Assignment 4 Report
ACME-6

Francesco Aquino (1851954),
Michele Kryston (1844733),
Ralph Angelo Tancio Almoneda (1837040),
Simone Di Valerio (1835412)

Practical Network Defense, Cybersecurity Master Degree
Sapienza University of Rome
Second Semester 2022/2023

Contents

1 Initial Brainstorming

In this assignment a new host named "Ethical" is added to the ACME topology. This host is active and we are supposing that it is running our machine from the Ethical Hacking class with three vulnerabilities for gaining local access and three vulnerabilities to perform privilege escalations.

Our task for this assignment is to propose possible solutions to protect the host from attackers who may be able to exploit the vulnerabilities and access our machine remotely. The proposed solution must not alter the configuration of the host.

We decided to analyse separately all the vulnerabilities and propose solutions for each of them. Then, for each precaution mentioned, we thought about possible tests to perform on the machine, possible because in reality we do not have our Ethical machine in a network.

To protect the network we thought about a deny by default rule and we tried to make the system robust with the introduction of a log policy and with the implementation of different mechanisms so even if the firewall is bypassed there is always a second way to defend the machine.

2 Vulnerabilities

In general to avoid the discovery of vulnerabilities (through e.g. nmap) in a host, it is possible to hide the services running. This can be done for example with the tool **portsentry** that, thanks to iptables, is able to block the attacker executing nmap.

Our Machine has three different vulnerabilities which can be exploited by an attacker to gain remote access. In this section we aim to find solutions and countermeasures in order to protect our machine from attacks.

2.1 Samba Dictionary Attack

One of the users of the system decided to start a Samba service and leave a reverse shell inside to access the host. This vulnerability includes a remote connection to Samba from the attacker. This service requires the login with username and password. Through social engineering the attacker is able to find the username and performing a dictionary attack the password will be easily found. After logging in, it suffices to modify the .reverse.sh file that is normally used by the allowed host by changing the IP in order to execute it and have access to the user's bash.

2.1.1 Proposed Solutions

The solutions we propose are:

- A first implementation of a security countermeasure would be to set a new **firewall rule** so that only authorized hosts or networks can access

Samba. This service (included the .reverse.sh file created by the user) is thought to only allow the legitimate user to access the machine remotely. Therefore, it is not needed to allow other devices to connect to it.

- Another option to restrict access to Samba is to use a proxy, which can provide a blacklisting and whitelisting service. A **forward proxy** in our case receives the packet directed to our machine and can analyse its content. Then it can either authorize the user and forward the requests that come from allowed sources or block unauthorized devices.

- To cope with the dictionary attack, we can set a maximum amount of tentative logins a user can try in a certain period of time and block if the number of attempts exceeds the maximum allowed. This can be done thanks to a **HIPS**. A useful software to implement this countermeasure is **Fail2Ban**. It can scan logs from Samba and block for example through iptables an IP with a lot of failed tentatives and send an email notification. We can also implement in our system a **NIPS** that can discover the presence of a large amount of data by a brute force attack.

- Regarding the password, a best practice must be to choose a password not so easy that can be found by performing a dictionary attack. Several fixes can be applied. First of all, using a **long password** with both upper and lower case, numbers and symbols can lower the chances that the password can be guessed using a brute force attack. Secondly, changing the password frequently may be an additional security measure. It is also possible to check if the password is not present in hacked databases through for example the website "https://haveibeenpwned.com". If needed, a password manager can be used to store and retrieve the password securely.

- The last option may be to constantly check for any changes in the.reverse.sh file and send an alert whenever some changes occur (like changes of the IP addresses). Keeping track of the actions on a specific file can be done using for example a HIDS, and it is possible to send them to an **ACME** like Graylog. However, we suggest not to use log files only to monitor the.reverse.sh file, but to save all actions related to connections with external hosts.

2.2 Object Injection & Path Traversal

Unsanitized PHP and upload filter bypass vulnerability is another type of web server vulnerability that can be exploited. This vulnerability arises due to a lack of proper sanitization in the PHP code, allowing an attacker to manipulate the source code and identify certain PHP classes used across multiple files. One particular class, the Post class in the class.php file, has an oversight that enables it to accept any value as a parameter. By understanding how an object of the Post class is created (PHP serialization), an attacker can forge a new object and pass it through the URI of the website. This manipulation allows

the attacker to gain access to files that should be hidden or restricted. For instance, an attacker may create an object that grants unauthorized access to sensitive system files and after use this object as an URL. By visiting this URL, the PHP interpreter on the server executes the malicious code within the URL. By exploiting this vulnerability, the attacker can access and retrieve the sensitive information stored in /etc/passwd. Moreover, if there is an old username credential in /etc/passwd with an easy password, the attacker can potentially get access to the account and further compromise the system.

2.2.1 Proposed Solutions

Precautions:

- The first precaution to take in a web server would be implementing a secure connection that can guarantee Confidentiality, Integrity and Availability. A valuable solution would be using a **SSL/TLS connection** which provides also authentication of the end points and encrypt the communication.

- Gobuster and Burp Suite are noisy scan tools. In order to prevent the scanning of our website, we can adopt different solutions. If a source address sends packets with a high frequency, on the other side of the host there may be an attacker trying to scan our website and find information about available ports and services, directory structure and so on. In order to prevent the scanning process, we can set some **firewall rules** that block these kinds of addresses or drop their requests if we notice an unusual flow. Moreover, if some open ports are not needed by our service, we can close them and reduce the attack surface.

- Another solution may be using a **proxy** that analyses the incoming requests and takes decisions based on the content of the packets or the flow or the source addresses. It is also possible to provide TLS security thanks to the **SSL forward proxy** that is invisible to the client and can receive, decrypt and analyse the packets directed to the web server. If in the requests there are unusual parameters to access files that should not be retrieved, the proxy can block the user or drop the packets. On the other hand, if in the package everything turns out to be OK, it could simply send the packets to the web server in clear text or even better encrypt them and send.

- Also, an **Intrusion Detection System** (IDS) or even better an **Intrusion Prevention System** (IPS) can help with the recognition of scanning activities and limit the damages. We can use the output of these systems as input to an ACME.

- As already stated, the user just by navigating the website can define the structure of the URL. By accessing the post's pages, the information about the parameters of the class Post can be replaced with an Injected Object

and access sensitive files. In order to hide or limit this leak of information, we can use a solution like the **.htaccess file**. This file allows to rewrite the URIs and hide the real directories and information that are normally showed. Moreover, it can **restrict accesses** or ask for **authentication** for certain directories and files. In this way the attacker will not be able to access files like /etc/passwd just by forging a new object.

- **Logs** are helpful since we can keep track of all incoming requests, the accesses to the files and recent changes. We can then figure whether sensitive files are recently been accessed or modified by authorized or unauthorized users. Logs can be generated in the proxy, the host itself, firewall and also by the IDS and can be sent to an ACME system.

- In this vulnerability, the /etc/passwd file is accessed and even if all the user credentials of the host are encrypted, an attacker can access and crack them. Preventions to take are, first of all, **deleting old users** that do not access the system anymore. For all other users, the suggestion is to frequently **change their passwords** and choose long and strong ones with combinations of upper and lower cases, numbers and symbols. It is also possible to set up user account expiration, for example through the commands usermode and chage.

- Finally, the ssh connection can be blocked using solutions like **firewalls**, closing the ssh port if not needed by other users, or restricted to certain groups of users. For users allowed to connect via ssh, using **user's private and public keys** instead of a login with passwords may be a more secure solution and also disable root user.

2.3 Unsanitized PHP & Upload Filter Bypass

This vulnerability is exploitable because of some misconfigurations and oversights by the developer of the site. An attacker intercepting calls with e.g. Burp Suite may notice that in addition to the username and password also the db parameter is passed to the database, and it is possible to inject it and replace the database host address (thanks also to the PDO exceptions). Thus the attacker, by creating a database on his own machine, will capture the packet that is used to retrieve information from the database (containing username in clear text and hashed password of the database developer) and luckily he will easily be able to decrypt it thanks to the fact that the admin used a simple password (through e.g. John The Ripper). Then, by capturing the packages with Wireshark, he will be able to recreate the identical database and log in to the site using his database. After logging in, he will notice the presence of a form where it is possible to upload images to the site, and by testing he will realize that it is also possible to upload a php file (.phar) which he will use to create his reverse shell. He will use Gobuster to find where the files are uploaded and also how they are renamed (thanks to random words in a wordlist) and finally he will use his uploaded reverse shell to connect to the web server.

2.3.1 Proposed Solutions

The possible solutions are very similar to the previous vulnerability:

- One possible way of avoiding and intercepting these types of attacks is the use of **IDS/IPS**, preferably **behaviour-based**. As far as the host itself (web server) is concerned, it is possible to install a **HIDS** that takes care of intercepting any unusual events or actions, such as abnormal input packets or upload of PHP files. As for the network, we could install a **NIDS** that monitors packets and message flow.
- A rule could also be added to the **firewall** or even better choose as default policy drop to not permit to send the packet with the database credentials to the user who has created his own database to simulate the attack.
- To read the attacker's input packets, one could use a **SSL reverse proxy** that pretends to be the web server itself and thus has a way of decrypting the packets, reading the content, encrypting them again and sending them back to the web server itself to execute.
- Translate the URL of the web server directories in a different way to hide the internal structure of the site and make it more difficult to find the files with Gobuster (**.htaccess file**).
- **Block the execution** of files (in our case PHP) in the upload folder of the web server (**.htaccess file**).
- **Password hardening** for the developer user of the database.

3 Privilege Escalation

Let us suppose that an attacker gained remote access on our machine by exploiting one of the above mentioned vulnerabilities. Once inside our machine, the attacker can perform a Privilege Escalation by exploiting three new vulnerabilities. In this section we want to propose possible solutions and countermeasures to avoid this scenario.

3.1 Altered Crontab Configuration

This privilege escalation is caused by an oversight of a privileged user while running tests on scripts with automatic execution on Crontab. The user added to the etc/crontab directory the /tmp folder, then created an automated script (logger.sh) to create a backup of the logs which is executed by root. The oversight consists in not typing the full path of the script but just the file name, which is then placed in the /usr/sbin folder. This mistake makes Crontab to check for the presence of that file in its own path.

An attacker just by checking the execution of the scripts in Crontab will notice the presence of the /tmp folder in the PATH and the execution of logger.sh

without the absolute path. With the command "find" the attacker will find the logger.sh file in the /usr/sbin folder located after /tmp folder.

It suffices for the attacker to create a new logger.sh file inside /tmp folder which will be executed by root and perform a privilege escalation creating a reverse shell.

3.1.1 Proposed Solutions

Solutions:

- Changing the **permission** to access or modify certain folders may be a solution to prevent any local users without root privileges to perform sensitive tasks. In this case, since the logger.sh is executed on Crontab by root, we can prevent a local user to create a new logger.sh file executable by Cronjob which can contain a reverse shell or other malicious code by editing the /tmp directory permission or by editing the /etc/crontab file permission.
- A **logging system** in this case is needed to periodically keep track of all the processes run by Crontab and check whether suspicious cronjobs are carried on. Moreover, it is also useful to check periodically any changes in the directories and find out when a new file is created or an existing file is modified. When this scenario occurs, it is important to have the full picture of the situation: information like the user who made the changes, the files changed or created and the timestamp are necessary to perform an efficient incident forensics whenever an attacker is able to perform a privilege escalation (AAA rule).
- An **HIDS** that is able to notice the execution of sudo commands or opening a root shell.

3.2 Horizontal SUID Escalation & Python2 Vulnerability RCE

In order to facilitate the creation of logs, SUID is set to the logsave command with the developer owner. With the help of tools like linenum.sh the attacker can check for unusual SUIDs and will find logsave.sh. With GTFOBINS it is possible to get a shell with developer permissions. The attacker can then find public and private keys needed for the connection in ssh in the developer's folder and pass them to the attacker's machine to allow remote connection without the password. Lastly, by exploiting a vulnerability of python2 in the backup.py file from root a simple bash from root can be run. This vulnerability in fact allows a remote code execution given by the input function. It's enough to type " __import__('os').system('COMMAND TO RUN')" as input to execute any command.

3.2.1 Proposed Solutions

Possible solutions:

- Also, for this vulnerability, maintaining **logs updated** is a necessary measure to keep track of which files are executed with SUIDs.
- A method to prevent ssh connections from unauthorized hosts is to use **firewall rules** that allow connection with ssh only to specific devices (or IPs). In this way, even if the attacker has the keys to log via ssh, its requests can be blocked.
- In order to filter ssh traffic, we can also use a **reverse proxy** which performs a blacklisting and whitelisting of users that try to connect to our machine. Our proxy receives the requests directed to our machine and allows only the connection from allowed users.
- The last one as always an **HIDS** that keeps track of sudo executions or root shells.

3.3 PAM Backdoor

pam_unix.so is an important assembly library which is responsible for controlling the authentication of the accounts. This vulnerability is caused by a change in the file, where a secret password which allows to access any user on the server is added. Moreover, a backdoor is added in the binary file. An attacker that wants to exploit this vulnerability must have a deep understanding of the Linux system and check for any changes in the files of the server. After a shortlisting of all the files, he will notice that the PAM file, which should not be modified, has been changed. The file is in Assembly, the attacker needs a decompiler like Ghidra to read the C code. The important information are 'pam_sm_authenticate' function, which is responsible for verifying the authenticity of the accesses, and a 20-character string being compared with a variable "p", which is intuitively the password. If strcmp returns the value 0 (secret string equal to the string entered) the backdoor is performed. So the attacker, knowing the password, is able to access all users of the system and also change his user to root.

3.3.1 Proposed Solutions

- A possible solution is through a **SIEM** and the analysis of event logs. When pam_unix.so is accessed or modified, by whom and when (AAA rule).
- Another solution could be the installation of a **HIDS** which is able to detect a strange user account change (su command).
- Monthly/Weekly **checks** of the main system files by comparison with the default ones. This can be done thanks to **file integrity monitoring systems** like **OSSEC**.

4 Proposed Tests

Most of the solutions mentioned above are applicable on more than one vulnerability (such as using firewall or proxies) while other countermeasures are just best practices (like the criteria for a strong password or the suggestion to frequently change it). For this reason, in this chapter we will propose tests for each of the single countermeasures proposed instead of testing each vulnerability, which would imply redundant tests.

In order to test our solution and avoid unwanted connections to our machine we need a new device for testing. It will be used to perform different kind of attacks and send requests to verify how our countermeasures perform. Concerning local vulnerabilities like privilege escalation, our tests will be performed also in our vulnerable machine.

4.1 Firewall Rules

Firewalls that are meant to block incoming connections to services from unauthorized users can be tested using a device that is not allowed to access the machine. If we want to exploit the Samba dictionary attack vulnerability, we can try to connect to Samba from our new device. If the connection is refused by the firewall, our countermeasure is efficient. Regarding the Object injection & path traversal, we try to use tools like Gobuster for scanning . If the scanning does not succeed, our firewall rules, along with other precautions like proxies and IDS, successfully detected this malicious activity. We also use firewalls to block ssh traffic. To test also these rules, we try to connect via ssh to the machine with the testing device. Even if we have the credential, since our host is not allowed to connect, our firewall should be able to block the connection.

4.2 Password Countermeasures

For this test, it is essential to avoid selecting passwords that are easily discoverable through dictionary attacks or other known brute force attacks. We will implement various solutions to enhance password strength. Firstly, we will employ a lengthy password containing a combination of uppercase and lowercase letters, numbers, and symbols while also avoiding common patterns (like "12345678" or "password"). This will significantly decrease the likelihood of successful brute force attacks. Additionally, we will evaluate the impact of regularly changing passwords as an extra security precaution and also user account expiration. If necessary, we will explore the utilization of a password manager to securely store and retrieve passwords, which will remove the problem stemming from the memorization of the complex password. To test and be sure that this countermeasure is respected, we could try to create a new user and see if it is obliged to follow the rules and also force all the users of the system to generate a new password for their accounts.

4.3 Logging

Storing logs is an activity that we proposed for all vulnerabilities. We use firewalls, proxies, IDS devices and the Ethical machine to generate logs to send to an ACME like Graylog for the normalization and display of all information gathered. Most important logs stored are related to the access and/or changes of sensitive and critical files (pam unix.so or Crontab directory, reverse.sh), to the execution of SUIDs files (logger.sh) and information about packets coming from the network, example of useful information are the frequency of incoming requests for certain services, tentative of remote connection via ssh, source of these packets, timestamp and so on.

We will use Graylog to normalize, process and show all information through a dashboard. We can then have an overall picture with relative statistics of all the recent activities in the network (as we have done for the third assignment). We can use logging systems also when performing our tests and check how our countermeasure performed and if the ACME is able to rightly detect them and send alerts.

4.4 SSL/TLS Connection

To enhance security against tools like Gobuster and Burp Suite, implementing SSL/TLS connections is crucial. This involves obtaining a valid SSL/TLS certificate and configuring it on the web server. By enabling HTTPS and redirecting all HTTP traffic to HTTPS, all communication between clients and the server becomes encrypted. It is important to use strong encryption algorithms and protocols, disable insecure protocols, and enable Perfect Forward Secrecy (PFS) to enhance security further. Additionally, implementing HTTP Strict Transport Security (HSTS) ensures that all connections are made over HTTPS, mitigating downgrade attacks. Regularly updating and renewing SSL/TLS certificates, as well as keeping the web server software up to date with security patches, helps maintain a secure environment. By implementing SSL/TLS connections, the confidentiality, and integrity of the communication are strengthened, making it more challenging for tools like Gobuster and Burp Suite to compromise the data. To test this solution, we could simply connect to the website through an external user and see if there is a valid certificate and if HTTPS is correctly implemented.

4.5 Proxies

A proxy server can enhance the security of a connection through various methods. Firstly, it can establish an encrypted connection between the client and the server, encrypting the data before transmitting it (forward proxy). This prevents unauthorized parties from intercepting and deciphering the information. Secondly, a proxy can provide anonymity by masking the client's IP address. By acting as an intermediary, the proxy server presents its own IP address to the server, protecting the client's identity and making it harder for attackers to

directly target them (anonymizer proxies). Additionally, proxies can be configured to filter and control content, blocking malicious websites and restricting access to unauthorized resources (content-filtering proxies). They can also cache frequently requested content, reducing server load and providing an extra layer of security (reverse proxy). Furthermore, proxies can distribute incoming traffic across multiple servers, enabling load balancing and mitigating DDoS attacks. An external user should not notice the presence of the proxy, but might notice the fact that the server (actually the proxy) has started using the HTTPS protocol (SSL reverse proxy implementation). To check if we correctly implemented the proxy to the web server, we could see if the server has an HTTPS connection (SSL reverse proxy) and if it is able to generate logs that are received by our ACME device (notice the fact that it is able to decrypt HTTPS packets).

4.6 Intrusion Detection Systems

We aim to use both Host Intrusion Detection/Prevention Systems and Network Intrusion Detection/Prevention Systems. HIDS will be placed on our Ethical Machine and its task is to detect any suspicious activities performed on the host. Such activities involve continuous tentatives of logins, changes and accesses to important files and unusual files run with SUIDs. NIDS on the other hand must detect suspicious network flows that may be generated by DDoS attacks, scanning tools or tentatives of remote connections like via ssh or to Samba service. A combination of Signature-based and Behavior-based approaches can be used. Signature-based approach can detect common attack patterns, like scanning with noisy tools or dictionary attacks. On the other hand, we can train try to exploit our vulnerabilities during our test phase and use the outcomes' information to train a Behavior-based system. If we try an attack, our IDS should detect it and then raise an alarm. An external user with malicious intentions should notice the fact that he is being blocked by the server, and the developer of the system should be able to see the event logs generated by our IDSs/IPSs in the ACME system.

4.7 SSH

As stated earlier, we must limit the problems associated with SSH. Therefore, we should not allow the connection as root user, and we should switch off the ability to log in with a password, but only with a private key. These changes can quickly be implemented with the SSH configuration file and can be easily tested by connecting with an external user.

4.8 Permissions and.htaccess

As we have stated, another security method concerns the permissions to folders, such as the image upload folder of the web server. We can restrict the execution of files within it thanks to.htaccess. This configuration file format can also be used to mask the URLs of the web server folders. To test the correct operation,

an external user could try to run gobuster on the web server, and he would notice that he is not able to find the folders, and even if he could, he would not be able to execute the uploaded PHAR files. Regarding the permissions of files and directories, if we for example limit the creation of files to the /tmp folder only to root (to avoid crontab privilege escalation), the user will no longer be able to create new files in it so he will not be able to execute the privilege escalation.

5 Final remarks

With this assignment we were able to define the critical points of a machine in our network topology and based on the vulnerability, design possible solutions to protect our system from attackers. In order to do so, we first had to study the theoretical solutions for Network Hardening, Intrusion Detection and Prevention Systems and Best Practices to enforce the security of the machine explained during our lessons. After that, we had to make various considerations based on different scenarios, available sources and requirements in order to make the right decisions and protect the machine from attacker who try to exploit the vulnerabilities detected. The process consisted on studying and understanding the concept, and then learn how these concepts may be practically applied in our case. As we know, implementing such security also has cons, such as system slowdown and the cost of maintaining it, but for specific networks, robust security is necessary and not an option.