created model of detecting these attacks. Such as brute force attack, it is a very common attack and to detect it there are some procedures we have to follow so these thesiss has showed various method and described how it can be detected with a model (Hossain et al., 2020) (Dave, 2013) (Najafabadi et al., 2015) (Jablon, 2002), Credential Stuffing, this attack is a very tough to detect and these thesis has shown some of methods and also enlightens all the information's about this attack. (Novotny et al., 2002) (Ke Coby Wang & Reiter, 2019) (K. C. Wang, 2021), URL Rewriting is also a common one which happens to be harmful attack, so the detection of these attack is very much important, to detect it we have to know it's nature, which these thesis has shown already with particular methods (Hans et al., 2013) (Pan et al., 2016), Session Fixation, this attack actually helps attacker to steal user session information, to detect this , there are various one which these thesis has demonstrated (Dawson, 2017) (R. Kumar & Goel, 2014) (Johns et al., 2011), Uses public unencrypted connections to send passwords, credentials, and session ID over them (Greenwood & Khan, 2014).

These thesis has been showing some of broken authentication attacks, collected total information about broken authentication from these thesiss (Alahmad et al., 2022) (S. Hossain & Mahmud, 2018) (Ami & Malav, 2013).

CHAPTER 3
RESEARCH METHODOLOGY

3.1 Solution Overview
In this thesis I have implemented a solution based on these three broken access Control vulnerabilities and to detect them automatically.

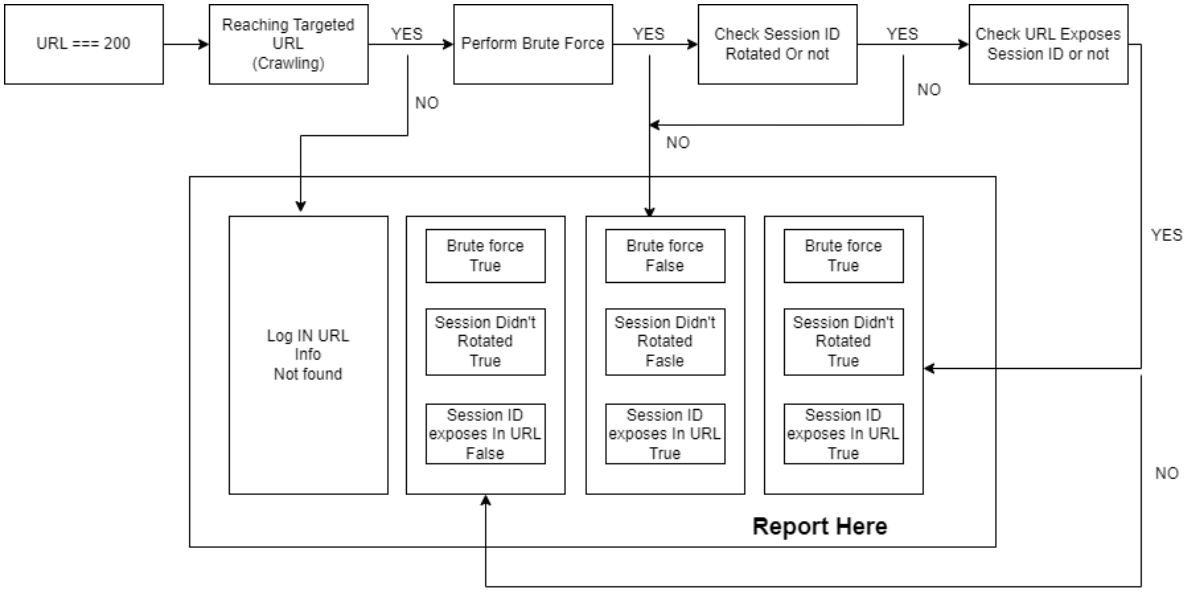

Figure 2: Diagram Of methodology

In this figure All measures are repeated, first we give an URL, if the URL is valid or not, if it is valid then it will start **CRAWLING,** it will do the task of finding log in field, where it performs the brute force, after finding the fields, it will perform **BRUTEFORCE** attack using the word list provided, then it will find the correct username and password by trying all the words, After Finding the credential it will log in automatically, and capture the session ID after log in, and compare it with the **session ID before log in**, And then It will check the URLS that they Exposes any **session ID in the URL** directly, and Generate a total report of the vulnerability is founded or not. Now each phases are described with details below:

3.2 URL:
Here we look for an URL to be given, then check that the URL is valid or not, if the URL is valid then it will proceed for the crawling phase.

3.3 Reaching Targeted URL(CRAWLING):
In the data collection area, collected data is completed with the help of automated Crawler (Dhenakaran & Sambanthan, 2011). I will handle it with a tingling system and a Beautiful soup caterpillar. Scrapy Structure is a fast and capable open source project. Separate data from huge pages. Beautifulsoup library good for parsing HTML components Already used (Udapure et al., 2014). First, the client needs to enter the base URL of her web application. Since the base

URL is first Signs of overall interaction. This is a fully programmed strategy, meaning no human experts requirement. First, the base URL is embedded in one line. This line is used to monitor which URLs are being visited, so entire interaction. Here we are satisfied with the URLs with log in page. Here we deal with blocks Overview of the programmed crawler framework (Jose et al., 2016).

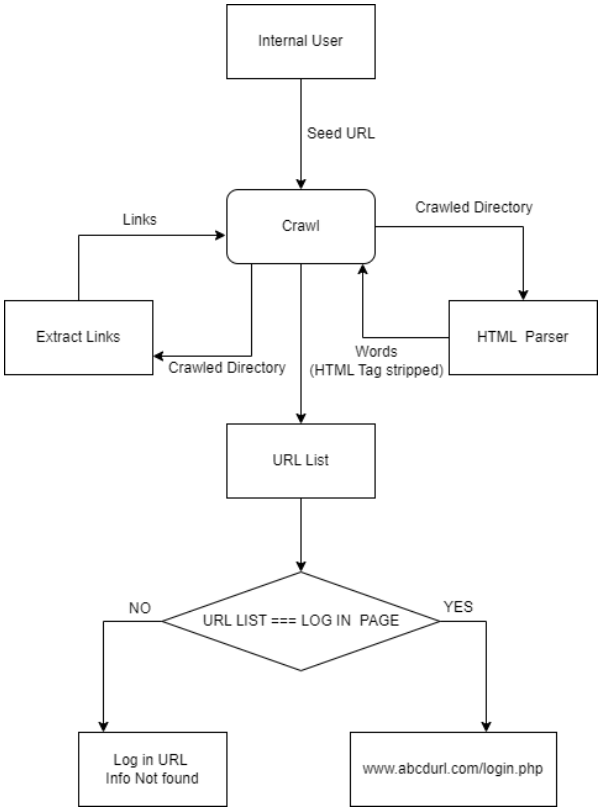

Figure 3: Diagram of Crawler

3.4 Perform Brute Force:

In this phase we find the input field from crawler, then we attempt various combinations, in this case we are using "**SecList**" which have a tons of word combinations, we divided it as two columns, as the first column is user name, the other one is password. These attacks are basic because many people actually use weak passwords. "Password 123" or "1234" or such unfortunate private key etiquette B. Using similar secret phrases on multiple websites. Passwords may also be guessed by programmers doing small surveillance tasks to decipher a person's supposed secret language, such as the name of a major gaming group. The detecting process is given below:

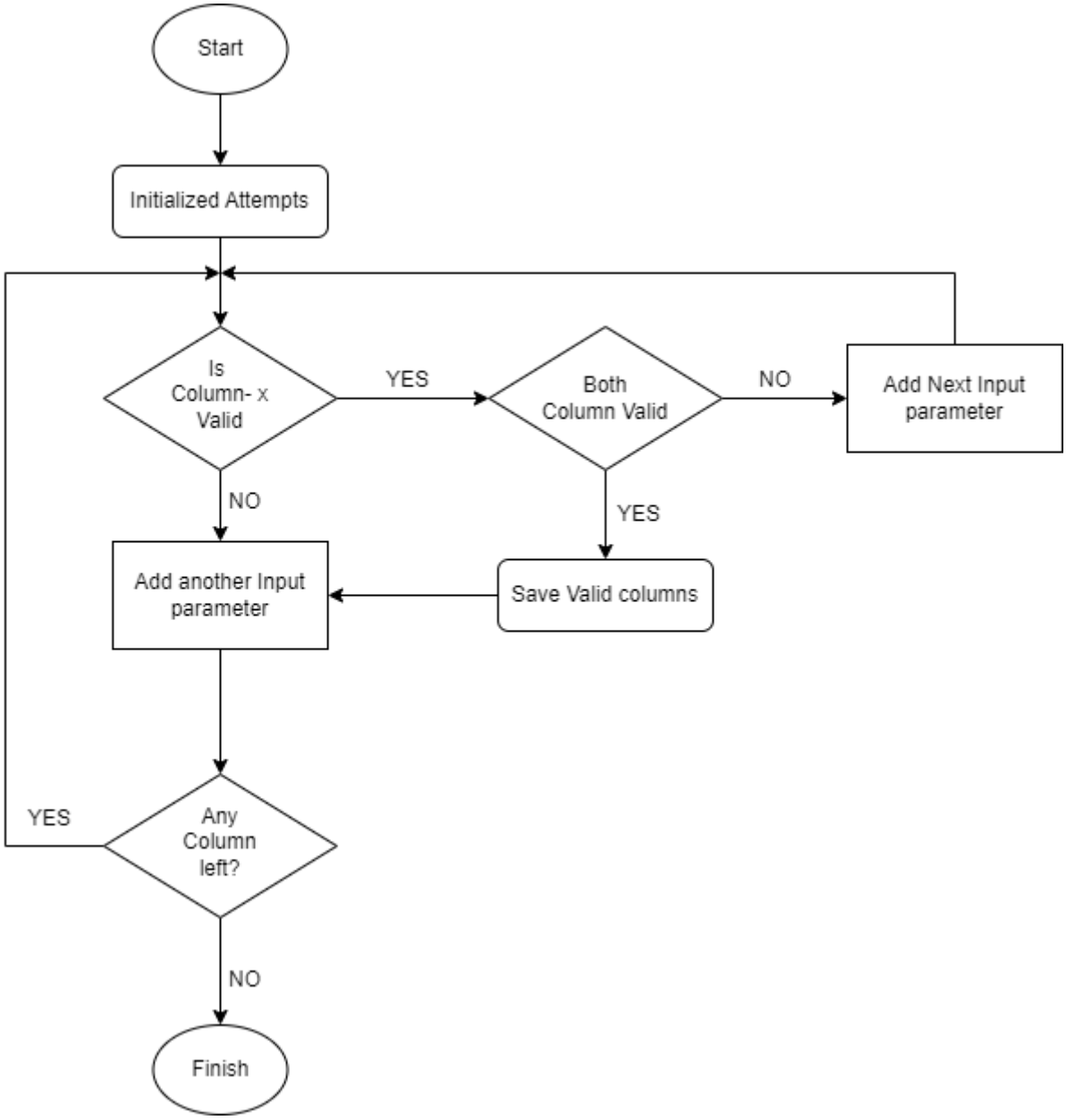

Figure 4: Diagram of Brute force Performance

3.5 Check Session ID rotated or not after a successful log in:

In this case we will collect the URL, username, password from the brute Force phase, then we will log in by using post method, then we will again log out and get the session ID by using get method, then again we will log in by using post method again, then we will get the session ID after log in. After getting both session IDs we will compare them and check that they are same or different, if they are same then the application is vulnerable, if not then the application is fine. The diagram is given below:

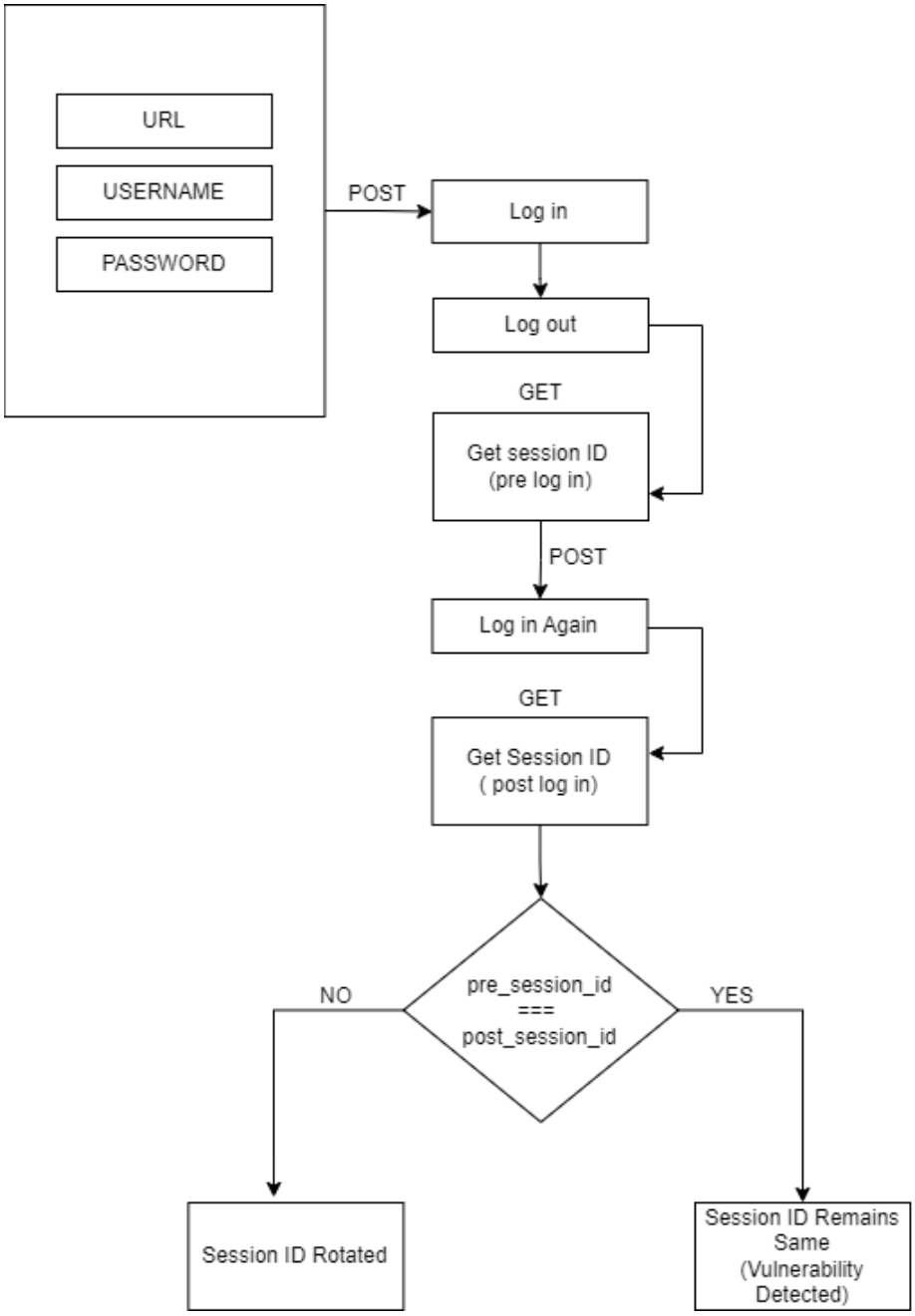

Figure 5: Diagram of Session ID rotation Check

3.6 Check Session ID exposes in URL:

In this phase we will use the same URL. After getting the URL we will check the session name, if there is an available session type, then we will extract its value from it, that is the session ID we were looking for, IF we find exact session ID from the URL then this site is vulnerable of exposing session info, or else it is not vulnerable, the diagram of this flow is given below:

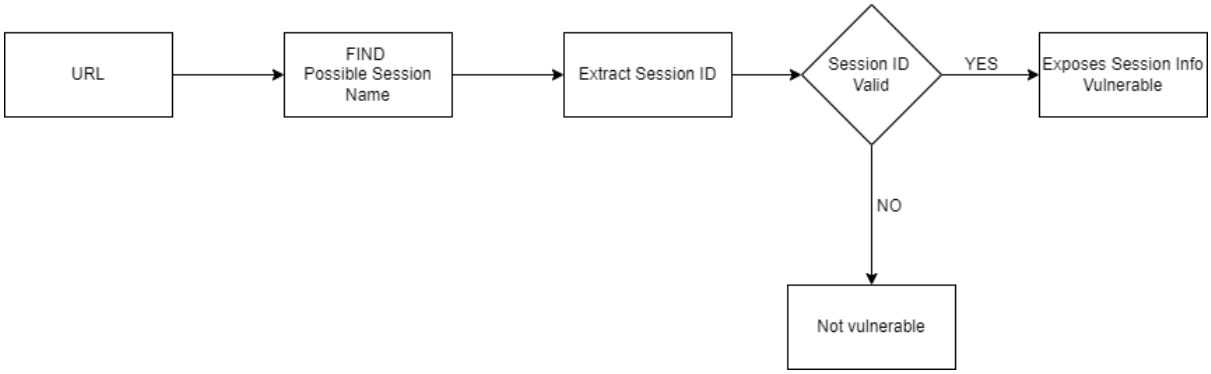

Figure 6: Checking Session ID in URL

3.7 Report:

Here we can see that If the crawling process didn't succeed then it will show, didn't find log in URL, else It will proceed to the brute force phase and see if its vulnerable or not, if that is true then it will show, brute force true, then it will do the same for the other vulnerabilities too. And then it will produce a report that will show that these vulnerabilities are available or not.

CHAPTER 4

IMPLEMENTATION AND RESULTS

4.1 Implementation Overview

In this thesis, we implement an automatic detection method for detecting common broken access control vulnerability automatically. In this chapter, we discuss the implementation of all modules of This detection. In this thesis, our model is divided into five modules such as URL input checking, Log in info gathering by crawling, brute force performing, checking session ID rotated or not after log in, checking session ID exposed on the URL and also shows a report. The implementation level details are described below, with a real life implementation process using URL: https://bwapp.hakhub.net

4.2 Log in info gathering by crawling

This module extracts the web application URL created by requesting it as a client. To Accumulating data using a crawler engine This document uses a structure called scrappy. This structure allows designers to submit requests without issue. There are many libraries that can do this Used to send requests, scrappy is great at siphoning information from URLs Separate them with a readable configuration. Now we

```
PS F:\new d tool\detection-tools-final\detection-tools> python app.py
Enter url: https://bwapp.hakhub.net

----- Start Crawling ------
2022-10-11 17:21:11,903 INFO: >>>> Crawling: https://bwapp.hakhub.net
2022-10-11 17:21:12,864 INFO: >>>> Crawling: https://bwapp.hakhub.net/login.php
2022-10-11 17:21:13,446 INFO: >>>> Crawling: https://bwapp.hakhub.net/login.php

----- Crawling End ------
```

Figure 7: Crawling

4.3 Performing Brute force

Brute force attacks involve guessing credentials and encryption keys through trial and error, or finding hidden web pages. Hackers go through all possible combinations in hopes of guessing correctly. And here we have done the same and performed it like below:

```
----- Start Brute Force ------

----- Trying url: https://bwapp.hakhub.net/login.php
>>>>>>>>>>>     Trying username: root and password: root
>>>>>>>>>>>     Trying username: admin and password: admin

Brute Force Successfull
Username: admin Password: admin
```

Figure 8: Brute Force Performing

4.4 Checking Session ID Rotated or not

A web application is vulnerable to session fixation attacks if an unauthenticated user's session ID does not change after authentication. A malicious user could initiate an unauthenticated session and provide the victim with the associated session ID. Once the victim is authenticated, the malicious user shares that authenticated session. In this URL the session id rotated after log in, the result given below:

```
----- Start Session Rotate check ------

Session id: 01erru7svfaijauh45t4mappc7

Session id: 2fs77h6v6s2rsnr40s8upe3507

No Vulnerability Found.
```

Figure 9: checking Session ID rotation

4.5 Checking Session ID in URL or not

This is intentional, not a bug. When a new session is created, the server generates her cookie and session id in the URL because it doesn't know if the client supports her cookie. When the client comes back for her second time and presents the cookie, the server knows that her session id is no longer needed and removes it for the rest of the session. Even if the client comes back without the cookie, the server should use her

session id rewrite. To detect that we done the below thing, in this URl there were no session id exposed:

```
----- Start Session Id in URL check ------

No Vulnerability Found

----- End Session Id in URL check ------
```

Figure 10: Checking session Id exposed or not in URL

4.6 Report

After detecting these vulnerability, it will generate a report and show it, this will be like below:

```
----- Report ------
Brute Force Vulnerablility Detect: True
Login Sessin Id Rotate Vulnerability Detect: False
Session Id in URL Vulnerability Detect: False
```

Figure 11: Total Report

4.7 Results

In this section we have evaluated the tool more and got a good result, these three broken authentication vulnerability are common so there is various way of detecting them, but our model detecting them in a very short time and automatic way, it's like no extra effort needed to get the report, we just have to provide a valid URL. We have checked on two environments; they are given below:

4.7.1 Environment 1

For the first test, we use BWAAP which is a vulnerable web application. This web application is publicly available. This is a vulnerable web application and this web application vulnerability is publicly known. Then crawl the information, generate illegal requests, analyze responses and report generate as mentioned in 4.6.

URL	Vulnerabilities	Report
https://bwapp.hakhub.net.	Brute Force	TRUE
	Didn't Rotate Session ID after Log in	FALSE
	Exposes session ID in URL.	FALSE

Table 1: Test Result of BWAAP

4.7.2 Environment 2

For the first test, we use Acunetix's vulnerable test site which is a vulnerable web application too. This web application is publicly available. This is a vulnerable web application and this web application vulnerability is publicly known. Then crawl the information, generate illegal requests, analyze responses and report generate as mentioned in 4.6.

URL	Vulnerabilities	Report
http://testphp.vulnweb.com/	Brute Force	TRUE
	Didn't Rotate Session ID after Log in	TRUE
	Exposes session ID in URL.	FALSE

Table 2: Test Result of Acunetix test site

So by seeing from the above tables the tool has performed pretty accurate to any poorly designed web applications. The ratio is given below:

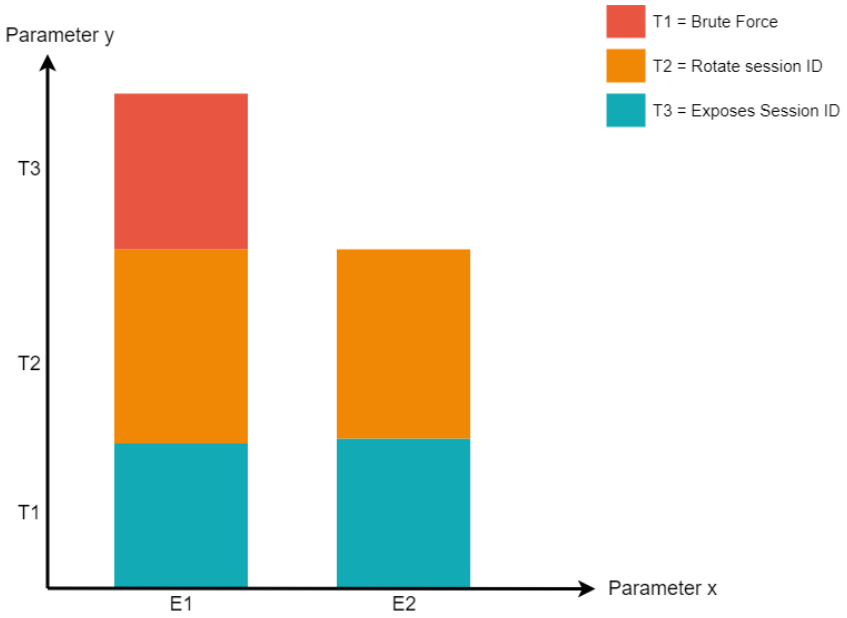

Figure 12: Result Charts

We have seen various brute force attack methodologies, which takes many times to finish, the main characteristics of brute force attacks are Usually a large number of failed login attempts over a period of time lag. This is the most extreme period we have seen About Our Importance 30 minutes' network stream. But in our term It will take 2-3 munities, because we have provided a good amount of datasets to perform the attacks, and after crawling it starts perform instantly. And then after log in it captures session ID and then get logged out and captures session ID again compares it, which is very easy to implement anywhere and get a desired result. So we can see that the implementation works perfectly and gave us an accurate result on any poorly designed site who has these vulnerabilities.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

Hackers can gain access to protected assets using simple passwords and weak network security. This is an old but easy and compelling hacking technique. The recent shift to remote work has seen an increase in brute force attacks aimed at obscuring normal security benefits and undermining traditional security controls. In general, guards cannot detect minutes or seconds and stop in a timely manner without specialized devices. That's why many associations use his PAM and secret phases of board programming to improve security. In this article, we've looked at eight ways to thwart beast attacks on your server or client endpoints.

The main ideal of this study was to produce a detection tool of broken authentication vulnerabilities, there are many vulnerabilities out there, but in this thesis we worked with three common vulnerabilities, brute force, exposes session Id in URL, do not rotate session ID after successful log in. Theses vulnerability exists in poorly designed This type of vulnerability could be get the loss of millions of sensitive information. Automatic black- box testing is hard if the access control policy is not known. This thesis's donation is a completely automatic tool. This tool only needs the main URL and also it generates a report.

The web contains an enormous amount of information. The ability to isolate the data you need is definitely helpful, even essential. Obviously, there are still many datasets available for download at this time, but usually WE can't find the specific information I need for a particular problem. Still, you'll likely find something you really want somewhere on the internet, and you should separate it from this point.

Web crawling is the most common method of removing information from pages. This article describes how to do web crawling in Python. There are several libraries available for this effort. Among them, we use Lovely Soup 4 here. This library is for extracting information from HTML reports, not for downloading. we would like to use a different library for downloading pages:

So, we'll need 2 packages

- requests — for downloading the HTML code from a given URL
- beautiful soup — for extracting data from that HTML string

As our solutions we used beautiful Soup library, Beautiful Soup doesn't take into consideration the simple utilization of intermediaries. In that capacity, it's difficult to utilize Beautiful Soup to separate a lot of information from a similar server without getting your IP prohibited or obstructed. Our implemented tool can only extract URLs and analyze them from HTML. Clearly Our crawler is Not intelligent we look for declared Para- metered URL, using machine learning can boost these tool to another level. So there could be many other works be done using this solution in the future.

References

Alahmad, M., Alkandari, A., & Alawadhi, N. (2022). SURVEY OF BROKEN AUTHENTICATION AND SESSION MANAGEMENT OF WEB APPLICATIONVULNERABILITY ATTACK. Journal of Engineering Science and Technology, 17(2), 874–0882.

Ami, P. V., & Malav, S. C. (2013). Top five dangerous security risks over web application. International Journal of Emerging Trends and Technology in Computer Science, 2(1), 41–43.

Bansal, C., Bhargavan, K., Delignat-Lavaud, A., & Maffeis, S. (2013). Keys to the cloud: Formal analysis and concrete attacks on encrypted web storage. In Lecture Notes in Computer Science (pp. 126–146). Springer Berlin Heidelberg.

Dabirsiaghi, A., Coblentz, N., & Van Der Stock, A. (2009). A Gap Analysis of Application Security in Struts2/WebWork.

Dave, K. T. (2013). Brute-force Attack 'Seeking but Distressing. Int. J. Innov. Eng. Technol. Brute-Force, 2(3), 75–78.

Dawson, M. (2017). Hyper-connectivity: Intricacies of national and international cyber securities.

Dhenakaran, S. S., & Sambanthan, K. T. (2011). Web crawler-an overview. International Journal of Computer Science and Communication, 2(1), 265–267.

Giles, C. L., Sun, Y., & Councill, I. G. (2010). Measuring the web crawler ethics. In Proceedings of the 19th international conference on World wide web (pp. 1101–1102).

Greenwood, D. S. J. S. G., & Khan, Z. L. L. (2014). Smv-hunter: Large scale, automated detection of ssl/tls man-in-the-middle vulnerabilities in android apps. In Network and Distributed System Security Symposium (NDSS). Internet Society (pp. 1–14).

Hans, K., Ahuja, L., & Muttoo, S. K. (2013). Characterization and detection of Redirection Spam. In International Conference on Computer and Communications (pp. 325–331).

Hofstede, R., Jonker, M., Sperotto, A., & Pras, A. (2017). Flow-based web application brute-force attack and compromise detection. Journal of Network and Systems Management, 25(4), 735–758. https://doi.org/10.1007/s10922-017-9421-4

Hossain, M. D., Ochiai, H., Doudou, F., & Kadobayashi, Y. (2020). SSH and FTP brute-force attacks detection in computer networks: LSTM and machine learning approaches. 2020 5th International Conference on Computer and Communication Systems (ICCCS).

Hossain, S., & Mahmud, K. A. (2018). The pros and cons of modern web application security flaws and possible solutions.

Jablon, D. P. (2002). Extended password key exchange protocols immune to dictionary attack. Proceedings of IEEE 6th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises.

Johns, M., Braun, B., Schrank, M., & Posegga, J. (2011). Reliable protection against session fixation attacks. In Proceedings of the 2011 ACM Symposium on Applied Computing (pp. 1531–1537).

Johns, Martin. (2011). Session Hijacking Attacks. In Encyclopedia of Cryptography and Security (pp. 1189–1190). Springer US.

Jose, S., Priyadarshini, K., & Abirami, K. (2016). An analysis of black-box web application vulnerability scanners in SQLi detection. In Proceedings of the International Conference on Soft Computing Systems (pp. 177–185). Springer India.

Koch, R., & Rodosek, G. D. (n.d.). Fast network-based brute-force detection. Retrieved October 9, 2022, from https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.671.7923

Kumar, R., & Goel, A. K. (2014). Automated session fixation vulnerability detection in web applications using the set-cookie HTTP response header in cookies. In Proceedings of the 7th International Conference on Security of Information and Networks (pp. 351–354).

Kumar, Rahul, Indraveni, & Goel, A. K. (2014). Automated Session Fixation Vulnerability Detection in Web Applications using the Set-Cookie HTTP response header in cookies. Proceedings of the 7th International Conference on Security of Information and Networks - SIN '14.

Modi, N. (2020). Comparative Analysis of Session Features in Session Hijacking and Performance Improvement using OTC.

Mujahid, U., Najam-ul-Islam, M., & Shami, M. A. (2015). RCIA: A new ultralightweight RFID authentication protocol using recursive hash. International Journal of Distributed Sensor Networks, 11(1), 642180. https://doi.org/10.1155/2015/642180

Najafabadi, M. M., Khoshgoftaar, T. M., Calvert, C., & Kemp, C. (2015). Detection of SSH brute force attacks using aggregated netflow data. 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA).

Najafabadi, M. M., Khoshgoftaar, T. M., Kemp, C., Seliya, N., & Zuech, R. (2014). Machine learning for detecting brute force attacks at the network level. 2014 IEEE International Conference on Bioinformatics and Bioengineering.

Novotny, J., Tuecke, S., & Welch, V. (2002). An online credential repository for the Grid: MyProxy. Proceedings 10th IEEE International Symposium on High Performance Distributed Computing.

OWASP top ten 2017. (n.d.). Owasp.org. Retrieved October 9, 2022, from https://owasp.org/www-project-top-ten/2017/A2_2017-Broken_Authentication

Pan, J., Mao, X., & Li, W. (2016). Taint Inference for Cross-Site Scripting in Context of URL Rewriting and HTML Sanitization. ETRI Journal, 38(2), 376–386.

Session fixation. (n.d.). Owasp.org. Retrieved October 8, 2022, from https://owasp.org/www-community/attacks/Session_fixation

Session token in URL. (n.d.). Portswigger.net. Retrieved October 8, 2022, from https://portswigger.net/kb/issues/00500700_session-token-in-url

Sounthiraraj, D., Sahs, J., Greenwood, G., Lin, Z., & Khan, L. (2014). SMV-HUNTER: Large scale, automated detection of SSL/TLS man-in-the-middle vulnerabilities in android apps. Proceedings 2014 Network and Distributed System Security Symposium.

Threats and vulnerabilities in web applications 2020–2021. (2022, June 14). Ptsecurity.com; Positive Technologies. https://www.ptsecurity.com/ww-en/analytics/web-vulnerabilities-2020-2021/

Udapure, T. V., Kale, R. D., & Dharmik, R. C. (2014). Study of web crawler and its different types. IOSR Journal of Computer Engineering, 16(1), 1–05.

Udapure, Trupti V., Wireless Communication and Computing) student, CSE Department, G.H. Raisoni Institute of Engineering and Technology for Women, Nagpur, India, Kale, R. D., & Dharmik, R. C. (2014). Study of Web Crawler and its Different Types. IOSR Journal of Computer Engineering, 16(1), 01–05. https://doi.org/10.9790/0661-16160105

Wang, K. C. (2021). Federated Detection of Cross-Site Credential Vulnerabilities and Attacks (Doctoral dissertation).

Wang, Ke Coby, & Reiter, M. K. (2019). Detecting stuffing of a user's credentials at her own accounts. In arXiv [cs.CR]. http://arxiv.org/abs/1912.11118