COURSE GUIDE

CYB 211
WEB HACKING

Course Team Prof. Ibrahim Alhaji Lawal (Course Writer)-NOUN
Prof. Idris Ismail (Course Editor)

NATIONAL OPEN UNIVERSITY OF NIGERIA

National Open University of Nigeria
Headquarters
University Village
Plot 91, Cadastral Zone
Nnamdi Azikiwe Expressway
Jabi, Abuja

Lagos Office
14/16 Ahmadu Bello Way
Victoria Island, Lagos

e-mail: centralinfo@nou.edu.ng
URL: www.nou.edu.ng

Printed 2024

ISBN:978-978-786-277-3

CONTENTS

Introduction

Welcome **to CYB211 Web Hacking,** CYB211 is a two-credit unit course that has a minimum duration of one semester. It is a compulsory course for graduate students that are enrolled in BSc Cybersecurity at the National Open University of Nigeria. The course guides you through the techniques and methodologies for an effective Web Hacking study by means of improving security awareness.

Course Competencies

- The ability to find and exploit web application vulnerabilities and other weaknesses
- Knowledge of how to apply the industry-recommended web security standards and approaches
- The fundamental principles of web application hacking

Course Objectives

- To exploit the applications via HTTP which can be done by manipulating the application via its graphical web interface, tampering the Uniform Resource Identifier (URI) or tampering HTTP elements not contained in the URI.
- To identify the methods that can be used in hacking web applications which include SQL Injection attacks, Cross Site Scripting (XSS), Cross Site Request Forgeries (CSRF), Insecure Communications, etc.

Working through this Course

To successfully complete this course, read the study units, listen to the audios and videos, do all assessments, open the links and read, participate in discussion forums, read the recommended books and other materials provided, prepare your portfolios, and participate in the online facilitation.

Each study unit has introduction, intended learning outcomes, the main content, conclusion, summary and references/further readings. The introduction will tell you the expectations in the study unit. Read and note the intended learning outcomes (ILOs). The intended learning outcomes tell you what you should be able to do at the completion of each study unit. So, you can evaluate your learning at the end of each unit to ensure you have achieved the intended learning outcomes. To meet the intended learning outcomes, knowledge is presented in texts, video and links arranged into modules and units. Click on the links as may be directed but where you are reading the text off line, you will

have to copy and paste the link address into a browser. You can download the audios and videos to view offline. You can also print or download the texts and save in your computer or external drive. The conclusion gives you the theme of the knowledge you are taking away from the unit. Unit summaries are presented in downloadable audios and videos.

There are two main forms of assessments – the formative and the summative. The formative assessments will help you monitor your learning. This is presented as in-text questions, discussion forums and Self-Assessment Exercises.

The summative assessments would be used by the university to evaluate your academic performance. This will be given as Computer Base Test (CBT) which serve as continuous assessment and final examinations. A minimum of three computer base test will be given with only one final examination at the end of the semester. You are required to take all the computer base tests and the final examination.

There are 11 study units in this course divided into four modules. The modules and units are presented as follows:

Study Units

Module 1 Understanding the HTTP Protocol

Unit 1 HTTP Protocol Basics
Unit 2 Information gathering
Unit 3 Transport Layer Security (TLS)
Unit 4 Faulty Password Reset

Module 2 Authentication Bypass Broken Access Control

Unit 1 Role Based Authorization Bypass
Unit 2 Cross Site Scripting (XSS)
Unit 3 Server Site Request Forgery (SSRF)
Unit 4 SQL injection (SQLI)

Module 3 Attacking File Upload Functionality

Unit 1 Executing Remote Code Through Malicious File Upload
Unit 2 Components With Known Vulnerabilities
Unit 3 Insufficient Logging and Monitoring

References and Further Readings

Friedman, Jon (2024). "Attack your Attack Surface" (PDF). skyboxsecurity.com. Archived from the original (PDF) on March 6, 2017. Retrieved March 6, 2024.

Michael and Howard (2022) "Mitigate Security Risks by Minimizing the Code You Expose to Untrusted Users". Microsoft. Archived from the original on 2 April 2022. Retrieved 12 March 2014.

Iman S., Arash H., Saqib H., and Ali A G. (2019) Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy. In 2019 international Carnahan conference on security technology (ICCST), pages 1–8, 2019. tex.organization IEEE

Norah A., Almubairik, M., and Gary W. (2023) Automated penetration testing based on a threat model. In 2016 11th international conference for internet technology and secured transactions (ICITST), pages 413– 414, tex.organization: IEEE.

Smaragdakis, Y., & Balatsouras, G. (2015). Pointer analysis. Foundations and Trends® in Programming Languages, 2(1), 1-69.

Wei L. (2016) Pointer Analysis. Lecture 2. [Lecture PowerPoint slides]. Retrievedfromhttp://web.cs.iastate.edu/~weile/cs513x/2.PointerAnalysis.pdf

Presentation Schedule

The presentation schedule gives you the important dates for the completion of your computer-based tests, participation in forum discussions and participation at facilitation. Remember, you are to submit all your assignments at the appropriate time. You should guard against delays and plagiarisms in your work. Plagiarism is a criminal offence in academics and is highly penalized.

Assessment

There are two main forms of assessments in this course that will be scored. The Continuous Assessments and the final examination. The continuous assessment shall be in three-fold. **There will be two Computer Based Assessment. The computer-based assessments will be given in accordance to university academic calendar. The timing must be strictly adhered to**. The Computer Based Assessments shall be scored a maximum of 10% each, while your participation in discussion

forums and your portfolio presentation shall be scored maximum of 10% if you meet 75% participation. Therefore, the maximum score for continuous assessment shall be 30% which shall form part of the final grade.

The final examination for CYB211 will be maximum of two hours and it takes 70 percent of the total course grade. The examination will consist of 70 multiple choice questions that reflect cognitive reasoning.

Note: You will earn 10% score if you meet a minimum of 75% participation in the course forum discussions and in your portfolios otherwise you will lose the 10% in your total score. You will be required to upload your portfolio using google Doc. What are you expected to do in your portfolio? Your portfolio should be note or jottings you made on each study unit and activities. This will include the time you spent on each unit or activity.

How to get the Most from the Course

To get the most in this course, you need to have a personal laptop and internet facility. This will give you adequate opportunity to learn anywhere you are in the world. Use the Intended Learning Outcomes (ILOs) to guide your self-study in the course. At the end of every unit, examine yourself with the ILOs and see if you have achieved what you need to achieve.

Carefully work through each unit and make your notes. Join the online real time facilitation as scheduled. Where you missed the scheduled online real time facilitation, go through the recorded facilitation session at your own free time. Each real time facilitation session will be video recorded and posted on the platform.

In addition to the real time facilitation, watch the video and audio recorded summary in each unit. The video/audio summaries are directed to salient part in each unit. You can assess the audio and videos by clicking on the links in the text or through the course page.

Work through all self-assessment exercises. Finally, obey the rules in the class.

Facilitation

You will receive online facilitation. The facilitation is learner centred. The mode of facilitation shall be asynchronous and synchronous. For the asynchronous facilitation, your facilitator will:

- Present the theme for the week;
- Direct and summarise forum discussions;
- Coordinate activities in the platform;
- Score and grade activities when need be;
- Upload scores into the university recommended platform;
- Support you to learn. In this regard personal mails may be sent.
- Send you videos and audio lectures; and podcast

For the synchronous:

- There will be eight hours of online real time contact in the course. This will be through video conferencing in the Learning Management System. The eight hours shall be of one-hour contact for eight times.
- At the end of each one-hour video conferencing, the video will be uploaded for view at your pace.
- The facilitator will concentrate on main themes that are must know in the course.
- The facilitator is to present the online real time video facilitation time table at the beginning of the course.
- The facilitator will take you through the course guide in the first lecture at the start date of facilitation

Do not hesitate to contact your facilitator. Contact your facilitator if you:

- do not understand any part of the study units or the assignment.
- have difficulty with the self-assessment exercises
- have a question or problem with an assignment or with your tutor's comments on an assignment.

Also, use the contact provided for technical support.

Read all the comments and notes of your facilitator especially on your assignments, participate in the forums and discussions. This gives you opportunity to socialise with others in the programme. You can raise any problem encountered during your study. To gain the maximum benefit from course facilitation, prepare a list of questions before the discussion session. You will learn a lot from participating actively in the discussions.

Finally, respond to the questionnaire. This will help the university to know your areas of challenges and how to improve on them for the review of the course materials and lectures.

Course Information
Course Code: CYB211
Course Title: Web Hacking
Credit Unit: 2
Course Status: Compulsory
Course Blub: This course covers security awareness and step by step web hacking which include recoinnaissance,scanning, exploitation, maitaninig access and covering tracks.
Semester: Second
Course Duration: 13 Weeks
Required Hours for Study: 65

Course Team
Course Developer: NOUNL
Course Writer: Prof. Ibrahim Alhaji Lawal
Content Editor:
Instructional Designer:
Learning Technologists:
Copy Editor

Ice Breaker

You are welcome to CYB211: Web Hacking, a two-unit course. Please upload your profile such as picture, workplace address, GSM number and other details on your wall. What are your expectations in this course? I am sure you are going to enjoy the course, please fasten your seat belt as you take off. Once again you are welcome.

MAIN COURSE

CONTENTS **PAGE**

MODULE 1 UNDERSTANDING HYPERTEXT TRANSFER PROTOCOL (HTTP)

Module Introduction

Hypertext Transfer Protocol (HTTP) is a set of rules for internet data sharing, such as text, images, video, and other multimedia files. Online browsers and web servers make heavy use of this protocol. HTTP is based on the Client / Server principle. HTTP allows the "computer A" (client) to connect and make a request to the "computer B" (server). The server accepts the connection initiated by the client, and returns a response.

Unit 1 HTTP Protocol Basics
Unit 2 Information gathering
Unit 3 Transport Layer Security (TLS)

In each unit, I will explore a particular topic in detail and highlight self-assessment exercises at the end of the unit. Finally, I highlight resources for further reading at the end of each unit.

UNIT 1 HTTP PROTOCOL BASICS

Contents

1.0 Introduction

You will learn from this unit the definition, component, Request Methods of HTTP and Differences between HTTP and HTTPS. After studying the unit, you will be equipped with skills to define a HTTP and identify software that is malicious based on its distinguishing features. You will also have the required background knowledge for the analysis of HTTP.

2.0 Intended Learning Outcomes (ILOs)

At the end of this unit you will be able to:

- Appraise a Mastering the HTTP Protocols for the purpose of secure and efficient data exchange.

3.0 Main Content

3.1 What is a HTTP?

HTTP (Hypertext Transfer Protocol) is the backbone of data communication on the World Wide Web. It is a protocol that enables the exchange of information between clients (such as web browsers) and servers, allowing users to access and interact with web resources. The HTTP request determines the client's resources and asks the server what "action" the resources are going to take. If a web browser user requests a file by typing an address on the website or by clicking a hyperlink, the browser will generate an HTTP request and send it to the server. The Web server receives the request in the destination computer, performs the processing required and responds with the requested file and any related media data.

Hypertext Transfer Protocol (HTTP) is an application layer protocol that is used for loading web pages consisting of hypertext links. It is designed within the framework of Internet Protocol Suite. This protocol aims to transfer information among network devices. It runs on top of other layers of the network protocol stack. It is used for transferring data in the format of audio, video, hypertext, and plain text. Hypertext Transfer Protocol has a client-server architecture. It allows the reliable transfer of resources between the web application server and the user agent.

When retrieving a Web page, the browser sends a request to a Web server using HTTP. Upon receipt of the request, the server interprets it, sometimes using a CGI script (see CGI-Common Gateway Interface), and returns data. These data, like HTML, text, pictures, sound and programs, can be almost anything.

3.2 Components of HTTP

In an HTTP-based system, there are three main components, including the Client, Proxies, and Server. Let us discuss these components one by one.

- Client: It is a user agent that initiates the request. For displaying a web page, the browser will send an original request to fetch HTML documents that will display the web page. It will parse this file and make an additional request that corresponds to execution scripts and sub-resources that are contained within the pages.

- Proxies: Between the web browser and server, several computers and machines relay HTTP messages. Due to the layered structure of the web stack, many operate at transport, physical, and network levels. Those operating at application layers are known as proxies.

- Web server: This exists on the opposite side of the communication channel that serves the requirement of the client. Virtually, it may appear as a single machine, but in reality, it is a collection of servers that share the load. It may also consist of complex pieces of software that interrogate other computers which partially or totally generate the document on demand.

3.3 Characteristics of HTTP

The following are the features of Hypertext Transfer Protocol:

- This is a stateless protocol since both client and server know each other during the current request only. Due to this, both client and server do not retain information between various requests of web pages.
- It is designed as a connectionless protocol to allow server resources to be equally shared by clients across the world.
- The connection between clients and server exists during the current request and response time only.
- It is media-independent in nature since data can be sent when both the client and server ways to handle data content. Both client

and server must specify the content type within the MIME-type header.

- It is a stateless protocol since every request is executed independently without knowledge of HTTPS requests earlier executed.
- Latency is the amount of time consumed by the user making a request to the server responding to the user. Since no handshaking is required for subsequent requests in HTTP/3, latency is reduced in Hypertext Transfer Protocol.

3.4 Differences between HTTP and HTTPS

used to enable sharing of content over the Internet. The "S" stands for safe, meaning the HTTP link is secured, preventing the exchange of information from being read in plain text or "as you see it." Even if someone were to get the encrypted data shared in the exchange somehow, it would be nonsense to recover the original content with almost no decryption process.

Think of HTTPS as locking a door before a meeting begins; only the parties in the room are able to see what is happening. HTTPS is an update to the HTTP (Hyper Text Transfer Protocol) standard used to enable sharing of content over the Internet. The "S" stands for safe, meaning the HTTP link is secured, preventing the exchange of information from being read in plain text or "as you see it." Even if someone were to get the encrypted data shared in the exchange somehow, it would be nonsense to recover the original content with almost no decryption process. Think of HTTPS as locking a door before a meeting begins; only the parties in the room are able to see what is happening.

3.5 Transactions in HTTP

For accomplishing a single task, an application issues multiple HTTP transactions. For instance, browsers use a transaction to fetch HTML ‘skeleton’ for describing the page layout while others for displaying graphics-rich web pages. Due to the cascade of these transactions, it is known as the collection of resources instead of a single resource.
This transaction consists of HTTP request and response. Here, the client sends a request message to the server to initiate a transaction. The server then replies to this request message by sending a response message. This communication takes place with the help of formalised data blocks called HTTP messages. Let us learn about these factors within the transaction in detail.

3.5.1 HTTP Request

This request carries a series of encoded data that carries different types of information. A request consists of HTTP version type, HTTP method, URL, request headers and HTTP body which is optional.

HTTP method or HTTP verb: This indicates the action that HTTP request expects from the queried server. ‘GET’, ‘PUT’, ‘DELETE’ and ‘POST’ are some of the popular HTTP methods.

HTTP headers: These consist of text information that is stored in key-value pairs and are included in each HTTP request. They are used in HTTP requests for providing information about request context to help the server tailor its response.

Uniform Resource Locator (URL): It is a unique identifier that specifies the location of a resource on the internet. It consists of a protocol and domain name that helps in identifying how and from where to retrieve a resource.

3.5.2 HTTP Response

Server provides an HTTP response to web servers to provide them with requested resources. These responses consist of status code, response headers and a body that is optional. Let us discuss each one of them one by one.

Status code: This is a three-digit code that is used for indicating whether a request has been completed or not. These codes start with 2 to indicate that the request has been properly completed. When a response starts with 4 or 5, it indicates that there is an error and that webpage will not be displayed. 4 indicates client-side error and 5 indicates server-side error. 1 refers to informational response and 3 means redirect.

Response header: Just like the request header, the response also has headers that consist of information such as the language and data format being sent in the response body. Through this information, the client can also learn about the server that has sent the response.

Status Message: It is a non-authoritative and short description of the status code. Servers always return a message for each request.

 **Discussion**

After reading unit 1 from module 1 of this course material, can you categorize those programs that install themselves in your computer without your consent but do not cause you harm as malicious? Start your response by defining a Transactions in HTTP.

 **4.0 Self-Assessment Exercise(s)**

1. Enumerate the advantage of HTTP

Answer

Advantages of Hypertext Transfer Protocol

- The following are the advantages of Hypertext Transfer Protocol:
- It helps in fetching not only hypertext documents but also media like videos and images.
- Enables easy communication among devices and applications on the web.
- It is extensible in nature since clients and servers may agree to add on new field names and information to suit their needs.
- Reduces network congestion as there are very few TCP connections.
- It is a text-based protocol that is simple and readable for users. This allows easier testing for developers, which, in turn, reduces complexity for newcomers.
- Enables receiving applications to quickly open the incoming file.
- It eliminates the need to ask the sender about applications that are required to read or view file content.
- The method in which documents can be cache and for how long is controlled by Hypertext Transfer Protocol. It can help clients instruct cache proxies to ignore the stored document.
- To prevent privacy invasions, web browsers enforce strict separation between websites. Due to this, only pages of the same origin can access each other’s information. HTTP headers help in relaxing this strict separation on the server side.

2. List disadvantages of HTTP?

Answer

HTTP has the following disadvantages due to which its popularity has gone down:

- It is difficult to confirm if the web server sending the request is the same one that has returned the response. This increases the probability that the client may be spoofed.
- There is difficulty of blocking Dos attacks under massive requests.
- Since it cannot prove the integrity of the message of communication, it is impossible to confirm the request and received response is the same. This allows attackers to intercept and tamper the content while the response is in transit.

5.0 Conclusion

You have learnt from this unit what is HTTP', you can also understand its components, applications, and advantages of HTTP. Since this protocol lacks the security of the message, many applications and web browsers appreciate the use of HTTPS over HTTP. While some of the versions have become obsolete, certain versions are still in use.

6.0 Summary

At the end of this unit, you have learnt the definition of HTTP, Components of HTTP, Characteristics of HTTP, Differences between HTTP and HTTPS and Transactions in HTTP.

7.0 References/Further Reading

Franks J. et al. (2020): HTTP Authentication: Basic and Digest Access Authentication. RFC 2617, June 1999, 34 pages.

David H. Crocker (Ed.) (2018): Standard for the Format of ARPA Internet Text Messages. RFC 822, 47 pages.

Wainwright, P. (2021): Professional Apache. Wrox Press, ISBN 1-861003-02-1, 617 pages.

David M. Kristol (2022) HTTP Cookies: Standards, privacy, and politics. ACM Transactions on Internet Technology (TOIT), Volume 1, Issue 2, Pages: 151- 198

UNIT 2 INFORMATION GATHERING

Contents

1.0 Introduction

You will learn from this unit Information Gathering. After studying the unit you will be equipped with the skills to enumeration techniques, Understanding Web Attack Surface Issues with Secure Sockets Layer (SSL)Transport Layer Security (TLS).

2.0 Intended Learning Outcomes (ILOs)

At the end of this unit you will be able to:

- Recognise the Information Gathering

3.0 Main Content

3.1 Enumeration Techniques

Enumeration is a data gathering process wherein a cyber attacker extracts information about a network, such as host IP addresses, DNS and user names, or sharing and network protocols, intending to find weak points and breach the network.

Enumeration is defined as the process of extracting user names, machine names, network resources, shares and services from a system. In this phase, the attacker creates an active connection to the system and performs directed queries to gain more information about the target. The gathered information is used to identify the vulnerabilities or weak

points in system security and tries to exploit in the System gaining phase.

Enumeration is fundamentally checking. An attacker sets up a functioning associated with the objective host. The weaknesses are then tallied and evaluated. It is done mostly to look for assaults and dangers to the objective framework. Enumeration is utilized to gather usernames, hostname, IP addresses, passwords, arrangements, and so on. At the point when a functioning connection with the objective host is set up, hackers oversee the objective framework. They at that point take private data and information. Now and again, aggressors have additionally been discovered changing the setup of the objective frameworks. The manner in which the connection is set up to the host decides the information or data the attacker will have the option to get to.

Techniques for Enumeration

- Extracting user names using email ID's
- Extract information using the default password
- Brute Force Active Directory
- Extract user names using SNMP
- Extract user groups from Windows
- Extract information using DNS Zone transfer

3.2 Understanding Web Attack Surface

The attack surface of a software environment is the sum of the different points (for "attack vectors") where an unauthorized user (the "attacker") can try to enter data to, extract data, control a device or critical software in an environment. Keeping the attack surface as small as possible is a basic security measure.

Attack surface scope also varies from organization to organization. With the rise of digital supply chains, interdependencies, and globalization, an organization's attack surface has a broader scope of concern (viz. vectors for cyberattacks). Lastly, the composition of an organization's attack surface consists of small entities linked together in digital relationships and connections to the rest of the internet and organizational infrastructure, including the scope of third-parties, digital supply chain, and even adversary-threat infrastructure.

Traditionally, a web attack surface has been the listening ports exposed to a threat. A vulnerability like log4shell expands the attack surface for knowledgeable threat actors. In effect, it gives internal applications a new, often externally- facing attack surface. Any system that can receive

attacker- controlled data can be a proxy to attack another system that ultimately logs it. Going forward, we must acknowledge that even systems with no clear direct relationship to an Internet- facing system may be easily exploitable, even if they are behind a firewall or on a network completely isolated from the original point of entry. But perhaps the real lesson here is that widely-used opensource libraries like log4j can serve as a conduit, connecting our most protected, sensitive systems to our most exposed—and our risk assessment processes must be prepared for the inevitable public disclosure of the next critical open-source library vulnerability.

Due to the increase in the countless potential vulnerable points each enterprise has, there has been increasing advantage for hackers and attackers as they only need to find one vulnerable point to succeed in their attack.

There are three steps towards understanding and visualizing an attack surface:

Step 1: Visualize. Visualizing the system of an enterprise is the first step, by mapping out all the devices, paths and networks.

Step 2: Find indicators of exposures. The second step is to correspond each indicator of a vulnerability being potentially exposed to the visualized map in the previous step. IOEs include "missing security controls in systems and software".

Step 3: Find indicators of compromise. This is an indicator that an attack has already succeeded.

One approach to improving information security is to reduce the attack surface of a system or software. The basic strategies of attack surface reduction include the following: reduce the amount of code running, reduce entry points available to untrusted users, and eliminate services requested by relatively few users. By having less code available to unauthorized actors, there tend to be fewer failures. By turning off unnecessary functionality, there are fewer security risks. Although attack surface reduction helps prevent security failures, it does not mitigate the amount of damage an attacker could inflict once a vulnerability is found.

3.3 Issues with Secure Sockets Layer (SSL)

Due to the fact that nearly all businesses have websites (as well as government agencies and individuals) a large enthusiasm exists for setting up facilities on the Web for electronic commerce. Of course,

there are major security issues involved here that need to be addressed. Nobody wants to send their credit card number over the Internet unless they have a guarantee that only the intended recipient will receive it. As businesses begin to see the threats of the Internet to electronic commerce, the demand for secure web pages grows.

A number of approaches to providing Web security are possible. The various approaches are similar in many ways but may differ with respect to their scope of applicability and relative location within the TCP/IP protocol stack. For example, we can have security at the IP level making it transparent to end users and applications. However, another relatively general-purpose solution is to implement security just above TCP. The foremost example of this approach is the Secure Sockets Layer (SSL) and the follow-on Internet standard known as Transport Layer Security (TLS). This chapter looks at SSL which was originated by Netscape. The Internet standard, TLS, can be viewed essentially as SSLv3.1 and is very close to and backward compatible with SSLv3. We will mainly be interested in SSLv3 at present.

As mentioned, the Secure Sockets Layer (SSL) is a method for providing security for web based applications. It is designed to make use of TCP to provide a reliable end- to-end secure service. SSL is not a single protocol but rather two layers of protocols as illustrated in figure 1.1. It can be seen that one layer makes use of TCP directly. This layer is known as the SSL Record Protocol and it provides basic security services to various higher layer protocols. An independent protocol that makes use of the record protocol is the Hypertext Markup Language (HTTP) protocol. Another three higher level protocols that also make use of this layer are part of the SSL stack. They are used in the management of SSL exchanges and are as follows:

1. Handshake Protocol.
2. Change Cipher Spec Protocol.
3. Alert Protocol.

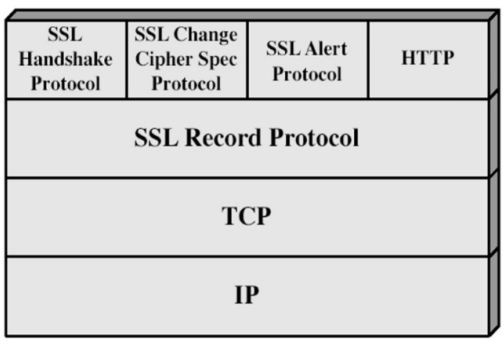


Fig.1.1: SSL protocol stack

The SSL record protocol, which is at a lower layer and offers services to these three higher level protocols, is discussed first.

 **Discussion**

After reading this unit, Understanding Web Attack Surface and Issues with Secure Sockets Layer (SSL).

 **4.0 Self-Assessment Exercise(s)**

1. What is Handshake Protocol?

Answer

This is the most complex part of SSL and allows the server and client to authenticate each other and to negotiate an encryption and MAC algorithm and cryptographic keys to be used to protect data sent in an SSL record. This protocol is used before any application data is sent. It consists of a series of messages exchanged by the client and server, all of which have the format. Each message has three fields:

1. Type (1 byte): Indicates one of 10 messages such as “hello request”
2. Length (3 bytes): The length of the message in bytes.
3. Content (≥ 0 byte): The parameters associated with this message such version of SSL being used.

 **5.0 Conclusion**

You have learnt from this unit Information Gathering. You have equipped with the skills to enumeration techniques, Understanding Web Attack Surface Issues with Secure Sockets Layer (SSL)Transport Layer Security (TLS).

 **6.0 Summary**

At the end of this unit, you have learnt the different types of Information Gathering that are used for Web Hacking which comprises Enumeration techniques, Understanding Web Attack Surface and Issues with Secure Sockets Layer (SSL) In the next unit, you will be introduced to the Transport Layer Security (TLS).

7.0 References/Further Reading

Geoge E. 2023 "Attack Surface Analysis Cheat Sheet". Open Web Application Security Project. Retrieved 30 October 2023.

Manadhata, Pratyusa (2018). An Attack Surface Metric (PDF). Archived (PDF) from the original on 2016-02-22. Retrieved 2024-10-30.

Manadhata, Pratyusa; Wing, Jeannette M. "Measuring a System's Attack Surface" (PDF). Archived (PDF) from the original on 2017-03-06. Retrieved 2019-08-29. {{cite journal}}: Cite journal requires |journal= (help)

UNIT 3 TRANSPORT LAYER SECURITY (TLS)

Contents

1.0 Introduction

You will learn from this unit the techniques for Transport Layer Security (TLS). After studying this unit, you will be equipped with the knowledge and skills for Transport Layer Security (TLS) analysis.

2.0 Intended Learning Outcomes (ILOs)

At the end of this unit you will be able to:

- SSL/TLS Misconfiguration
- Username Enumeration

3.0 Main Content

3.1. SSL/TLS Misconfiguration

Transport Layer Security (TLS) is a cryptographic protocol that is designed to provide both security and data integrity for communications over a reliable transport protocol such as Transport Control Protocol (TCP). TLS allows client-server applications to communicate across a public network while preventing eavesdropping, tampering, and message forgery by providing end point authentication and confidentiality over the Internet. The goals of the TLS protocol, in order of priority, are cryptographic security, interoperability, extensibility, and relative efficiency. TLS is designed to be application protocol

independent. TLS protocol consists of two main components: Handshake protocol, to set session states and shared private keys, and Record protocol, to transmit data securely using the shared keys.

Transport Layer Security, or TLS, is a widely adopted security protocol designed to facilitate privacy and data security for communications over the Internet. A primary use case of TLS is encrypting the communication between web applications and servers, such as web browsers loading a website. TLS can also be used to encrypt other communications such as email, messaging, and voice over IP (VoIP). In this article we will focus on the role of TLS in web application security.

The handshake also handles authentication, which usually consists of the server proving its identity to the client. This is done using public keys. Public keys are encryption keys that use one-way encryption, meaning that anyone with the public key can unscramble the data encrypted with the server's private key to ensure its authenticity, but only the original sender can encrypt data with the private key. The server's public key is part of its TLS certificate.

Once data is encrypted and authenticated, it is then signed with a message authentication code (MAC). The recipient can then verify the MAC to ensure the integrity of the data. This is kind of like the tamper-proof foil found on a bottle of aspirin; the consumer knows no one has tampered with their medicine because the foil is intact when they purchase it as describe in figure 1.2.

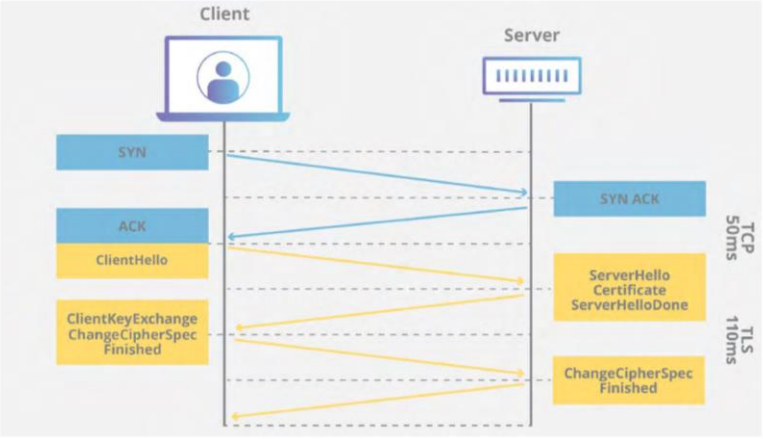


Fig. 1.2 How does TLS affect web application performance

An unauthenticated attacker on the same network as the victim could potentially intercept credentials being sent and hijack the victim’s session. This can lead to a compromise in data confidentiality and integrity.

An attacker able to modify a legitimate user's network traffic could bypass the application's use of SSL/TLS encryption, and use the application as a platform for attacks against its users. This attack is performed by rewriting HTTPS links as HTTP, so that if a targeted user follows a link to the site from an HTTP page, their browser never attempts to use an encrypted connection.

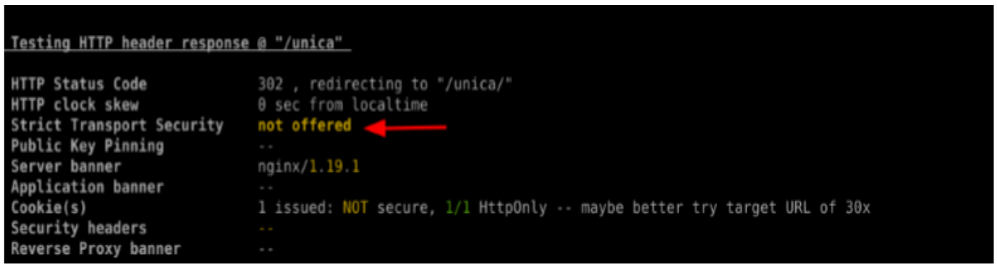


Fig. 1.3 CBC ciphers supported

The server is offering CBC ciphers which are not safe as they are vulnerable to known cryptographic attacks. Successful exploitation is however very complex and highly unlikely.

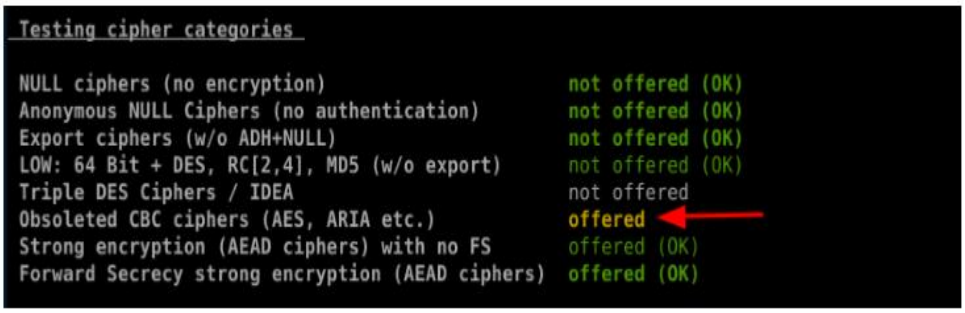


Fig. 1.4 Obsolete TLS 1.0 and TLS 1.1 supported

The server is offering TLS 1.0 and TLS 1.1 which are deprecated and are vulnerable to known cryptographic design flaws. In this case, it appears you are vulnerable to the BEAST and Lucky13 attacks. Successful exploitation is however very complex and highly unlikely.

3.2 Username Enumeration

By comparing the responses, the system gives to different username inputs, one can determine which usernames are valid within a system through a process known as “username enumeration through response analysis.” When a system responds differently to authentication attempts or data requests, it can give away whether or not a submitted username exists. This is known as a security vulnerability. By taking advantage of these variations, attackers can generate a list of legitimate usernames that they can use for additional attacks like phishing, targeted social engineering, or brute-force password cracking.

By verifying which usernames are legitimate targets, username enumeration carries the risk of minimizing the attacker's effort to compromise accounts. In addition, it violates user privacy and may result in further exploitation of system vulnerabilities and unauthorized access.

When an attacker can spot behavioral shifts on the website to determine whether a given username is true, this is known as username enumeration.

Username enumeration usually happens on registration forms when you enter a username that is already taken, or on the login page when you enter a valid username but an incorrect password. Because the attacker can quickly generate a shortlist of valid usernames, this significantly reduces the time and effort required to brute-force a login.

When trying to brute-force a login page, you should be especially aware of any variations in

Status codes: Because the majority of guesses made during a brute-force attack will be incorrect, the returned HTTP status code will almost certainly be the same for the great majority of guesses. A distinct status code in response to a guess suggests that the username was accurate. Although it is recommended that websites always return the same status code, regardless of the result, this recommendation is not always adhered to.

Depending on whether the username and password are wrong together or if the password is wrong alone, the error message that is returned may vary at times. While using identical, generic messages in both scenarios is best practice for websites, occasional typos do happen. Even in situations where the character is not visible on the rendered page, the two messages can be distinguished by just one character being out of place.

Response times: If most of the requests were handled with a similar response time, any that deviate from this suggest that something different was happening behind the scenes. This is another indication that the guessed username might be correct. For example, a website might only check whether the password is correct if the username is valid. This extra step might cause a slight increase in the response time. This may be subtle, but an attacker can make this delay more obvious by entering an excessively long password that the website takes noticeably longer to handle. Even in situations where the character is not displayed on the rendered page, a single misplaced character distinguishes the two messages.

3.3 Username Enumeration

The past two decades have seen an enormous increase in the development and use of networked and distributed systems, providing increased functionality to the user and more efficient use of resources. To obtain the benefits of such systems parties will cooperate by exchanging messages over the network. The parties may be users, hosts or processes; they are generally referred to as principals in authentication literature.

Principals use the messages received, together with certain modelling assumptions about the behaviour of other principals to make decisions on how to act. These decisions depend crucially on what validity can be assumed of messages that they receive. Loosely speaking, when we receive a message, we want to be sure that it has been created recently and in good faith for a particular purpose by the principal who claims to have sent it. We must be able to detect when a message has been created by a malicious principal or when a message was issued some time ago (or for a different purpose) and is currently being replayed on the network.

An authentication protocol is a sequence of message exchanges between principals that either distributes secrets to some of those principals or allows the use of some secret to be recognised. At the end of the protocol the principals involved may deduce certain properties about the system; for example, that only certain principals have access to particular secret information (typically cryptographic keys) or that a particular principal is operational. The principals may then use this knowledge to verify claims about subsequent communication, for example, that a received message encrypted with a newly distributed key must have been created after distribution of that key and so is timely. A considerable number of authentication protocols have been specified and implemented. The area is, however, remarkably subtle and many protocols have been shown to be flawed a long time after they were published. Since authentication relies heavily on encryption and decryption to achieve its goals, we first provide the appropriate knowledge about cryptography.

Due to flaws in many conventional authentication systems, many password attacks have occurred. Determining one’s identity helps to maintain their user accounts on online transactions and services. The authentication is essential to avoid identity theft. Authentication is the process of confirming an individual, whether he is the person that he claims to be. Authentication is one important aspect of security that has to be addressed effectively. Otherwise, the other aspects of security such as authorization, availability, auditing, confidentiality, integrity and non-

repudiation may also be easily compromised. various authentication methods have been discussed in detail.

3.4 Faulty Password Mechanisms

Passwords are one of the most important components in web security. They give authentication and authorization for people to access a web application and personal information. Standridge gave an overview of how password management applications work as well as discussed, the technical implementation and potential security vulnerabilities. Chen proposed a server-side design, seed-based authentication approach for mobile systems. Guan proposed a password-based multi-server authentication protocol to improve the security of verification data by distributing storing data on the cluster.

A web application must be a client-server architecture in the real world. The data entered by user should be properly validated at the client side against the organization's business rules before they are processed. This could prevent attacking on the important servers and improve the servers' performance. There are two places to verify the input - either at front-end or back-end.

Users' logins and passwords should be stored in the database for authentication by the programs running on the web server. There are two different cases to using a password: 1. authentication for login and 2. changing the password.

 **Case Studies**

How to Resolve SSL Configuration Risks

Answer:
Transport Layer Security (TLS) provides security for internet communications. TLS is the successor to the now-deprecated Secure Sockets Layer (SSL), but it is common for TLS and SSL to be used as synonyms for the current cryptographic protocols that encrypt digital communications through public key infrastructure (PKI).

To ensure that websites offer this level of communications security, web administrators generate SSL certificates that use the Hypertext Transfer Protocol Secure (HTTPS) to authenticate communications between the client and the server.

To avoid vulnerabilities that result from security misconfiguration, keep your SSL certificates up-to-date and maintain a secure protection using the SSL protocol on your web server.

4.0 Self-Assessment Exercise(s)

How to Resolve SSL Configuration Issues?

Answer

Each of these configuration issues can be remedied by requesting a new SSL certificate from your certificate authority. Though the certificate authorities may require different information, the process typically follows these four steps:

1. Generate a certificate signing request with your choice of certificate authority.
2. Complete the domain control validation process.
3. Activate the certificate and install it on your servers.
4. Repeat the process for all certificates in your certificate chain.

As you submit a new certificate signing request for your updated SSL certificate, ensure that it has the appropriate parameters to mitigate the configuration finding and prevent misconfiguration attacks.

5.0 Conclusion

You have learnt from this unit that using techniques for Transport Layer Security (TLS), SSL/TLS Misconfiguration, Username Enumeration, Attacking Authentication and Faulty Password Mechanisms.

6.0 Summary

At the end of this unit, you have learnt about the different Transport Layer Security (TLS), SSL/TLS Misconfiguration, Username Enumeration, Attacking Authentication and Faulty Password Mechanisms. In the next unit, you wil be learning about Role Based Authorization Bypass.

7.0 References/Further Reading

Michael S & Andrew H. (2012). Practical Malware Analysis (1st edition). No Starch Press, San Francisco

Oktarianto D & Muhardianto I. (2013). Cuckoo Malware Analysis. Packet Publishing Ltd

John A. (2006). Computer Viruses and Factor Authentication Bypass Broken Access Control

Bruce D, Alexandre G & Elias B. (2014). Practical Reverse Engineering: x86, x64,Arm, Windows Kernel, Reversing Tools and obfuscation. (1st edition). Wiley press.

Michael H, Andrew C, Jamie L & Aaron W. (2014). The Art of Memory Forensic: Detecting Malware and Threats in Windows, Linux and Mac memory. (1st edition). Wiley press.

MODULE 2 AUTHENTICATION BYPASS BROKEN ACCESS CONTROL

Module Introduction

Broken Authentication is a kind of web vulnerability which occurs due to the misconfiguration of session management. After an authentication process completed, a session will be created which will be activated for data communication between the server and a particular user. The problem of Broken Authentication exploiting session is a mismanagement problem. If any intruder can get access in the active session of any specific user bypassing the authentication process, the scenario is treated as broken Exploiting Authentication problem of the given application.

A session request is raised by a user of a web application through the login page where the user credential has been provided. Once the given request has been sent from the client side to server side, the server initiates a query to the database for checking whether the user provided credential is matched ith the record of the database or not. As soon as the validation process is successful, a session with a specific ID will be allocated for the user to communicate the application. A user then can access the system with a given privileges provided by the administrator of the system for getting different services.

A valid session works for a certain duration which is predefined by the system designer. Browsers stores the user credential in the authentication cookie so that the session will remain continue once the session is expired its period by sending the authentication information to the server side. This process is performed automatically behind the user interface which will reduce the effort of the user to authenticate they repeated. However, the intruder can catch and get access into other's active session by using different applications like, cookie manager, eat my cookie, advanced cookie manager+, etc., in case the user missed to close the session as directed bythe application designer.

Unit 1 Role Based Authorization Bypass
Unit 2 Cross Site Scripting (XSS)
Unit 3 Server Site Request Forgery (SSRF)
Unit 4 SQL injection (SQLI)

In each unit, I will explore a particular topic in detail and highlight self-assessment exercises at the end of the unit. Finally, I highlight resources for further reading at the end of each unit.

UNIT 1 ROLE BASED AUTHORIZATION BYPASS

Contents

1.0 Introduction

You will learn from this unit how to detect the Role Based Authorization Bypass. The Authentication and Authorization Processes, Password-Based Authentication and Role-Based Authorization will be discussed. After studying this unit, you will be able to perform basic analysis on how to perform Authorization Bypass.

2.0 Intended Learning Outcomes (ILOs)

At the end of this unit you will be able to:

- Authentication and Authorization Processes
- Password-Based AuthenticationIdentify

3.0 Main Content

3.1 Authentication and Authorization Processes

To allow a user to access a protected resource, the user needs to pass two layers of authentication and one layer of authorization. The first authentication level is password-based authentication conducted by Authentication Microservice. It requires the user to send his/her credentials to the login API to be verified. The verification includes accessing the Authentication Microservice's database to look for a matching username and an identical password hash. And then, in case of

success, generating a token that is embedded with the user's claims which are retrieved from the same database. Using the SQL or NoSQL technology to complete this process may make a difference in terms of performance. Thus, it was decided to include password-based authentication in the experiments to see which database technology performs better in this context.

The second level of authentication is token-based. This is performed by any microservice that exposes protected APIs. For more explanation, when a protected API is called, the responsible microservice checks whether an access token is attached to the request or not. If a token is found, the microservice verifies that token against some criteria such as the issuer of the token as well as the expiration date. Finally, if all tests are passed, the procedure of token-based authorization commences. The experimentation stage was not concerned about the second layer of authentication because database access is not required. However, it was planned to test the authorization process. The token-based authorization in the suggested microservice architecture is the role-based authorization process, which the SQL and NoSQL databases' structures were prepared for in the previous two sections. It is a procedure performed whenever an HTTP request arrives by the Authorization middleware that is found in each microservice.

The first step in this process is extracting the user's roles from the token. Then, slicing the HTTP request to find the request path and the HTTP method, i.e., determining the information of the requested API. The third step is to find an association between the roles of the user and that API in the Authorization Microservice's database. And finally, allowing the user to access the functionality provided by the API or return a refusal in case of failure. Similar to password-based authentication, employing a SQL or NoSQL database to be accessed during the role-based authorization may present different performance levels. To make a proven comparison, an experimentation phase is required. In the next section, all the aspects of the conducted database performance tests will be discussed in detail

3.2 Password-Based Authentication

The password-based authentication process comprises two programmatic main steps. The first is the search to find a match for a username and password, submitted by a user, inside the authentication database. The second is the generation of a token that contains the user's claims which are retrieved from the authentication database Aswell. To complete such a process efficiently, the right database selection decision must be made. Regarding this context, the expectation is that using a NoSQL database gives better performance. As proven in research works,

the NoSQL technology performs well in the execution of simple CRUD operations.

In the password-based authentication process, the token generation step includes straightforward read operation and what happened in the search step is reading as well but without a retrieval action. Despite this assumption, it is speculated that SQL technology will outperform NoSQL at some point with the increase in the number of user records. The reason for that is that both the search and the claims fetching tasks depend on indexed columns. Indexes in SQL work on enhancing the search and querying speed to achieve optimization. The results of authentication affirmed these two hypotheses. In the beginning, they revealed the superiority of the NoSQL database in performance, then, at a certain limit, they showed reversing in favor of the SQL database.

To authenticate users based on passwords efficiently, it was discovered that the NoSQL database technology is better to use with small microservice-based systems which involve about 1500 users or less. On the other hand, SQL was found to be more suitable with medium- to large-scale microservice-based systems that deal with more than 1500 users

3.3 Role-Based Authorization

To authorize a user based on roles that are associated with APIs, simple database read operations are required. That leads to the anticipation of the suitability of the NoSQL technology for this context in terms of performance because, as was mentioned before, NoSQL was proven to carry out simple CRUD operation sufficiently. After performing experiments, surprising results were obtained. The SQL and NoSQL databases spent equal time completing the authorization process. The process was too short and lasted solely one millisecond to finish. The shortness of the process is the reason behind the equivalence since there was no opportunity for NoSQL to exceed. Even when the authorization performance testing was repeated to include two roles each of which is holding the half set of the system's APIs, it was expected that NoSQL will exhibit more superiority.

The reason behind that is each role in NoSQL is represented by a document that stores an array of the linked APIs' IDs. This contrasts with SQL which preserves the relationships between all roles and APIs inside one table. Nevertheless, as discussed, the procedure's duration was not long enough to show any difference in performance. That shows the significance of the conducted performance tests since developers may think in the same way as the authors. And consequently, insist on employing a NoSQL database in authorization losing the feature of

preserving data consistency that SQL provides. Security contexts are very sensitive so if there is a possibility of using reliable data sources like SQL databases, then it should be done.

This branch of experimentation focused on measuring the required time to authenticate one virtual user when the SQL and NoSQL authentication databases are filled with a particular number of users' records. However, when the NoSQL partis scanned, a disparity between numbers is noticed besides an obvious performance drop in the last test.

3.4 Authorization and Authentication Roles

Authorization (A2) is the process of verifying that a known person has the right to perform a certain operation. Prior to authorization, an operator starting a client application needs to be authenticated (A1). The authentication process is described elsewhere [2] and we assume that it has produced an authentication token, containing data needed for the authorization. Since it is not practical to describe authorization restrictions for every single user, we group users into roles. Roles correspond to the actual role played by a group of users in the control system such as LHC Operator, Beam Transfer Expert, Beam Instrumentation Developer, etc. User can be member of several roles, according to different roles he might take, operating, maintaining or developing the control system. Similarly, authorization permission is a function of location of the node issuing an operation. Location is represented by a set of host names and can be used to group e.g. all the computers from the CCC. For certain locations (e.g. from the CCC) all hosts are eligible for authorization by location, meaning that no user role is needed for an operation to succeed. However, critical
operations should be protected by limiting the access only to selected user roles.

Correspondingly, an application issuing an operation can be subject to authorization. Some demands may be permitted only from a given application while for the others this argument is irrelevant. Certain operations may be dependent on the accelerator mode, representing a phase in the work of the accelerator (e.g. SHUTDOWN or COOLING). Again, we expect the majority of permission rules to be independent of this argument.

In-Text Question: What is the deference between Authentication and Authorization

Answer
Authentication is the process of proving that the user with a digital identity who is requesting access is the rightful owner of that identity.

Depending on the use-case, an 'identity' may represent a human or a non-human entity; may be either individual or organizational; and may be verified in the real world to a varying degree, including not at all.

Authorization Determining a user's rights to access functionality with a computer application and the level at which that access should be granted. In most cases, an 'authority' defines and grants access, but in some cases, access is granted because of inherent rights (like patient access to his/her own medical data). Authorization is evaluating what accessor rights an identity should have in an environment.

 **Discussion**

After reading this unit, explain the Role Based Authorization Bypass process affect the Web Hacking process, start your response by clearly stating whether you are proposing or opposing the motion.

 **4.0 Self-Assessment Exercise(s)**

1. What is Multi-Factor Authentication?

Answer: An approach whereby a user's identity is validated to the trust level required according to a security policy for a resource being accessed using more than one factor (something you know (e.g., password), something you have (e.g., smartphone), something you are (e.g., fingerprint).

2. Explain Role-based access control

Answer: A pattern of access control system involving sets of static, manual definitions of permissions assigned to "roles", which can be consistently and repeatably associated with users with common access needs. Role-based access control is a control scheme in which roles are granted to identities, and those roles determine what access to resources those identities should have. Basic roles might beading" and "read-only user" – an admin would be able to make changes to a system and a read-only user would only be able to view resources.

5.0 Conclusion

You have learnt from this unit the Authentication and Authorization Processes, Password-Based Authentication, Role-Based. You have also learnt Authorization and Authorization and Authentication Role.

6.0 Summary

At the end of this unit, you have learnt Authentication and Authorization Processes, Password-Based Authentication, Role-Based. You have also learnt Authorization and Authorization and Authentication Role. In the next unit, you learn about Cross Site Scripting (XSS).

7.0 References/Further Reading

Javaid, U. Siang, A. K., Aman, M. N and Sikdar, B. (2018) “Mitigating loT Device based DDoS Attacks using Blockchain,” in Proceedings of the 1stWorkshop on Cryptocurrencies and Blockchains for Distributed Systems, Munich, Germany, 2018, pp. 71–76. [Online] Available doi:10.1145/3211933.3211946.

Ting, P. Tsai, J. and Wu, T. (2022) "Signcryption Method Suitable for Low-Power IoT Devices in a Wireless Sensor Network," in IEEE Systems Journal, vol. 12, no. 3, pp. 2385-2394. [Online] Available doi: 10.1109/JSYST.2017.2730580URL: https://ieeexplore.ieee.org/document/8010797/

Moinet, A. Darties, B. and Baril, J. L “Blockchain based trust &authentication for decentralized sensor networks,” pp. 1–6.

Huh, S. Cho, S. and Kim, S. (2022) "Managing IoT devices using blockchain platform," 2017 19th International Conference on Advanced Communication Technology (ICACT), Bongpyeong, pp. 464-467. [Online] Available doi: 10.23919/ICACT.2017.7890132

UNIT 2 CROSS SITE SCRIPTING (XSS)

Contents

1.0 Introduction

The use of web applications has now become an essential part in our lives. It is because of the wide range of online services that it provides, such as banking, shopping, entertainment and social media etc. However, web applications are vulnerable to a range of security threats, including cross-site scripting attacks. The cross-site scripting is an exploitable vulnerability allowing an attacker to execute malicious codes in a website. It can affect the website or steal confidential data from the website or taking over a user's account. These attacks can lead to various consequences, including data theft, website defacement, and malware distribution. As web applications continue to grow in popularity, the importance of understanding XSS attack techniques and prevention strategies becomes increasingly critical.

Regardless of size or popularity, any web application can be vulnerable to a type of attack known as cross-site scripting (XSS). The attack can be launched through various vectors, such as input fields, cookies, and URLs, and can affect all users who visit the affected web page. As such, XSS has become one of the most significant security concerns for web developers and users alike. While XSS attacks can have serious consequences, they are often difficult to detect and prevent. The dynamic and interactive nature of web applications means that data can flow between different components of the application, making it challenging to identify and sanitize user input effectively.

Moreover, the complexity of modern web applications means that XSS vulnerabilities can arise from multiple sources, such as third-party libraries or plugins. To address these challenges, developers and security experts have developed various techniques and best practices to prevent and mitigate XSS attacks. These include input validation, output encoding, and content security policies, among others. However, the effectiveness of these techniques depends on the specific context and nature of the web application, and there is no one-size-fits-all solution. As such, ongoing research and development in the field of web security are crucial to staying ahead of emerging threats and vulnerabilities. By understanding the nature of XSS attacks and implementing effective prevention and mitigation strategies, web developers can help ensure that web applications remain safe and secure for users.

2.0 Intended Learning Outcomes (ILOs)

At the end of this unit you will be able to:

- Examine a Cross Site Scripting (XSS)

3.0 Main Content

3.1 Cross-site Scripting XSS

Cross-Site Scripting (XSS), refers to a web application vulnerability that enables an attacker to execute malicious code on a website, and it can occur on any website, regardless of its popularity or size. Within a web application's trusted environment, a harmful script, usually containing HTML/JavaScript code, is executed by an attacker who has taken control of a user's browser. Whenever a user visits the website, in which the attacker injected the payload, the payload gets executes and the attacker will be able to steal the user’s data or manipulate the user. For example, an attacker can inject a payload in a trusted/popular website such that whenever the page load, it will display of message like “click here to get money” or a payload such that it has potential to seal user cookies, then whenever a user visits the website, the user will get the popup and can be manipulated or the data of the user can be stolen.

3.2 Payload

Cross-site scripting is a vulnerability that enables an attacker to run malicious code within a website. Those malicious code are known as ‘payload’. A payload is a part or snippet of a code. It is not the whole

code instead it is a small snippet of a code that an attacker uses to carry out malicious activities.

For example, <script>alert (1); </script> or These are the simple payloads that will simply popup an alert box on user’s screen. There are countless payloads and an attacker can even create a custom payload in order to bypass the securities in the website.

(i) Tag Payloads In this, the payload is in form of tag, example <script>, <img> etc. For example, In A vulnerable website www.example.com- If a user gives input as ‘test’
The source code would look like:

```
<div class = “searchMessage”>
     <p>Search result for test</p>

<div>
```

The output of the page will be: ‘Search result for test’- If an attacker gives payload as an input, <script>alert(1);</script>

```
<div class = “searchMessage”>
     <p>Search result for
<script>alert(1);</script></p>
```

The input would be an alert box saying ‘1’.

(ii) Attribute Payload This type of payloads can be used where the attribute value changes according to user input. Some attribute payloads are: onmousemove = “alert (1)”, onclick = “alert (1)” etc. for example, in a vulnerable website if a user enter input as ‘test’ The source code would look like:

```
<p name = “test”>test</p>
```

If the user enter payload as, “onmousemove = “alert (1)” The source code would look like: When the user moves the mouse, the payload would be executed.

```
<p name = “” onmousemove = “alert(1)”></p>
```

3.3 Types of cross Site Scripting (XSS)

Three primary categories of Cross-Site Scripting include Reflected XSS, Stored XSS, and DOM XSS. These types of XSS exploit the same channel for both the attack and output, allowing attackers to receive the attack's output on the same page.

3.3.1 Reflected XSS

The simplest form of a cross-site scripting attack is referred to as Reflected XSS. This type of attack is often observed when a webpage accepts user input and subsequently displays it on the page. For example, many websites that contains search bars, such as Amazon, it contains a search bar where we can search for any product and when we search for any product, the page displays the message “search result for <Input>”, but any payload won’t be executed as it has taken measures to avoid it. So, basically any website that that reflects the given input without any security can lead to reflected cross-site Scripting.

Without proper secure practices, if an attacker passes payload in the input field, when the web page displays the input, the web page would render the payload too which will lead for execution of the code. An attacker can simply send the URL of the website which contains payload in the parameters. When the victim opens the URL, the payload gets executes in the victim’s browser and can lead to manipulation and data stealing. On injecting a payload in the website the URL would look like, “http://example.com/?page=<script>alert(1);</script>” and the attacker would send this link to the victim. Instead of alert, an attacker can put some malicious code that can lead to manipulation or data stealing and the attacker can share the URL to the victim. Once the victim opens the URL, the payload would be executed.

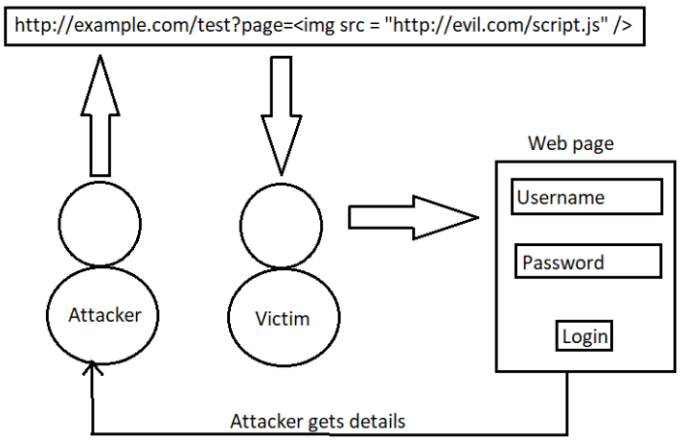


Figure 3.1 Reflected Cross-site scripting working

3.3.2 Stored XSS

Stored XSS is a type of XSS attack that occurs when a website takes input from a user, stores it in a database, and subsequently retrieves the data to display it on the page. This type of attack is more dangerous than Reflected XSS as the malicious script is permanently stored on the server and can affect multiple users who access the same page. For

example, in a shopping website, a seller can post product details to sell them to user, and the website will show all the products to the visited users. In this case, an attacker would pass a payload as an input in the website and whenever a user tries to visit and load the page, the payload would be fetched from the database and gets executed. An attacker will need not send any link to the victim, even if the victim himself tries to lead the website, the payload would be fetched from database and gets executed.

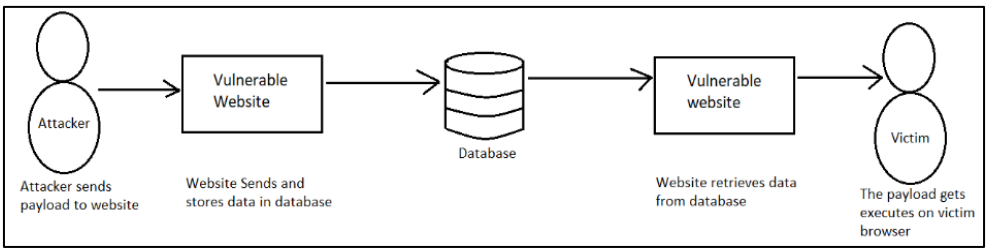


Figure 3.2 Stored Cross site scripting Working

3.3.3 DOM (Document Object Model) XSS

DOM XSS, also known as Cross-Site Scripting, is a type of web application vulnerability that permits attackers to insert harmful code into a web page. Once executed, the malicious code operates in the victim's browser. DOM-based cross-site scripting attacks differ from traditional XSS attacks in that they take place on the client-side. In other words, these attacks occur when a web application neglects to adequately sanitize user input. An example of how a DOM XSS attack might work: Suppose there is a web application that allows users to enter their name, which is then displayed on the page using JavaScript. The application uses the following code to display the user's name:

```
document.getElementById("name").innerHTML = user_input;
```

An attacker can exploit this vulnerability by entering the following as their name:

```
<script>
  // Malicious code goes here
</script>
```

The execution of the attacker's code happens simultaneously with the display of the user's name on the page, causing the code to execute in the victim's browser. This can result in the attacker gaining unauthorized access to sensitive information or carrying out harmful activities.

3.4. Blind XSS

Blind XSS is a type of a XSS where the attack is done on one channel and the output is displayed on another channel. For example, in many websites there is a feed back form where the users can give feedback or help messages and the feedback will be sent to the admin panel of the website. In such websites if the website is not secure, an attack can pass payload in the feedback form and submit it. In the admin panel side, once the admin loads the feedback, the payload will be executed. Once a payload is executed on the admin side of the website, the attacker will be able to do more dangerous things other than normal impacts. An attacker will also be able to steal admin credentials or hijack the session. In this type of XSS, an attacker may not be confident of the attack will be successful or not. The attacker Just gives it a try and if the system is not secure, the payload executes on the other channel as describe in figure 3.3. below.

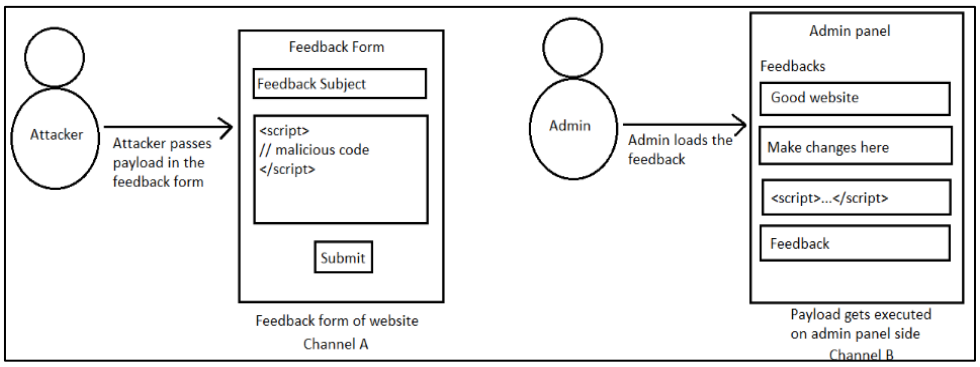


Figure 3.3Blind XSS working

Discussion

Can you Discuss Document Object Model? You can choose to respond individually or as a group of three (3) members at maximum.

Case Studies

Assume that you are working with your computer on a normal day and an alarm pops up on your screen notifying you of malicious software on your computer. As malware analyst, you checked and noticed that the malware binary is obfuscated. What will be your first point of call for analyzing the malware?

4.0 Self-Assessment Exercise(s)

Explain the Impact of XSS

Answer: XSS can have impact on both individual and organizations. Some of the potential impact of XSS are:

(i) XSS can be used by attackers to steal sensitive information, including but not limited to login credentials, credit card numbers, and other personal data that the user enters on the vulnerable website.
(ii) (Website defacement: Attackers can inject malicious code into a website to deface it, making it unusable or displaying inappropriate or offensive content.
(iii) Phishing attacks: Attackers can use XSS to redirect users to fake login pages or other fraudulent websites designed to steal user credentials and other sensitive information.
(iv) XSS can also be used by attackers to distribute malware, including viruses, Trojans, or ransomware, to unsuspecting users who visit the affected website.
(v) Reputation damage: Websites that are vulnerable to XSS attacks can suffer damage to their reputation and lose the trust of their users.
(vi) Legal and regulatory consequences: Companies that suffer data breaches resulting from XSS attacks can face legal and regulatory consequences, including fines and lawsuits

5.0 Conclusion

You have learnt from this unit XSS attacks pose a major security threat to web applications and can result in severe consequences such as website defacement and data theft. While various techniques and best practices have been developed to prevent and mitigate XSS attacks, there is no one-size-fits-all solution. By implementing effective security measures, web developers can help ensure that web applications remain safe and secure for users.

6.0 Summary

As you have learnt the Cross-site Scripting XSS, Payload, Types of cross Site Scripting (XSS) that include the Reflected XSS, Stored XSS

XSS at the last part of the unit the Blind XSS has been explained. The conducting regular security audits as best practices for preventing XSS attacks. By following these measures, web developers can enhance the security of their web applications and protect their users from potential harm caused by XSS attacks. In the next unit the Server Site Request Forgery (SSRF) will be discuss.

7.0 References/Further Readings

Khonji, M. Iraqi, Y. and Jones, A. (2023) "Phishing detection: a literature survey," IEEE Commun. Surv. Tutorials, vol. 15, no. 4, pp. 2091–2121.

Alimadadi, S. Mesbah, A. and Pattabiraman, K. "Understanding asynchronous interactions in full-stack JavaScript," in 2016 IEEE/ACM38th International Conference on Software Engineering (ICSE), 2016, pp. 1169–1180.

Sood, A. K. and Zeadally, S. (2020) "Drive-by download attacks: A comparative study," vol. 18, no. 5, pp. 18–25.

Shanmugasundaram, G., Ravivarman, S. and P. Thangavellu, S. (2021)"A study on removal techniques of Cross-Site Scripting from web applications,"4th IEEE Spons. Int. Conf. Comput. Power, Energy, Inf. Commun. ICCPEIC 2015, pp. 436–442,

UNIT 3 SERVER SITE REQUEST FORGERY (SSRF)

Contents

1.0 Introduction

This unit will introduce you on how to identify a Server Site Request Forgery based on Server Site Request Forgery Vulnerability and How to identify SSRF with Crashtest security as well as Server-Side Request Forgery Prevention Techniques. You will learn the best practices in preventing security logging and monitoring failures and best practices in preventing SSRF vulnerabilities.

2.0 Intended Learning Outcomes (ILOs)

At the end of this unit you will be able to:

- Analyse Server Site Request Forgery

3.0 Main Content

3.1 What is Server Site Request Forgery?

Server-side request forgery (SSRF) is a security vulnerability that arises from a vulnerability in web applications. For example, when the services are accessed via URL the attacker supply or modify a URL to access services on servers that he is not permitted to use. In this research, various types of SSRF attacks are discussed, and how to secure web applications are explained. Various techniques have been used to detect and mitigate these attacks, most of which are concerned with the use of machine learning techniques. The main focus of this research was the application of deep learning techniques (LSTM networks) to create an intelligent model capable of detecting these attacks.

Server-side request forgery attacks are orchestrated mainly by inducing the server-side application to make malicious requests. This allows a malicious actor to obtain unauthorized access to restricted internal services and connect with arbitrary external entities, potentially exfiltrating sensitive data. By modifying the URL parameter, attackers can also read the server‘s configuration settings and connect to internal systems that are not intended for exposure. With this attack, hackers can compromise the application itself or other backend systems with which the target server communicates.

Most server-side request forgery attack vectors are easy to find since each application‘s traffic flow includes URL parameters within the request body. Some standard techniques that attackers use to uncover these vulnerabilities include:

- Partial request URLs - In some vulnerable web servers, the application only includes a partial path in the request‘s URL parameters. This value is parsed on the server side and incorporated into a full URL. Attackers can recognize this value as a URL path and modify it, enabling the server to make malicious requests.

- Inclusion of URLs in data formats - Some applications rely on data formats that enable the data parser to allow the inclusion of URLs, making them susceptible to attacks. For instance, if an application receives data in XML format and parses it, attackers might include External XML Entities (XXE) in an incoming request, creating an SSRF attack vector.

- Request forgery via the referrer header - In applications that employ analytic solutions to track users, the application server logs the referrer header to track incoming links. In such instances, the analytics solution includes links in the header to visit and analyze the contents of other third-party sites. These referrer headers offer attack surfaces that allow a malicious actor to obtain and alter a legitimate user‘s incoming request.

SSRF attacks are typically categorized into:

- Blind SSRF attacks - These attacks exploit vulnerabilities that allow attackers to issue a server-side request to a URL. Still, the response to the request is not reflected in the application‘s client-side response. While these vulnerabilities are harder to exploit, a successful attack often leads to severe consequences, including remote code execution on backend systems.

- Direct SSRF attacks - In this type of attack, the hacker tricks the web application into issuing a server-side request and obtains the contents of this server- side response through the application‘s client-side response. Attackers can use this response to compromise the vulnerable server itself or other backend systems connected to it.

SSRF attacks can target almost all public-facing servers that access resources from external systems without validating user-supplied URLs. The vulnerable web server submitting the request is automatically assumed to be trusted. This allows the external attacker to request targets outside the internal network even when protected by application layer controls, firewall policies, network access control rules, or a VPN.

3.2 SSRF Vulnerability - Severity Level

The SSRF vulnerability is ranked number 10 on the OWASP 2021 Top 10 list of vulnerabilities. The vulnerability has a relatively low attack incidence rate of 2.72% since exploiting it requires an application that does not validate user-controlled data while using server-side requests to access resources. On account of the multiple ways to circumvent application layer controls against SSRF, the vulnerability, on the other hand, has a high average weighted exploit of 8.28.

The server-side request forgery vulnerability has a relatively high average weighted impact of 6.72. Some effects of a successful SSRF attack include the following:

- Sensitive data exposure - This approach allows attackers to exploit a target URL and exfiltrate data from services that should not be directly exposed to the internet. Some of such services include metadata storage services, database HTTP interfaces, internal REST interfaces, and files.

- Remote code execution - An attacker can leverage input validation errors to inject malicious code into the server that only expects to read data from trusted sources.

- Cross-site port attacks - Some responses to the server-side request allow an attacker to obtain system-level information of the target server. For instance, data on the server‘s response time may reveal whether the request was processed successfully. Such instances are common targets of cross-site port attacks where hackers can also use port scans to identify good host-port pairs for orchestrating deeper, system-level attacks.

- Denial of Service attacks - These are orchestrated by manipulating backend systems and flooding the target server with large requests, resulting in a server crash. Most internal servers do not support large amounts of traffic and are susceptible to denial-of-service attacks.

3.3 How to identify SSRF with Crashtest Security?

Crashtest Security helps reduce security risks through automated penetration testing and vulnerability scanning. The platform offers a suite of vulnerability scanners that helps detect vulnerabilities and misconfigurations. Vulnerability scanners provided by the Crashtest Security Suite include:

- Microservices scanner - this scanner evaluates inbound and outbound traffic between microservices to ensure no modifications can be done to service requests. The scanner also identifies other vulnerabilities in the microservices that can allow a malicious actor to request other internal services, providing further defense against SSRF attacks.

- HTTP header scanner - Applications use special request headers such as host headers and referrer headers to enable specific functionality for allowing access to resources. Malicious attackers craft header injection attacks to include unintended information within these headers. Crashtest Security‘s HTTP header scanner helps identify and remediate all host header injection vulnerabilities, including those that can lead to SSRF

attacks, such as cross-site port attacks and open redirection exploits.

- XXE vulnerability scanner - The XXE vulnerability scanner helps developers detect issues before attackers leverage External XML Entity (XXE) vulnerabilities to perform SSRF attacks in production.

- URL fuzzer scanner - The scanner helps security analysts find resource files, routes, and directories that are sensitive, hidden, or susceptible to SSRF attacks. This website directory scanner prevents sensitive data exposure and the exfiltration of information that can be used to compromise an entire system.

- OWASP scanner - This scanner performs benchmark tests against all vulnerabilities, including SSRF (A10: 2021), identified by the Online Web Application Security Project.

Crashtest Security also enables security teams to perform ethical hacks and black-box penetration tests to simulate scenarios and assess how attackers leverage SSRF vulnerabilities for attacks. The platform also outputs actionable reports that outline security levels and remediation advice to help security administrators adopt best practices against SSRF attacks.

3.4 Best practices in preventing SSRF Vulnerabilities

Some recommended practices to prevent SSRF vulnerabilities include Disable unused URL Schemas, enforce input sanitization and validation and perform authentication on all internal.

3.4.1 Disable Unused URL Schemas

The application server should strictly accept the input schema that is currently being used for making requests while discarding the rest. This helps prevent SSRF vulnerabilities since it makes it difficult for attackers to craft malicious requests and submit them with their own URLs. Attackers commonly exploit the file://, ftp://, gopher://, and dict:// URL schemas for SSRF attacks as they enable directory and administrative ports access, allowing them to craft malicious server-side requests. It is recommended instead to use the https:// schema that enforces transport layer security and prevents attackers from accessing internal resources even with access to the network.

3.4.2 Enforce input Sanitization and Validation

A common approach to exploiting SSRF vulnerabilities is manipulating the application through user-controllable input and making a malicious request. The application should never trust any incoming input by default to avoid this. Additionally, all incoming inputs should be sanitized to remove unexpected characters to follow a standardized format and ensure no malicious code or commands are injected into the system.

3.4.3 Perform Authentication on all Internal

Some services, such as MongoDB, ElasticSearch, and MemCached, do not require additional authentication to process requests. In such instances, an attacker can exploit a vulnerable server to craft malicious requests and obtain unauthorized access to such services. To keep configuration information and sensitive data secure, it is important to secure these services through an additional layer of user authentication and authorization.

3.5 Best practices in preventing security logging and monitoring failures

Security measures to prevent server-side request forgery (SSRF) attacks include Strict Access Controls, Firewall Policies, Whitelists and DNS Resolution and Response Handling.

3.5.1 Strict Access Controls

Robust network access control rules prevent attackers from exploiting an organization‘s internal networks and submitting malicious requests. Enforcing access controls with multi-factor authentication, role-based authorizations, or other rule-based security measures at the network perimeter restricts attackers from identifying and exploiting SSRF vulnerabilities.

When implementing access controls, organizations should also consider the following and administer rules accordingly for robust security:

- who should have access to what data and systems?
- what level of access does each user need?
- how will users be authenticated?
- how will authorization be granted?
- how will access be monitored and audited?

3.5.2 Strict Access Controls

As the first line of defense, organizations can prevent SSRF attacks by implementing firewall policies defining the external servers that an application can connect to. The policies can either be applied at specific points at the network level or closer to the host using access control rules at the machine‘s loopback network interface.

3.5.3 Whitelists and DNS Resolution

A common approach to combating SSRF attacks is to whitelist all the DNS names and decimal IP addresses the server should connect to. The whitelist should also apply to user-controllable inputs, ensuring the application only accepts known requests. A whitelist approach enforces stricter control over server-side requests, as the application can only accept, bind and transmit content following a pre-configured standard.

4.0 Self-Assessment Exercise(s)

1. Explain the Impact of SSRF Attacks

Answer

If an attacker is able to control the destination of the server side requests they can potentially perform the following actions:-

- Bypass IP whitelisting.
- Bypass host-based authentication services.
- Read files from the web server.
- Retrieve sensitive information such as the IP address
- of the web server behind a reverse proxy.
- Port Scans or Cross Site Port Attack.
- Protocol Smuggling.
- Server Side Rendering.
- Sensitive data exposure.
- Scan the internal network to which the server is connected to.
- Sensitive data exposure.
- Scan the internal network to which the server is
- connected to.

5.0 Conclusion

You have learnt from this unit This unit will introduce you on how to identify a Server Site Request Forgery based on Server Site Request Forgery Vulnerability and How to identify SSRF with Crashtest security as well as Server-Side Request Forgery Prevention Techniques. You will learn the best practices in preventing security logging and monitoring failures and best practices in preventing SSRF vulnerabilities.

6.0 Summary

This unit introduced you on how to identify a Server Site Request Forgery based on Server Site Request Forgery Vulnerability and How to identify SSRF with Crashtest security as well as Server-Side Request Forgery Prevention Techniques. You will learn the best practices in preventing security logging and monitoring failures and best practices in preventing SSRF vulnerabilities.

7.0 References/Further Reading

Ahmad, Z., Khan, A., Cheah, W.S., Abdullah, J., & Ahmad, F. (2021).

Alma, T., & Das, M. (2020). Web Application Attack Detection using deep Learning. ArXiv, abs/2011.03181.

C. Hasegawa and H. Iyatomi, "One-dimensional convolutional neural networks for Android malware detection," 2018 IEEE 14th International

Colloquium on Signal Processing & Its Applications (CSPA), Penang, Malaysia, 2018, pp. 99-102. Sean McElroy, Detecting Server-Side Request Forgery Attacks on Amazon Web Services ISSA Journal February 2020, volume 18, issue 2.

Ghods, A., & Cook, D. (2019). A Survey of Techniques All Classifiers Can Learn from Deep Networks: Models, Optimizations, and Regularization. ArXiv, abs/1909.04791.

Hoffman, A. (2020). Web Application Security: Exploitation and Countermeasures for Modern Web Applications. Web Application Security, accessed, March 2021, https://www.synopsys.com/glossary/what-is-web application-security.html.

Jabiyev, B., & Kharraz, A. (2020). Preventing Server-Side Request Forgery Attacks.

L. (2019). Detecting web attacks with end-to-end deep learning. Journal of Internet Services and Applications, 10, 1-22.

Network intrusion detection system: A systematic study of machine learning and deep learning approaches. Trans. Emerg. Telecommunication Technol., 32.Neural Networks, 61, 85–117.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview.

Server-Side Request Forgery (SSRF) & the Cloud Resurgence, (2020), accessed 2021, https://appcheck-ng.com/server-side-request-forgeryssrf/.

UNIT 4 SQL INJECTION (SQLI)

Contents

1.0 Introduction

This unit will introduce you on how to identify an SQL injection the types of SQL Injection, SQL Queries and SQL Injection Attacks. You will learn SQLI prevention and mitigation.

2.0 Intended Learning Outcomes (ILOs)

At the end of this unit, you will be able to:

- identify an SQL injection the types of SQL Injection, SQL Queries, SQL Injection Attacks and SQLI prevention and mitigation.

3.0 Main Content

3.1 Definition of SQL injection

SQL injection, also known as SQLI, is a common attack vector that uses malicious SQL code for backend database manipulation to access information that was not intended to be displayed. This information may

include any number of items, including sensitive company data, user lists or private customer details.

The impact SQL injection can have on a business is far-reaching. A successful attack may result in the unauthorized viewing of user lists, the deletion of entire tables and, in certain cases, the attacker gaining administrative rights to a database, all of which are highly detrimental to a business.

When calculating the potential cost of an SQLi, it's important to consider the loss of customer trust should personal information such as phone numbers, addresses, and credit card details be stolen.

While this vector can be used to attack any SQL database, websites are the most frequent targets.

SQL is the acronym for Structured Query Language. It is used to retrieve and manipulate data in the database. SQL query is the way to insert, modify, extract, and delete the data in the database. SQL injection the attack in which attacker interfere the queries that are done to the Attacker can perform this attack with many different intentions like:

1. To pull out data: Attackers can take out the sensitive information stored on the database. One example is if the attacker gains access to the admin database.
2. To extract data – Sensitive data will be grabbed by the attacker. Suppose if the admin database is hacked the entire database becomes vulnerable
3. 3.To access data – They try to break the privileges and get access to the entire database and try to manipulate the data.
4. 4.Finger print the database- In this attack, the database version and its type will be derived out by the attacker. This attack helps them to try different types of queries in different applications.
5. Injectable parameters are found – using some of the automatic tools the vulnerable parameters will be found for attack.
6. Authentication Bypass –application authentication mechanisms will be bypassed to enter inside the database.
7. Database schema identification - From the database table name, data type of each field, column name, etc. will be retrieved to gather information successfully.
8. 8.To perform denial of service – Dropping table and system shutdown falls under this category. Attacker tries to intrude inside the system to perform some specific instruction within the database.

3.2 Types of SQL Injections

SQL injections typically fall under three categories: In-band SQLi (Classic), Inferential SQLi (Blind) and Out-of-band SQLi. You can classify SQL injections types based on the methods they use to access backend data and their damage potential.

3.2.1 In-band SQLi

The attacker uses the same channel of communication to launch their attacks and to gather their results. In-band SQLi's simplicity and efficiency make it one of the most common types of SQLi attack. There are two sub-variations of this method:

Error-based SQLi—the attacker performs actions that cause the database to produce error messages. The attacker can potentially use the data provided by these error messages to gather information about the structure of the database.

Union-based SQLi—this technique takes advantage of the UNION SQL operator, which fuses multiple select statements generated by the database to get a single HTTP response. This response may contain data that can be leveraged by the attacker.

3.2.2 Inferential (Blind) SQLi

The attacker sends data payloads to the server and observes the response and behavior of the server to learn more about its structure. This method is called blind SQLi because the data is not transferred from the website database to the attacker, thus the attacker cannot see information about the attack in-band.

Blind SQL injections rely on the response and behavioral patterns of the server so they are typically slower to execute but may be just as harmful. Blind SQL injections can be classified as follows:

Boolean—that attacker sends a SQL query to the database prompting the application to return a result. The result will vary depending on whether the query is true or false. Based on the result, the information within the HTTP response will modify or stay unchanged. The attacker can then work out if the message generated a true or false result.

Time-based—attacker sends a SQL query to the database, which makes the database wait (for a period in seconds) before it can react. The attacker can see from the time the database takes to respond, whether a query is true or false. Based on the result, an HTTP response will be

generated instantly or after a waiting period. The attacker can thus work out if the message they used returned true or false, without relying on data from the database.

3.2.3 Out-of-band SQLi

The attacker can only carry out this form of attack when certain features are enabled on the database server used by the web application. This form of attack is primarily used as an alternative to the in-band and inferential SQLi techniques.

Out-of-band SQLi is performed when the attacker can't use the same channel to launch the attack and gather information, or when a server is too slow or unstable for these actions to be performed. These techniques count on the capacity of the server to create DNS or HTTP requests to transfer data to an attacker.

3.3 SQL queries

SQL is a standardized language used to access and manipulate databases to build customizable data views for each user. SQL queries are used to execute commands, such as data retrieval, updates, and record removal. Different SQL elements implement these tasks, e.g., queries using the SELECT statement to retrieve data, based on user-provided parameters.

A typical eStore's SQL database query may look like the following:

```
SELECT ItemName, ItemDescription
FROM Item
WHERE ItemNumber = ItemNumber
```

From this, the web application builds a string query that is sent to the database as a single SQL statement:

```
sql_query= "
SELECT ItemName, ItemDescription
FROM Item
WHERE ItemNumber = " & Request.QueryString("ItemID")
```

A user-provided input http://www.estore.com/items/items.asp?itemid=999 can then generates the following SQL query:

```
SELECT ItemName, ItemDescription
FROM Item
WHERE ItemNumber = 999
```

As you can gather from the syntax, this query provides the name and description for item number 999.

3.4. SQL Injection Attacks

SQL Injection Attacks (SQLIA's) are performed by uploading maliciously designed inputs, for example interactive websites, to database-drive programs. Applications then use these inputs to create dynamic SQL queries and have the potential to modify the query's semitone structure due to lack of control and separation of data in SQL. The various SQLIA techniques that attackers use are based on the multiple variations that SQL provides in statement structures. They also frequently benefit from additional features, particularly DBMS implementations, especially the Microsoft SQL Server.

They pursue various goals at different points, from enabling the use of other techniques (escalation of SQLIA) to actually extracting data from databases. The resulting threats are complex, ranging from computer fingerprints to Denial-of - Service (DoS) and data theft sensitive.

Consequently, SQL Injections Attacks jeopardize the confidentiality, integrity and availability of data and database functionality, their hosting systems and their dependent applications, and as such require considerable attention from application developers and effective prevention solutions to be implemented.

Firstly, the attacker "goes out" of the text field by starting its input with a single quote. In doing so, he eliminates the login-side SQL Where clause, allowing SQL control code to be inserted right into the application. No username, for example, is null, and the first argument should always compare with false. To circumvent this problem, the attacker inserts the expression OR 1=1 which will always evaluate to true-this is called a tautology. Next, the — (double dash) operator marks the start of statements which causes the SQL parser to ignore where the PWD field clause is in. Consequently, the resulting meaning of the modified SQL query is equivalent to "select all users" Thus, the user authentication controlling application logic will authorize the attacker and, in the worst-case scenario, the application might even return an error message containing the data returned by the DBMS, i.e. the user credentials list.

Interception is a scheme that cybercriminal use to steal user credentials using malware. The Graphical Identification and Authentication (GINA) developed by windows was intended to allow legitimate third parties to customize the logon process by adding support for things like authentication with hardware radio-frequency identification (RFID)

tokens or smart cards. However, malware authors take advantage of this third-party support to load their credential stealers.

3.5 SQLI Prevention and Mitigation

There are several effective ways to prevent SQLI attacks from taking place, as well as protecting against them, should they occur.

The first step is input validation (a.k.a. sanitization), which is the practice of writing code that can identify illegitimate user inputs.

While input validation should always be considered best practice, it is rarely a foolproof solution. The reality is that, in most cases, it is simply not feasible to map out all legal and illegal inputs—at least not without causing a large number of false positives, which interfere with user experience and an application's functionality.

For this reason, a web application firewall (WAF) is commonly employed to filter out SQLI, as well as other online threats. To do so, a WAF typically relies on a large, and constantly updated, list of meticulously crafted signatures that allow it to surgically weed out malicious SQL queries. Usually, such a list holds signatures to address specific attack vectors and is regularly patched to introduce blocking rules for newly discovered vulnerabilities.

Modern web application firewalls are also often integrated with other security solutions. From these, a WAF can receive additional information that further augments its security capabilities.

For example, a web application firewall that encounters a suspicious, but not outright malicious input may cross-verify it with IP data before deciding to block the request. It only blocks the input if the IP itself has a bad reputational history.

Imperva cloud-based WAF uses signature recognition, IP reputation, and other security methodologies to identify and block SQL injections, with a minimal amount of false positives. The WAF's capabilities are augmented by Incap Rules—a custom security rule engine that enables granular customization of default security settings and the creation of additional case-specific security policies.

Our WAF also employs crowdsourcing techniques that ensure that new threats targeting any user are immediately propagated across the entire user-base. This enables rapid response to newly disclosed vulnerability and zero-day threats.

In-Text Question: Give SQL injection example

An attacker wishing to execute SQL injection manipulates a standard SQL query to exploit non-validated input vulnerabilities in a database. There are many ways that this attack vector can be executed, several of which will be shown here to provide you with a general idea about how SQLI works.

For example, the above-mentioned input, which pulls information for a specific product, can be altered to read http://www.estore.com/items/items.asp?itemid=999 or 1=1.

```
SELECT ItemName, ItemDescription
FROM Items
WHERE ItemNumber = 999 OR 1=1
```

And since the statement 1 = 1 is always true, the query returns all of the product names and descriptions in the database, even those that you may not be eligible to access.

Attackers are also able to take advantage of incorrectly filtered characters to alter SQL commands, including using a semicolon to separate two fields.

For example, this input http://www.estore.com/items/iteams.asp?itemid=999; DROP TABLE Users would generate the following SQL query:

```
SELECT ItemName, ItemDescription
FROM Items
WHERE ItemNumber = 999; DROP TABLE USERS
```

As a result, the entire user database could be deleted.

Another way SQL queries can be manipulated is with a UNION SELECT statement. This combines two unrelated SELECT queries to retrieve data from different database tables.

For example, the input http://www.estore.com/items/items.asp?itemid=999 UNION SELECT user-name, password FROM USERS produces the following SQL query:

```
SELECT ItemName, ItemDescription
FROM Items
WHERE ItemID = '999' UNION SELECT Username, Password FROM Users;
```

Using the UNION SELECT statement, this query combines the request for item 999's name and description with another that pulls names and passwords for every user in the database.

4.0 Self-Assessment Exercise(s)

1. State Accelion Attack flow

Answer

According to a report commissioned by Accellion, the combination SQLi and command execution attack worked as follows:

- Attackers performed SQL Injection to gain access to document_root.html, and retrieved encryption keys from the Accellion FTA database.
- Attackers used the keys to generate valid tokens, and used these tokens to gain access to additional files
- Attackers exploited an operating system command execution flaw in the sftp_account_edit.php file, allowing them to execute their own commands
- Attackers created a web shell in the server path /home/seos/courier/oauth.api
- Using this web shell, they uploaded a custom, full-featured web shell to disk, which included highly customized tooling for exfiltration of data from the Accellion system. The researchers named this shell DEWMODE.
- Using DEWMODE, the attackers extracted a list of available files from a MySQL database on the Accellion FTA system, and listed files and their metadata on an HTML page
- The attackers performed file download requests, which contained requests to the DEWMODE component, with encrypted and encoded URL parameters.
- DEWMODE is able to accept these requests and then delete the download requests from the FTA web logs.

This raises the profile of SQL injection attacks, showing how they can be used as a gateway for a much more damaging attack on critical corporate infrastructure.

5.0 Conclusion

You have learnt from this unit an SQL injection the types of SQL Injection, SQL Queries and SQL Injection Attacks. You will learn SQLI prevention and mitigation.

6.0 Summary

As you have learnt in this unit, an SQL injection the types of SQL Injection, SQL Queries and SQL Injection Attacks. You will learn SQLI prevention and mitigation..

7.0 References/Further Reading

Anley, C. (2022) Advanced SQL Injection in SQL Server Applications. White paper, Next Generation Security Software Ltd.

Anley, C. (2023) More Advanced SQL Injection. White paper. Next Generation Security Software Ltd., 2002.

Boyd, S. W and Keromytis, A. D. (2024) SQLR and: Preventing SQL injection attacks. In Proceedings of the 2nd Applied Cryptography and Network Security (ACNS) Conference, pages 292-302.

Christensen, A. S., M0ller, A. and Schwartzbach. M. I. (2020) Precise analysis of string

expressions. In Proc. 10th International Static Analysis Symposium, SAS '03, volume 2694 of LNCS, pages 1-18. Springer-Verlag.

MODULE 3 ATTACKING FILE UPLOAD FUNCTIONALITY

Module Introduction

In Module 2, you have learned how to conduct basic static and dynamic analysis of malware using static code analysis, in Unit 2, and dynamic program tracing techniques in Unit 3. In this module, I will take you through the advance concepts of malware analysis such as program pointer variable analysis and program data flow analysis.

This module is classified into the following three (3) units:

Unit 1 Executing Remote Code through Malicious File Upload
Unit 2 Components with Known Vulnerabilities
Unit 3 Insufficient Logging and Monitoring

In each unit, I will explore a particular topic in detail and highlight self-assessment exercises at the end of the unit. Finally, I highlight resources for further reading at the end of each unit.

UNIT 1 EXECUTING REMOTE CODE THROUGH MALICIOUS FILE UPLOAD

Contents

1.0 Introduction

In this unit the Executing Remote Code Through Malicious File Upload will be discuss to starting from definition of Remote Code Execution, understanding the Risks, File Size Restrictions and File Upload Attacks.

2.0 Intended Learning Outcomes (ILOs)

BY the end of this unit, you will understand how to perform pointer analysis on a malware

3.0 Main Content

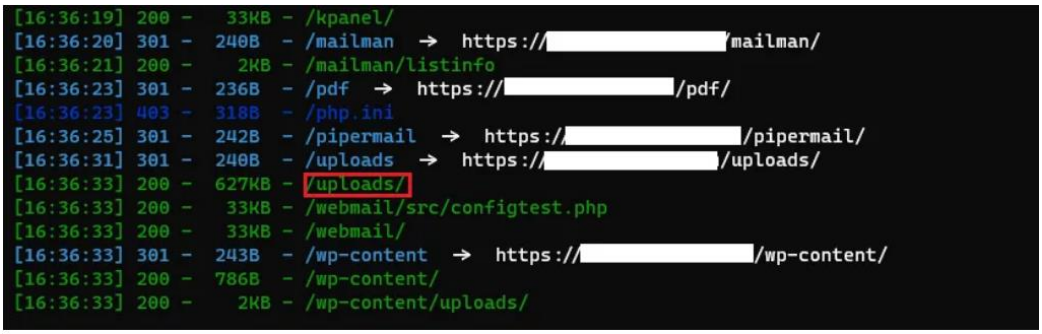
3.1 Define Remote Code Execution

Remote code execution (RCE) refers to the ability of a cyber attacker to access and make changes to a computer owned by another, without authority and regardless of where the computer is geographically located. RCE allows an attacker to take over a computer or a server by running arbitrary malicious software (malware).

Found a target using google dorks which having a responsible disclosure program. Let’s consider the target as abc.com the target website. During recon, I found that the target company owns 7 subdomains and each of them is integrated with the parent domain abc.com with some functionalities.

So, I quickly started doing directory brute-forcing to figure out the directories and guess what the website is fully made up of PHP, so if I found a way to upload a PHP file I can get a reverse shell. The website having a career page which we can upload pdf file as a resume.

At the time of my recon, I found an uploads folder which contains all the uploaded documents of peoples who applied for jobs through this career page.



3.2 Understanding the Risks

Before we dive into the solutions, it's crucial to grasp the potential risks associated with file uploads. The primary concerns include:
Malware Uploads: If an attacker manages to upload a malicious file, they could potentially harm the server, other files, or even users who download and execute the file.

File Inclusion Vulnerabilities: Improper handling of file uploads can lead to remote code execution (RCE) vulnerabilities, where attackers execute arbitrary code on the server.

Storage Consumption: Unrestricted file uploads can lead to denial of service (DoS) attacks by exhausting the server's storage capacity.

Understanding these risks is the first step toward securing your Flask application against file upload vulnerabilities.

3.2.1 Risk Factors

- The impact of this vulnerability is high, supposed code can be executed in the server context or on the client side. The likelihood of detection for the attacker is high. The prevalence is common. As a result, the severity of this type of vulnerability is high.
- It is important to check a file upload module's access controls to examine the risks properly.
- Server-side attacks: The web server can be compromised by uploading and executing a web-shell which can run commands, browse system files, browse local resources, attack other servers, or exploit the local vulnerabilities, and so forth.
- Client-side attacks: Uploading malicious files can make the website vulnerable to client-side attacks such as XSS or Cross-site Content Hijacking.
- Uploaded files can be abused to exploit other vulnerable sections of an application when a file on the same or a trusted server is needed (can again lead to client-side or server-side attacks)

- Uploaded files might trigger vulnerabilities in broken libraries/applications on the client side (e.g. iPhone Mobile Safari LibTIFF Buffer Overflow).
- Uploaded files might trigger vulnerabilities in broken libraries/applications on the server side (e.g. Image Magick flaw that called Image Tragick!).
- Uploaded files might trigger vulnerabilities in broken real-time monitoring tools (e.g. Symantec antivirus exploit by unpacking a RAR file)
- A malicious file such as a Unix shell script, a windows virus, an Excel file with a dangerous formula, or a reverse shell can be uploaded on the server in order to execute code by an administrator or webmaster later – on the victim's machine.
- An attacker might be able to put a phishing page into the website or deface the website.
- The file storage server might be abused to host troublesome files including malwares, illegal software, or adult contents. Uploaded files might also contain malwares' command and control data, violence and harassment messages, or steganographic data that can be used by criminal organizations.
- Uploaded sensitive files might be accessible by unauthorized people.
- File uploaders may disclose internal information such as server internal paths in their error messages.

3.3 File Size Restrictions

In the realm of web development, allowing users to upload files can significantly enhance the functionality of your web application. However, it can also open it up to a plethora of security risks. When improperly handled, file uploads can serve as a gateway for attackers to inject malicious files, leading to potential data breaches or server compromises. However, with the right precautions in place, you can mitigate these risks and secure your Flask application against such vulnerabilities. This blog post delves into secure file upload practices in Flask, focusing on filtering and validation techniques that safeguard your application.

Limiting the size of uploaded files is another critical step in securing file uploads. This can prevent attackers from overwhelming your server's storage with excessively large files or launching DoS attacks. Flask allows you to set a maximum file size limit using the MAX_CONTENT_LENGTH configuration:

app.config ['MAX_CONTENT_LENGTH'] = 16 * 1024 * 1024 # 16 megabytes

Attempting to upload a file larger than the specified limit will result in a 413 Request Entity Too Large error from Flask. Since Flask applications often have a web server sitting in front of them, it is a good idea to also configure this limit for the web server itself.

For example, nginx and Apache have configuration options that specify the maximum allowed size of a client's request body, thus providing an additional layer of defense against large file uploads and other large payloads. Configuring these limits ensures that oversized files are rejected before they even reach the application, enhancing security and efficiency.

For example, nginx and Apache have configuration options that specify the maximum allowed size of a client's

3.4 File Upload Attacks

File upload is one of the most common functionalities applications has to offer. This functionality, however, is implemented in many different forms based on the application's use case. While some applications only allow uploading a profile picture and support only image-related extensions, on the other hand, some applications support other extensions based on their business case. Storing and retrieving these files on the server-side is again a huge task and required to be handled with caution.

Due to the involved complexity and level of caution that is required to implement a file upload functionality, this becomes one of the interesting attack vectors and can open doors to multiple critical security vulnerabilities such as Remote Code Execution. In this blog series, we will be focusing on various File Upload Attacks, Bypasses, and some robust mitigation around them from both an attacker's well a defender's perspective.

In the first part of the file upload attack series, we will look at an attack surface that one gets when there's a file upload functionality and we will focus on some of the interesting file upload attacks.

The below mindmap gives a picture of various attacks that are possible when an application implements File Upload Functionality. This series of blogs will be based on this mindmap itself.

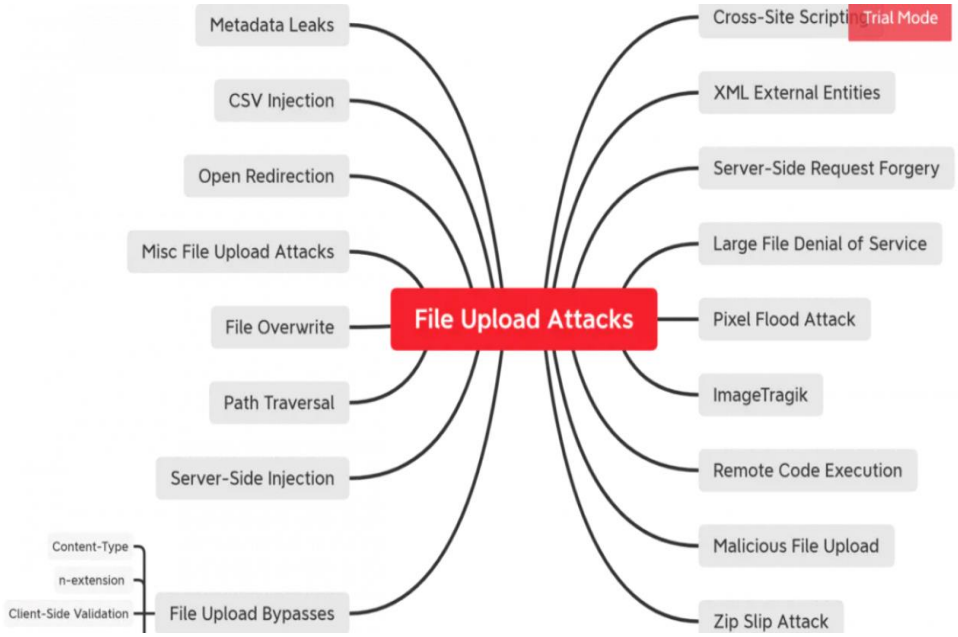


Figure 3.1 File Upload Attack

In-Text Question: 1. What do understand by the term Remote Code Execution?

Answer

One of the most interesting attacks that come into mind whenever there is a file upload functionality is Remote Code Execution. There are several ways to execute a code execution with malicious files, one of the most common is to upload a shell and gain further access. Let's assume a scenario where a PHP based application provides a file upload functionality and stores the files as it is on the server which can later be accessed.

Enumerate the steps involve in achieving remote code execution,

Answer

1. Create a PHP shell or use an existing shell.
2. Upload the shell (bypassing restrictions, if any)
3. After successful upload, navigate to the shell path, for example, https://testweb.com/files/shell.php to see if it is accessible.
4. If the shell is accessible, based on how the shell gets executed, attempt for the shell execution, for example, https://testweb.com/files/shell.php?cmd=whoami.

5.0 Conclusion

In this unit the Executing Remote Code Through Malicious File Upload will be discuss to starting from definition of Remote Code Execution, understanding the Risks, File Size Restrictions and File Upload Attacks.

6.0 Summary

In this unit the Executing Remote Code Through Malicious File Upload has been discussed in order define the areas that include the Remote Code Execution, understanding the Risks, File Size Restrictions and File Upload Attacks.

7.0 References/Further Reading

Smaragdakis, Y., & Balatsouras, G. (2015). Pointer analysis.

Foundations and Trends® in Programming Languages, 2(1), 1-69.

Marshall, D. (2019). Pointers. Retrieved fromhttps://users.cs.cf.ac.uk/Dave.Marshall/C/node10.html

Wei L. (2021 Pointer Analysis. Lecture 2. [Lecture PowerPoint slides]. Retrieved fromhttp://web.cs.iastate.edu/~weile/cs513x/2.PointerAnalysis.pdf

Sikorski, M., & Honig, A. (2022). Practical malware analysis: the hands-on guide to dissecting malicious software. no starch press.

UNIT 2 WITH KNOWN VULNERABILITIES

Contents

1.0 Introduction

In this unit Components of known Vulnerabilities, how to prevent known vulnerabilities and how to identify known vulnerabilities.

2.0 Intended Learning Outcomes (ILOs)

By the end of this unit, you will learn the Components of known Vulnerabilities, how to prevent known vulnerabilities and how to identify known vulnerabilities.

3.0 Main Content

3.1 Components of known Vulnerabilities

Top 10 OWASP describes the term components as a very broad term. It can either be a full piece of software that our target relies on from a third party such as "It's me" which is an identity confirmation service or it can be a small dependency in a script that is hidden somewhere far away.

If we talk about A9: Using Components with Known Vulnerabilities we are often talking about either outdated piece of software or software that is not actively maintained anymore. Usually, well-maintained software will attempt to fix any issues as they arise but often pushing out those updates can be cumbersome and users of that component can be reluctant to update.

These issues are very hard to fix as they often rely on the human psyche to be lazy and careless. They have to be fixed at the source of the issue and they have to be fixed properly but often finding a root cause and implementing a proper update strategy can be expensive which is why many companies opt to skip on what they perceive as a money sink, instead opting to update only when things break if they do not.

We can implement a strategy as a provider of these components to force our users to update but that practice is not considered very user friendly and is often avoided which is good news for ethical hackers and penetration testers as it gives us material to report as discuss in figure 3.2.

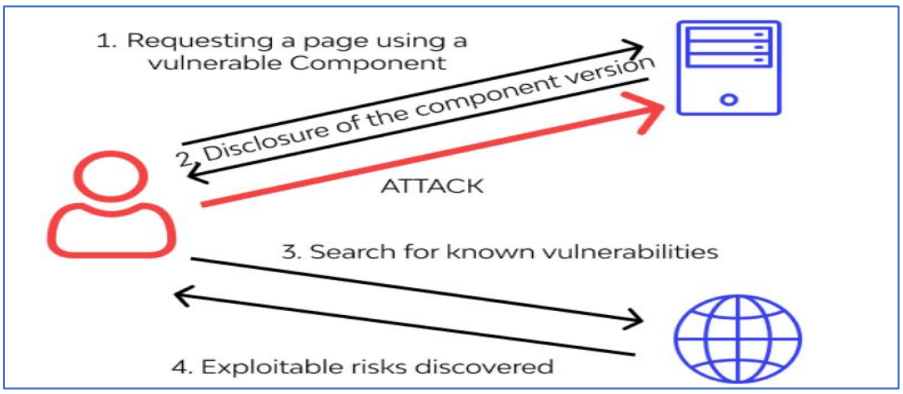


Fig.3.2 Components of known Vulnerabilities

3.2 How to prevent known vulnerabilities

If the client in case of pentesting or target in case of bug bounties has a poor understanding of the dependencies they use and any nested dependency they are much more likely to be vulnerable to A9: Using Components with Known Vulnerabilities. This may be hard to gauge at first but anything is, it really matters to spend time on your application under test and with passing time will come solidifying knowledge.

These dependencies do include more than just what you might think of in the first place. We are of course talking about things that are included in any configuration file of the software but also of things like any database system, API security and even the operating system itself.

To find these A9: Using Components with Known Vulnerabilities vulnerabilities we can use automated scanners but we should be very careful not to outstep our boundaries when we do so. I always prefer to just do passive scanning with no spidering or any requests to speak of. This means that I only analyse the things that I find while I test manually. Automated scanners need to be talked over with the client

and agreed upon as they can cause serious damage to the infrastructure of the client.

I am on the lookout for any version number of software that I can find. Jquery versions, database versions, operating system banners, anything i can find. I will look these version numbers up manually but while I hack I also use a MiTM proxy such as burp to automatically scan the source code of the websites i am visiting. There are several plugins available to look for outdated software of which my favourite is retireJS.

- Shodhan: An excellent site for attackers to trace through the server that has vulnerabilities which are unpatched and is a great way to exploit
- Heartbleed: A great way to check on the SSL vulnerabilities that are a part of OWASP A9. A detailing of the vulnerability will be shown with a simple command line argument and this is best used in integration with Kali Linux
- Burp: Helps in intercepting the packet and trying to get the response for the packet that it tapped. There are other features that are available such as performing brute force attack, version identification of components etc.
- Wireshark: A tool which will enable you to check through the packets and check out the encryption mechanism. If there are components related to TLS in lacking in the site, we could easily be able to identify the vulnerabilities.

3.3 How to identify known vulnerabilities

If the client in case of pentesting or target in case of bug bounties has a poor understanding of the dependencies they use and any nested dependency they are much more likely to be vulnerable to A9: Using Components with Known Vulnerabilities. This may be hard to gauge at first but anything is, it really matters to spend time on your application under test and with passing time will come solidifying knowledge.

These dependencies do include more than just what you might think of in the first place. We are of course talking about things that are included in any configuration file of the software but also of things like any database system, API security and even the operating system itself.

To find these A9: Using Components with Known Vulnerabilities we can use automated scanners but we should be very careful not to outstep our boundaries when we do so. I always prefer to just do passive scanning with no spidering or any requests to speak of. This means that I only analyses the things that I find while I test manually. Automated

scanners need to be talked over with the client and agreed upon as they can cause serious damage to the infrastructure of the client.

I am on the lookout for any version number of software that I can find. Jquery versions, database versions, operating system banners, anything i can find. I will look these version numbers up manually but while I hack, I also use a MiTM proxy such as burp to automatically scan the source code of the websites i am visiting. There are several plugins available to look for outdated software of which my favourite is retireJS.

is said that prevention is better than cure and hence there are various steps that could be taken into consideration when developing the web application. And post the development, the maintenance of the application is very important to make it more resilience from the cyber-attacks.

- Post the development of web application it is recommended to remove all the unwanted files, paths, change the default credentials if any.
- When using the APIs ensure that the server side and client-side components are in sync and for this versioning control is very important
- Ensure the libraries are updated with the latest releases and patches. One need to monitor the various notification that are shared by the vendors and CERT bodies who would inform about the retirement of the components version or upgrade related information.

The web application development usually starts from the development environment which would be trickled down to pre-production and finally to production. We need to ensure that the tools used for developing are genuine and up to date across all environments.

In-Text Question: Give Examples of Using Components with Known Vulnerabilities in the world?

To explain the severity of one of these vulnerabilities we need to look at this from all angles. After all, you want to fix things that are found by your internal scanner. Best practice would have you fix all outdated dependencies but logic dictates that when a bug bounty report comes in with no proveable impact on our infrastructures that we are hesitant to pay out.

After all, what's the harm in keeping an outdated dependency, right? Well rest assured that it can definitely harm your company greatly and

opens them up to A9: Using Components with Known Vulnerabilities! After all, what one hacker was not able to exploit might still be exploitable to another hacker.

One example that is often seen in real life targets would be outdated versions of JS being vulnerable to XSS for example. So, should you report this? Well, it depends. In an ideal world I would want these reports to always be accepted no matter the impact because what might not hurt now might become a giant eyesore later on when new functionality is added.

That being said, if a risk analysis points to the issue being no direct danger in the future it can be understood why sometimes these reports are denied. If there is no impact and there never will be, there is no point in spending money to fix something that hurts nobody.

Another example we can think of is an outdated version of Linux, again it really depends on the exploitability here. If the exploit in question would require us to use a port that was not even open on our target and never would be there is no real impact and the reporting should be held off until further impact can be proven.

A hacktivity report I would also like to draw your attention to is coming from none other than slack. They were using an outdated version of JavaScript which led to the attacker being able to load forms from domains not owned by slack.

4.0 Self-Assessment Exercise(s)

1. What is the goal of conducting a data flow analysis?

Answer

The goal is to associate with variables that have constant values and those that will be used at a that point (program statement) before being redefined.

5.0 Conclusion

In this unit, you have learned Components of known Vulnerabilities, how to prevent known vulnerabilities and how to identify known vulnerabilities. As for hackers themselves, I always recommend we judge our impact extremely carefully and that we do not report any vulnerability like this in bug bounties until impact is clear. For

pentesters it is always advised to at least report these issues as informative but ideally, they should be exploited further if possible and at least investigated carefully.

6.0 Summary

In this unit, you have learned the components of known Vulnerabilities, how to prevent known vulnerabilities and how to identify known vulnerabilities.

7.0 References/Further Readings

Accunetix Team, (2019) “Accunetix web application vulnerability Report, pp 14,19,23,26,

Anantharaman, N. (2021) “To study the effectiveness of digital governance with cyber swachhta kendra initative”, AU FAIT, Vol 5(11)

Shebli, A and Beheshti, B. D. (2022) “A study on penetration testing process and tools” IEE Long Island Systems, Applications and Technology conference (LISAT), Farmingdale, NY, pp 1-7.

Dennis, M., Zena, C., Thaier, H. *2020) “Penetration Testing: Concepts, Attack Methods and Defense Strategies”, Farmingdale, NY, pp. 1-6.

Muller A, Meucci M, Keary E, Cuthbeart D,OWASP Testing Guide, 4.0,pp 165,166.

UNIT 3 INSUFFICIENT LOGGING AND MONITORING

Contents

1.0 Introduction

In this unit, we are going to learn about Insufficient logging and monitoring. Exploitation Insufficient Logging & Monitoring vulnerabilities, Privilege Escalation Techniques for Insufficient Logging & Monitoring vulnerabilities, General methodology and checklist for Insufficient Logging & Monitoring vulnerabilities and Insufficient Logging & Monitoring vulnerabilities.

2.0 Intended Learning Outcomes (ILOs)

At the end of this unit, you would be able to:

- learn about Insufficient logging and monitoring. Exploitation Insufficient Logging & Monitoring vulnerabilities, Privilege Escalation Techniques for Insufficient Logging & Monitoring vulnerabilities, General methodology and checklist for Insufficient Logging & Monitoring vulnerabilities and Insufficient Logging & Monitoring vulnerabilities

3.0 Main Content

3.1 What is Insufficient logging and monitoring?

Insufficient logging and monitoring is a vulnerability that occurs when an application or system does not adequately record and monitor events, making it difficult to detect and respond to security incidents. This can allow malicious actors to carry out attacks without being detected, and can also hinder incident response efforts. To mitigate this vulnerability, it is important to implement robust logging and monitoring capabilities and establish procedures for reviewing and responding to events in a timely manner.

Insufficient Logging & Monitoring (ILM) vulnerabilities can pose a significant threat to the security of an organization's systems and data. Attackers can exploit ILM vulnerabilities to gain access to sensitive information, steal credentials, execute malicious code, and cause other damage. To prevent and mitigate these vulnerabilities, it is important to implement proper logging, alerting, and monitoring, analyze access and privilege levels, apply security patches and updates, conduct regular security assessments, train staff, and use security technologies. By following these best practices, organizations can reduce the risk of ILM vulnerabilities and improve their ability to detect and respond to potential security incidents.

3.2 Exploitation Insufficient Logging & Monitoring vulnerabilities

Data theft: Attackers can exploit insufficient logging and monitoring to steal sensitive information such as login credentials, personal data, and financial information from systems without being detected.

Malware attacks: Attackers can use malware to exploit insufficient logging and monitoring vulnerabilities to gain access to systems and carry out malicious activities such as stealing data, conducting DDoS attacks, and distributing spam.

Privilege escalation: Attackers can exploit insufficient logging and monitoring to gain elevated privileges on a system, which enables them to carry out more damaging attacks.

Denial of service: Attackers can exploit insufficient logging and monitoring to conduct denial of service attacks that overload a system and cause it to crash or become unresponsive.

Advanced persistent threats (APTs): APTs are targeted attacks that use sophisticated techniques to gain access to systems and remain undetected for extended periods. Attackers can use insufficient logging and monitoring to cover their tracks and remain hidden.

3.3 Privilege Escalation Techniques for Insufficient Logging & Monitoring vulnerabilities

Privilege escalation techniques for Insufficient Logging & Monitoring vulnerabilities are typically achieved by taking advantage of the fact that security events and activities are not being adequately monitored or logged, which can allow an attacker to remain undetected and perform additional malicious actions. Some common techniques for privilege escalation in this context include:

1. Exploiting blind spots in logging and monitoring: Attackers may attempt to perform actions that they believe will not be detected because they are not being actively monitored. This could include manipulating data or systems in ways that do not trigger alarms or logging events.

2. Injecting false information into logs: Attackers may attempt to inject false information into logs in order to hide their activities or divert attention away from the actions they are taking. This could involve tampering with log files, altering timestamps, or manipulating other metadata.

3. Using administrative privileges: If an attacker is able to gain administrative privileges on a system, they may be able to bypass logging and monitoring restrictions and perform actions without leaving a trace.

4. Abusing legitimate user accounts: Attackers may attempt to gain access to legitimate user accounts with elevated privileges and use them to perform malicious activities. By using legitimate accounts, attackers can evade detection and appear as if they are conducting normal activities.

Overall, the key to preventing privilege escalation through insufficient logging and monitoring vulnerabilities is to ensure that all activity on a system is being properly monitored and logged, and to implement

mechanisms that prevent attackers from manipulating or injecting false information into those logs.

3.4 Privilege Escalation Techniques for Insufficient Logging & Monitoring vulnerabilities

1. Identify areas where logging and monitoring are necessary: Identify critical systems, services, applications, and data that require logging and monitoring. This will help you focus your efforts and resources on the most important areas.
2. Define logging and monitoring requirements: Determine what data should be logged and monitored, how frequently, and by whom. This will help ensure that all necessary activities are captured in the logs and that relevant events are flagged for follow-up.
3. Implement logging and monitoring solutions: Choose appropriate tools and systems for logging and monitoring, such as security information and event management (SIEM) systems, intrusion detection and prevention systems (IDPS), and network monitoring tools. Configure these tools to capture and store the required data.
4. Test the logging and monitoring solutions: Test the logging and monitoring solutions to ensure that they are capturing all necessary data and that the data is being stored and analyzed correctly. This will help ensure that you can detect and respond to security incidents effectively.
5. Develop response procedures: Develop procedures for responding to security incidents that are detected through logging and monitoring. This should include steps for investigating incidents, determining the scope of the incident, and containing and remediating the damage.
6. Monitor and review: Regularly monitor and review your logging and monitoring solutions to ensure that they are working as expected and capturing all necessary data. This will help you identify areas for improvement and ensure that your security posture remains strong.

3.5 Insufficient Logging & Monitoring vulnerabilities

Insufficient Logging & Monitoring (ILM) vulnerabilities refer to the lack of sufficient logging and monitoring mechanisms in a system, which can result in various security issues. When proper logging and monitoring are not in place, it can be difficult to detect and respond to security incidents, such as unauthorized access, data breaches, and malicious activities.

Here are some of the exploits that can occur as a result of ILM vulnerabilities:

1. Data breaches: Without proper logging and monitoring, it may be difficult to detect a data breach and respond in a timely manner. This can result in sensitive information being leaked, stolen, or sold on the black market.

2. Unauthorized access: Insufficient logging and monitoring can make it difficult to detect when someone has gained unauthorized access to a system or network. This can result in the attacker being able to steal sensitive information or compromise the system in other ways.

3. Malicious activities: If logging and monitoring are not in place, it can be difficult to detect when someone is engaged in malicious activities, such as attempting to steal sensitive information, spreading malware, or engaging in other types of cyber-attacks.

4. Compliance violations: In some industries, regulations require organizations to maintain proper logging and monitoring systems in order to ensure that sensitive information is being handled properly. Insufficient logging and monitoring can result in non-compliance with these regulations, which can result in fines or other penalties.:

If you want to study Insufficient Logging & Monitoring (ILM) vulnerabilities, here are some topics that you should focus on:

- Definition and impact of ILM vulnerabilities: Understand what ILM vulnerabilities are and the impact they can have on the security of an organization. This includes the various exploits that can occur as a result of insufficient logging and monitoring.

- Best practices for logging and monitoring: Learn about the best practices for logging and monitoring, including what events should be logged and monitored, how to identify critical data, and how to establish effective logging and monitoring procedures.

- Tools and technologies for logging and monitoring: Become familiar with the various tools and technologies used for logging and monitoring, including log management systems, security information and event management (SIEM) tools, and intrusion detection and prevention systems (IDS/IPS).

- Compliance requirements: Understand the compliance requirements for logging and monitoring in various industries and how to ensure that an organization remains compliant.

- Threat modeling and testing: Learn how to perform threat modeling and testing to identify potential vulnerabilities and weaknesses in logging and monitoring systems, as well as how to remediate any vulnerabilities that are identified.

- Incident response: Understand the role of logging and monitoring in incident response, including how to use logs to identify and investigate security incidents, and how to develop effective incident response procedures.

- Continuous monitoring and improvement: Learn about the importance of continuous monitoring and improvement of logging and monitoring systems, including regular testing and evaluation of security controls, and the implementation of new technologies and procedures as needed.

- By studying these topics, you can gain a solid understanding of ILM vulnerabilities and how to address them to ensure the security of an organization's data and systems.

4.0 Self-Assessment Exercise(s)

1. How to be protected from Insufficient Logging & Monitoring

Answer

Sigma rules and firewall rules can be useful in preventing and detecting Insufficient Logging & Monitoring (ILM) vulnerabilities. Here are some examples of rules that can be used to block or stop ILM vulnerabilities:Log Retention and Monitoring Rule: This rule requires logging and monitoring of critical events and retention of logs for a certain period. If the logs are not retained, the rule can trigger an alert and block access to the system until the logs are being collected and monitored.

1. Cross-Site Scripting (XSS) Rule: This rule detects and blocks web requests that contain known XSS payloads. The rule can be implemented in a web application firewall or as a network security rule.

2. SQL Injection Rule: This rule blocks web requests that contain SQL injection payloads, preventing attackers from exploiting SQL injection vulnerabilities in web applications.

3. Command Injection Rule: This rule detects and blocks web requests that contain command injection payloads, preventing attackers from executing malicious commands on a system.
4. Remote Code Execution Rule: This rule blocks web requests that contain malicious code, preventing attackers from executing code on a remote system.
5. Credential Stuffing Rule: This rule detects and blocks login attempts with known breached credentials, preventing attackers from using stolen credentials to gain access to a system.
6. Man-in-the-Middle (MitM) Attack Rule: This rule detects and blocks network traffic that contains indicators of MitM attacks, such as intercepted data or forged certificates.

These rules can be implemented in various security systems, including firewalls, intrusion detection and prevention systems (IDPS), and Security Information and Event Management (SIEM) systems, to prevent and detect ILM vulnerabilities. It's important to continuously monitor and update the rules to ensure their effectiveness against new vulnerabilities and attack techniques.

5.0 Conclusion

In this unit, I explored learned about Insufficient logging and monitoring. Exploitation Insufficient Logging & Monitoring vulnerabilities, Privilege Escalation Techniques for Insufficient Logging & Monitoring vulnerabilities, General methodology and checklist for Insufficient Logging & Monitoring vulnerabilities and Insufficient Logging & Monitoring vulnerabilities.

.

6.0 Summary

In this unit, you have learned how I explored learned about Insufficient logging and monitoring. Exploitation Insufficient Logging & Monitoring vulnerabilities, Privilege Escalation Techniques for Insufficient Logging & Monitoring vulnerabilities, General methodology and checklist for Insufficient Logging & Monitoring vulnerabilities and Insufficient Logging & Monitoring vulnerabilities. hackers use Anti-debugging to deter the examination of malware behaviour. You also learned the different techniques used to detect

debugging activity in Microsoft Windows and how to perform certain tasks in order to detect a debugger.

7.0 References/Further Reading

Mark Dowd, John McDonald and Justin Schuh. "The Art of Software Security Assessment". Chapter 2, "Accountability", Page 40. 1st Edition. Addison Wesley. 2002.

Center for Internet Security. "CIS Microsoft Azure Foundations Benchmark version 1.5.0". Sections 3.5, 3.13, and 3.14. 2022-08-16. <https://www.cisecurity.org/benchmark/azure>. URL validated: 2023-01-19.

Microsoft. "Enable and manage Azure Storage Analytics logs (classic)". 2023-01-23. <https://learn.microsoft.com/en-us/azure/storage/common/manage-storage-analytics-logs>. URL validated: 2023-01-24.