BIS Project - The FITfather 2020/2021

http://fit.vut.cz

Information System Security (BIS) Documentation - The FITfather

Šimon Stupinský, xstupi00@fit.vut.cz,
Faculty of Information Technology, Brno University of Technology

1. Introduction

The aim of this project is to obtain as many secrets as possible hidden on private servers in the internal network **bis.fit.vutbr.cz**. The following chapters describe the mapping of the internal network and the procedures for obtaining individual secrets.

2. Network Mapping

After the connection to the server **bis.fit.vutbr.cz**, was by using the command `ip address` determined, that the computer is located in the following network: **192.168.122.0/24**. Subsequently, with the usage of command `nmap -p- 192.168.122.0/54` was found the following computers and their open ports:

PORT	SERVICE	PORT	SERVICE
22/tcp	ssh	80/tcp	http
111/tcp	rpcbind	887/tcp	iclcnet_svinfo
888/tcp	accessbuilder		

Table 1. `Nmap` scan report for **s1 (192.168.122.234)**.

PORT	SERVICE	PORT	SERVICE
22/tcp	ssh		

Table 2. `Nmap` scan report for **s2 (192.168.122.5)**.

PORT	SERVICE	PORT	SERVICE
22/tcp	ssh		

Table 3. `Nmap` scan report for **s3 (192.168.122.55)**.

PORT	SERVICE	PORT	SERVICE
22/tcp	ssh	80/tcp	http
3306/tcp	mysql		

Table 4. `Nmap` scan report for **s4 (192.168.122.211)**.

PORT	SERVICE	PORT	SERVICE
21/tcp	ftp	22/tcp	ssh
111/tcp	rpcbind	613/tcp	hmmp-op

Table 5. `Nmap` scan report for **s5 (192.168.122.36)**.

3. Secrets Collecting

This chapter describes the procedures to obtain the individual secrets.

3.1 Secret A

In the beginning, the directory `.ssh` was examined to obtain some useful information to login to some available server. Therefore, the file with name `config`, in this directory, was also examined and it contains the following content:

Host	Hostname	User
s1	192.168.122.234	server1
s2	192.168.122.5	server2

Table 6. The content of the file `.ssh/config` at **bis.fit.vutbr.cz**.

According to the obtained information, was performed the command *ssh server1@192.168.122.234*, which executed the successful login at **server1**, without having to enter a password. To explore the unknown environment on this server, a command `ls -al` was executed, and it showed the existence of directory with name `.secret`. This directory included two files. First of them, was the *bash* script, which generates the required **secret** according to the current time and hash of the given *decrypted* text. The second file with name **cipher** included the *ciphertext* to *decoding* and result of this process serves as the input for generating *bash* script.

In the next step, it was necessary to find out what *cipher* algorithm was used to generate found *ciphertext*. An online tool `BOXENTRIQ`[1] was used for this, specifically, its available feature *Cipher Identifier and Analyzer*, which detects that it was used the *Columnar Transposition Cipher*. Subsequently, with using of the next feature of `BOXENTRIQ` tool, namely *Columnar Transposition Cipher Tool*, was *decrypted* the found

[1] www.boxentriq.com

ciphertext. In more detail, was used the feature of *Auto Solve (without key)*, where was set the length of the key from range **1 – 10** and maximal number of results to **1000**. The highest score achieved the text, which hid the names of the countries and the key has the length of **5**:

- **Output**: ALGERIA RWANDA MONTENEGRO DOMINICA SIERRA POLAND
- **Key**: abcde

In the last step, the `bash` script was run with the *decrypted* text as its first argument and the secret was printed to the standard output.

3.2 Secret D

Based on the information obtained in the `.ssh/config` file at the **server1** (see Table 6), the exploration continued at **server2**. The command `ssh server2@192.168.122.5` successfully logged in to this server without entering a password. Followed by a survey of the contents of this server, which discovered the presence of an executable file `secret_app`. Of course, the launch of this binary file following, which printed out the message: "`Weclome in secret application!!`", and subsequently waited to some entered input. Assuming that some specific input – e.g. server2, bis, welcome, etc. – will cause the active operation of the application, various possible inputs have been tried, but without success. Subsequently, this binary file was analysed using the command `objdump -d secret_app`, which disassembled it to the assembly code. However, even this step did not reveal the searched secret. Subsequently, various directories on this server were searched, with the intention of finding some clues to the given binary file. After a failed search, has been performed the peek into the binary file itself, where was surprisingly hidden the secret, which was located in front of string which prints this application.

3.3 Secret E

When researched some clues about the binary file at the **server2**, the following information was found in the file `.ssh/config`:

Host	**Hostname**	**User**
s3	192.168.122.55	joe
s4	192.168.122.211	server

Table 7. The content of the file `.ssh/config` at **server2** (s2).

Therefore, the next step was to log on to the **server3** with the following command: `ssh joe@192.168.122.55`. However, this command already required a password and the cracking of password could begin. Firstly, some intuitive passwords were attempted, but without the successful login: e.g.: joejoe, iamjoe, server2, bis, bis2020, ilovebis, etc. Another attempt led to the search for most frequently used passwords, and the following page from Wikipedia[2] helped. A list from the last year *2019* was chosen and the available passwords were tested one by one. The successful login was recorded only at the very end of the list when entering the password with the number **24: password1** meant a successful login to the **server2**. The first command in the unknown environment of the new server was again `ls -al`, which immediately revealed the presence of the file `secret.txt`. After the printing out of the content of this file was found this secret.

3.4 Secret I

According to the information obtained at network mapping, in Table 5 can be seen, that **server5** has an open port **21** with *ftp* service. The access credentials were required when trying to connect to this server via FTP with command `ftp 192.168.122.36`. Firstly, was tried to login as user *anonymous* and with the empty password, and such login was successful. After the successful connection, the remote directory was examined and various other attempts were made to obtain some useful information, but without success. However, at login to this server via listed command, was printed out the following notice: `220 (vsFTPd 2.3.4)`. The next step led to the internet, where some information about the version of the ftp service was tried to obtain. This step was significant to obtain this secret because the information about the vulnerability in this version was come across when searching. After a brief follow-up, we found out more accurate information about this vulnerability[3].

This version (*vsFTPd 2.3.4*) contains such vulnerability, which allows logging to the server with a username that contains a sub-string **':)'**, and also subsequent entering an empty password for a successful login. With using the same command (`ftp 192.168.122.36`) as in the first login attempt to the **server5**, this time the username **bis:)** and blank password were used to log in. After the successful login, the following message was printed out: `220 Opened port 51503, take a look ;)`. Therefore, followed by the logout and then repeatedly lo-

[2] https://en.wikipedia.org/wiki/List_of_the_most_common_passwords

[3] https://westoahu.hawaii.edu/cyber/forensics-weekly-executive-summmaries/8424-2/

gin using the open port: *ftp 192.168.122.36 51503*. Such a login to the **server5** via *FTP* protocol on the specific port **51503** meant obtaining this secret, which was printed out to the console.

3.5 Secret B

At the **server1** runs the service `http` at open port **80**. Therefore, the further steps led to the investigation of this service on this server. First, the available console browsers were researched at the server **bis.fit.vutbr.cz**, and the analysis showed, that there is no such service to use from the following list: `lynx`, `w3m`, `links`, `elinks`, `links2`, `netrik`. The next feature to obtain some information from the running `http` server is the using of tool `curl`. The first basic command `curl https://192.168.122.234/` printed out the following information:

```
<h1> Check host IP </h1>

<h2>
  Simple web app,
  executing host utility
</h2>
```

The function of the running application on this port and server has thus been sufficiently described. The obtained response to the queried request further included the information about the specific `form`:

```
<form action="" method="post">
<label> Host: </label>
  <input>
    type="text" id="url" name="url"
  </input>
  <input type="submit"/>
</form>
```

Based on these provided information, the command `curl -d "url=xstupi00" http://192.168.122.234/` was run, the response of which looked like the first response plus the following content:

```
Array
(
  [0] => xstupi00 has address
    192.168.122.165
)
```

The form of this array revealed that the queried `http` application running at the given server is implemented in the language `PHP`. This fact was verified by a supplementary query to the given `http` server, which was supplemented by the extension `index.php`. The response to this request was identical as in the first case at querying to this server. Another goal was to get some new response from the server through the available form. The interesting response was received after the query the `POST` request with the following data content `-d "url=form"`: *Don't try to break it to others!!*. By the inspiration of the success in the previous secret, the search for possible vulnerabilities followed again. At this combination of features and options was found out the possibility of the running the `bash` command through the data part of form. Therefore, the command to obtain the content of the remote server was run: `curl -d "url=;ls" http://192.168.122.234/`. The received response from the `http` server includes again the information about the running application and the following `array` part:

```
Array
(
  [0] => index.php
  [1] => secret.txt
)
```

With using the command `curl http://192.168.122.234/secret.txt` was obtained this secret.

3.6 Secret H

The same as at the **server1**, the `http` service runs also at the **server4**, also bind to the open port **80**. For initial acquaintance with the running web application the command `curl http://192.168.122.211/` was runned and its output is as follows:

```
<h1> Check user information </h1>

<h2>
  Simple web app,
  showing user information
</h2>
```

The purpose of this web application is provides the information about individual users. However, the `username` and `password` are required to login into this application, both within form items in `POST` request. Several combinations of `username` and `password` were tried in the next steps, in an effort to obtain some useful information in the form of response from the application. For example, the following pairs were tried: *root, root*; *admin, admin*; *user, user*; *root, password*; *user, 123456*; *test, test*, *server, server*; *joe, password1*, *username, password*, etc. After several unsuccessful attempts, the several extensions as the suffixes of the server address were tried, whether they provide any useful data or not: `/secret.txt`, `/www`, `/app`, `/var/www/html`, `/etc/shadow`, `/bin/get_-`

secret, etc. Another goal was to rediscover some vulnerabilities in the properties of this application.

At this **server4** runs also the service mysql at open port **3306**, what gives the presumption that information about the individual users is stored in the database. The attack, which uses the vulnerabilities of web applications, which requires the username and password within the text forms, is called SQL injection[4]. It is the placement of malicious code in SQL statements, via web application input. Therefore, the hacker might get access to usernames and passwords in a database by simply inserting " OR ""=" into the username and/or password text box of http POST request. With running the curl POST command to the address of the **server4** and addition of the data part name=" OR ""="&password=" OR ""=" was obtained this secret. It was stored in the database of users information, and the secret itself was stored in the item with name: password.

3.7 Secret C

When researching the **server2** was revealed that the directory home includes two another sub-directories. With except of home directory server2, the directory with name joe was there presented also. Since, the Secret **D** was found in the directory server2, there was the assumption, that the directory joe might also contains some secret. The command cd /home/joe/ was unsuccessful, because of the owner of this directory is user with name joe, and the login to the server was realised as user server2. Therefore, follows a switch of user with the following command: su - joe. The potential directory /home/joe was available this time, but it was not included some interesting content. In the next step, the goal was to look at other directories or folders, which are also owned by this user joe and it is not just his home directory. For this purpose, the following command was used, with the redirection of the stderr to /dev/null to avoid the error messages about the missing permission to access: find / -group joe 2>/dev/null. The output of this command contains **778** lines, but most of them were from directory /proc, thus they have not any relevant value. Of course, the output also contained all the files from the user's home directory (/home/joe), which were already examined at the beginning of this process. The only interesting file in this listing that might contain some useful information was a file located in the following destination: /var/spool/mail/joe. We used the cat command to list the contents of this file, and was showed, that it includes a large number of the email messages. The contents of this file were therefore filtered using the grep Tajemstvi command, and the relevant secret was printed out to the output.

3.8 Secret F

Thanks to the obtained password, in Section 3.3 (Secret **E**), for access to the **server3** as user joe, the researching of this server following. The secret **E** was found in the home directory (/home/joe) of this server, stored in the file secret.txt. In this home directory, was also the file junk.gdbm, which represents the possible presence of GNU database manager (GDBM). With using the gdbmtool was open this file, but it did not include any relevant information, which might be useful for the process of finding another secret at this server.

In the next step, as is traditional in the unknown environment, the root directory was examined using the command: ls -al /. There are present all expected directories, such as: /var, /etc, /home, /bin, etc., however, also the directory database_backup, which is definitely not common. After the opening of this directory was found that it contains only one file with the name 2020_dump. Subsequently, the command cat 2020_dump was used to print out its content. It was of type GDBM dump file, which was dumped from the file with name secret_db.gdbm with version 1.0 of GDBM tools. Since this dumped file showed, that origin database contained any data, the command gdbm_load was used to recreate the file secret_db.gdbm.

The file secret_db.gdbm was stored to the home directory of the user, where the interactive gdbmtool was run. After the listed out the available commands in this tool (command ?), was used the command open secret_db.gdbm to load the required database file. Another chosen command was the list, that already revealed the presence of the secret in this database. This secret could be also obtained with using the command fetch Secret within gdbmtool, or using the command cat secret_db.gdbm.

3.9 Secret J

At the **server5** runs the service rpcbind at the open port **111**. When searching the info about this service, respectively about its vulnerabilities, the following finding has been made. It is possible to check the configuration at another server by running rpcinfo -p from an unauthorised systems[5]. Therefore, with the

[4] https://www.w3schools.com/sql/sql_injection.asp

[5] https://ddos-guard.net/en/terminology/attacks/

using of this service has been found, that at the **server5** are bound two specific services and that portmapper and ypbind. This information was obtained in more detail by the following command: rpcinfo -p 192.168.122.36. When logging at to the **server5**, the username and password are required, so it was necessary to obtain a specific login detail in another way. In this computer network is one another computer where the service rcpbind is running, at that is the server1. The rpcinfo command was also run for this server, which detected that services portmapper and ypserv are binding on the relevant port. The service ypserv runs only on NIS server machines with a complete NIS database. Therefore, as the next step, the command find / -name "yp*" 2>/dev/null was used to find the potential useful locations of NIS server data.

The results included several interesting files, which can have valuable importance in the process of finding the secret. The most interesting locality appeared to be a directory /var/yp that contains also sub-directory bis and file with name ypservers. After looking at the contents of this folder, it turned out that it also contains a file called Makefile. It was the Makefile for the NIS databases, that should only be run on the NIS masters servers of a domain. Makefile further includes definition of files from which the NIS databases are build. Importantly, there are two items that read content from different locations than most others. In more specific, it is the variables PASSWD and SHADOW, whose inputs are located in the home directory /home/server1, in the files with name paswwd, respectively shadow.

When the look has been taken to second of these files, it was possible to see, that at the user with name bis_user is directly free placed the *encrypted password*. Since the Makefile reads from this file at building the database, the change of this *encrypted* password will affect the created maps. The new *encrypted* password was created with the following command: mkpasswd -method=SHA-512 -stdin, and subsequently replaced the origin *encrypted* password in the shadow file. At searching the files containing the prefix yp, was also detected the presence of command ypinit in the directory /usr/lib64. It builds the domain sub-directory of /var/yp/ for the current default domain, and subsequently builds a complete set of administrative maps for your system and places them in this directory. An *NIS* database on a slave server is set up by copying an existing database from a running server. It means, that after the updating and generating maps at the **server1**, the slave **server5** copy this updated database. To ensure this, the command ypinit -m was run, since the current local host is the NIS master server.

Finally, with using the command ssh bis_server@192.168.122.36 and the created password was realised the successful login to the **server5**. In this server, the home directory includes the hidden directory secret, which contains only one file with the name secret.txt. The content of this file is made up of only this secret. After returning to **server1**, the backup of the shadow file was performed and it was returned to its original state.

3.10 Secret G

Table 7 includes the host s4, which contains the hostname as address of this **server4** and the user with the name server. When attempting to login at this server with this credentials, the target server required entering the password. When researching the home directory at the **server2** was observed, that the directory .ssh contains the pair of keys within the following files: id_rsa and id_rsa.pub. Therefore, the successful logging to the **server4** was performed with the following command: ssh -i .ssh/id_rsa server@192.168.122.211. At this server, in the first step was researched the home directory, where the expected directories were present, but there are also is interesting directory, and its name is libgd. After the opening of the content of this directory, was showed, that it contains the full content of GD Graphics Library. Due to the presence of the files .git and .travis.yml was detected that it is also the git repository. Firstly, was opened the file README.md, which did not contain any information about the secret. Further, the directories such as: docs, examples, config, were also researched in more detail without the success. The command find was used, unsuccessful, to find out some directories or files, which contains in the name some from the following sub-strings: secret, tajemstvi, etc. In the next step, the properties of git repository was analysed. The current status of the repository shows, that it contains one unpublished local commit. With using the command git log -oneline was found that the message of this commit was as follows: Super secret commit message. After this, was tried the command git log to obtain the sub-messages of the commits too, but this relevant commit did not contain any information. By usage, the command git show <commit-id> was showed the log message and textual diff, which contains this secret.

server-port-111-rpcbind-vulnerability