WEAK RSA

[Publish Date]

European Cyber Security Challenge 2018
London, United Kingdom

1. Initial Write-Up

Description:

During the OSINT phase of a regular vulnerability assessment evidences of a data breach have been found on pastebin. Further investigation showed that one of the frontend webserver of Know Your Brand PLC had been compromised. The investigation also revealed that the attacker managed to capture network traffic. Analysis of the capture files the attacker left on the compromised machine showed that the capture files should be the source of the recent data breach. Tracing back the attacker's IP address has been failed. However since the captured communication is encrypted and the private keys of the server are only accessible by the system administrators, the IT operation was incriminated.

Believing in the innocence of his collages, the head of IT operation hire your company to further investigate this case and prove their innocence.

Your company is provided with the network capture file the attacker left on the compromised server. The IP address of the compromised server is 172.30.11.10 and the IP address of the backend server is 172.30.10.10. The URL of the source of the sensitive information is https://172.30.10.10/FLAG.

Note: the provided nextstep.zip file is encrypted by the flag you have to recover by solving the first problem. The zip container contains the last step of this challenge and the questions you have to answer.

2. Challenge specifications

- Category: Web

3. Questions and answers

Q1: What is the used cipher suite?
A1: TLS_RSA_WITH_AES_256_GCM_SHA384

Q2: What is the FLAG?

A2: :#m2h$U"!{+G5,?hUH|JTi5;d\N]h4V_

Q3: What is the impact of the attack?
A3: The SSL communications are eavesdropped.

Q4: Who was the original developer of the generator script (genCsr.py)?
A4: Lars Kruse

Q5: What is the name of the developer who made the malicious change?
A5: Jon L. Boyer

Q6: Since when were all generated keys vulnerable?
A6: 2018-07-10 13:01:00

4. Tools needed

Description:

Tools needed for the solution of the challenge:

- A network traffic analyser (eg. Wireshark)
- An arbitrary tool to compute the private key from the public key. (eg. RsaCtfTool)

5. Artifacts hashes

File	MD5	SHA256
cap01.pcap	b152fd9873deeeb62c2ab079922cccd6	b2d08a7a174782faad60723b918e6ac646c828566e96dfe69bf224b03be86463
nextstep.zip	2b1d7313b1aae4cf0fe5bc16a1b7977a	fd0bf181cc5d5608c2fbb81af90fd29cf974ff3dbaae0a765f27665e17c6da0b

6. Walkthrough (writeup)

Questions serve as a guide to solve the main question outlined in the write-up: 'what could possibly happen here'. As a convention 10.0.0.0/8 belongs to external/public networks, while 172.X.0.0/16 and 192.168.X.0/24 are private LANs.

1. Analyze the provided network capture file and find the key exchange algorithm used in the https communicaton

   In Wireshark set a filter using the ip address of the backend server: ip.addr==172.30.10.10

   Find the "Server Hello" message of a handshake and examine the content of the message.

   Notice that the cipher suite used is TLS_RSA_WITH_AES_256_GCM_SHA384 which means the initial key exchange algorithm is RSA.

Q1: What is the used cipher suite?

A1: TLS_RSA_WITH_AES_256_GCM_SHA384

2. Extract the server certificate from the provided network capture file

   Extract the server certificate.[1]

3. Extract the publikey from the certificate

   Using the openssl tool extract the publikey from the server certificate:

```
$openssl x509 -inform der -in <filename of the cert extracted from pcap> -
pubkey -noout > cert_pubkey.pem
```

4. Test the public key for known weaknesses and find the private key

   There are many tools available to test and "crack" weak RSA public keys. The weakness implemented in the challenge is one of the most basic one: it chooses a 'q' (given the primes used to create the RSA private key are denoted with 'p' and 'q' as usually) which is very close to 'p'. Therefore, all the attacker has to do is to search the environment of the square root of n (where n = pq) to find the factors of 'n' so the private exponent 'd' can be calculated from the public exponent "e" and from phi(n)=(p-1)(q-1) solving the following problem:

$$ed \equiv 1 \ (mod\ \varphi(n))$$

[1] https://security.stackexchange.com/questions/123851/how-can-i-extract-the-certificate-from-this-pcap-file

using the extended Euclidean algorithm.

This is the theory behind the challenge but the contestant should preferably use an existing tool like the "RsaCtfTool" to solve this part of the challenge.

Download the tool and use it to extract the private key:

```
$git clone https://github.com/Ganapati/RsaCtfTool.git
$cd RsaCtfTool
$python RsaCtfTool.py --publickey ../cert_pubkey.pem --private
>../recovered.key
```

5. Using the computed private key decrypt the ssl communication

   In Wireshark add the recovered private key to RSA keylist:

   - In Preferences- >Protocols->SSL – next to "RSA keys list" click on "Edit.. "
   - add the recovered key to list using the following data set: ip = 172.30.10.10, port = 443, protocol = "http", Key File - <select the recovered key file>

6. Find the Https request-response pair containing the FLAG information to prove the communication can be decryptable without the private key

   Find the response for the /FLAG http GET query and read the key from the json:

- In Wireshark set a filter using the previously defines FLAG URL (https://172.30.10.10/FLAG): http contains "/FLAG"
- Find the number of the searched packet. And clear the filter.
- CTRL+G for jumping the /FLAG's packet and find below it's http response (3806 149.892536122 172.30.10.10 172.30.11.10 HTTP 435 HTTP/1.1 200 OK (application/json))
- Select the row and Find below (in the tree view) the JavaScript Object Notation […] item, expand it, find the flag key and its value. Copy it. :#m2h$U"!{+G5,?hUH|JTi5;d\N]h4V_

Q2: What is the FLAG?

A2:

```
:#m2h$U"!{+G5,?hUH|JTi5;d\N]h4V_
```

7. Using the flag value unpack the next task

   The recovered flag is the password for the zip file provided.

```
$unzip nextstep.zip
```

   Read the README file.

8. Clone the git repository and find the python script which generates the weak certificate

```
$git clone kyb-devops/ workdir/
```

9. Using git blame identify the developer who made the malicious modification in the source code

```
$cd workdir

$git blame genCsr.py
```

   Read the output and find the name of the malicious actor: “Jon L. Boyer”.

Q3: What is the impact of the attack?

A3: The SSL communications are eavesdropped.

Q4: Who was the original developer of the generator script?

A4: Lars Kruse

Q5: What is the name of the developer who made the malicious change?

A5: Jon L. Boyer

Q6: Since when were all generated keys vulnerable?

A6: 2018-07-10 13:01:00