Lecture Network Security
Assignment Sheet 4

Jens Tölle, Wolfgang Moll
Jonathan Chapman, Martin Clauß, Martin Lambertz, Christian Meier

Summer Term 2016

Publication date of this sheet:	26.05.2016
Submission deadline (via email):	07.06.2016, 23:59:59
Discussion of results:	09.06.2016

Results must be submitted via email to
network-security-worksheet@lists.iai.uni-bonn.de
in one archive file named after the scheme
`sheet4_lastname1_lastname2[_lastname3].{tar|tar.gz|tgz|zip}`

Task 4.1 (practical): DNS spoofing

On the last assignment sheet, you prepared a DNS sniffer [1]. On this sheet it's your task to expand your program and write a component which generates fake DNS responses to sniffed requests.

The domain name you have to generate faked responses for is **fakeme.seclab.bonn.edu**. The fake IP address for your response depends on your SecLab username. It has to be generated by calling the command `name2ip` on hellgate. Note that you have to call this program as a normal user and not as root or with sudo.

You have to run your program on the host hellgate in the SecLab. Please verify your solution by using the service running on **white** on port 5353/tcp. You can contact the service with telnet or netcat. When connected, the service performs a DNS lookup for fakeme.seclab.bonn.edu. This is the query you have to send spoofed responses for. If your spoofing attempt was successful, it shows a personalized secret for your username.

telnet white 5353

Submit:

- The source code of your program.
- The response of the verification service on port 5353/tcp (including the secret) for every group member.

Hints:

- Multiple active spoofers may interfere with each other. Therefore, please turn off your spoofer after having tested it.
- One possibility for implementing the spoofer is the use of "raw sockets". If you use raw sockets, make sure that you put the network interface into promiscuous mode (requires root privileges), e.g. like this `ifconfig eth0 promisc`.

[1] In case you didn't, you can use the one on the lecture homepage. See "supplementary material" next to the assignment sheet download link

- We have slowed down the DNS server to delay all responses by 200 ms. This should give you plenty of time to deliver your response before the original server's arrives. However, some of the services inside the SecLab rely on DNS and may be slowed down, too. Please be patient and use IP addresses where possible.
- You can use `nc <ip> <port>` to read from and `echo 'Something' | nc <ip> <port>` to write to an open network port.

Task 4.2 (theoretical): Message Authentication Code (MAC)

Show that the following method to calculate MACs is not reasonable. The method takes a message m of l bytes and divides this text into blocks of length 8 (with zero padding if needed). Afterwards, it calculates a hash value K and encrypts it with a key k only known to the communication partners.

- Input: message $m = (x_1, x_2, \ldots, x_n)$, with x_i being 64-bit values
- Calculate $K(m) := x_1 \oplus x_2 \oplus \ldots \oplus x_n$, where $\oplus$ is an exclusive or (XOR)
- Output: $H_k(m) := E_k(K(m))$, where $E_k(.)$ is some encryption function with parameter k

Please show that an attacker, eavesdropping a message $M = (m, H_k(m))$, may, without knowledge of key k, create a message $M' = (m', H_k(m))$ with almost arbitrary message text m', which leads to the same MAC like the original message. In this case, $H_k(m) = H_k(m')$ holds.

Task 4.3 (practical): Diffie-Hellman

On host `blue` in our SecLab is a service which uses a well-known key exchange protocol to agree on a common shared secret. Try to find a vulnerability in the key exchange protocol. Exploit this vulnerability to be able to read the encrypted communication. Have a look at ports 3333/tcp and 4444/tcp at blue (e.g. by using `nc`).

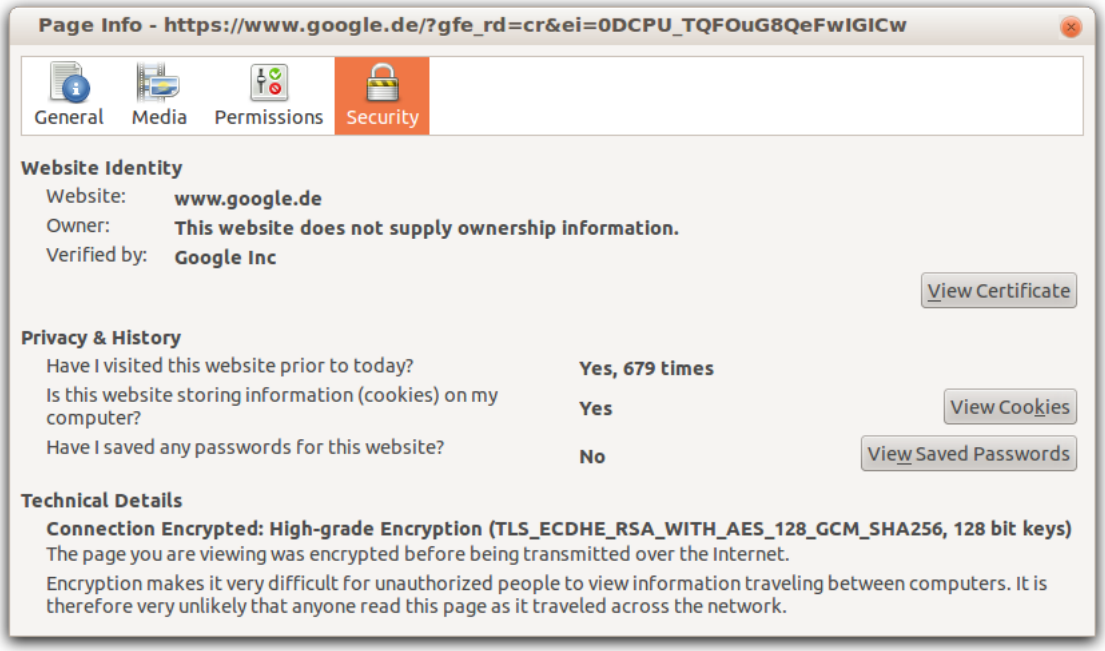
Submit:

- A short description of the vulnerability you exploited
- The source code of your program
- The top secret passphrase transfered during the encrypted communication

Task 4.4 (theoretical): TLS Cipher Suites

TLS offers a variety of different combinations of methods for authentication, encryption and integrity. These combinations are called cipher suites. Some of them are currently believed to be secure, while others are already known to be broken. When opening a

TLS secured website (via https://...), few users actually check which suite is used, so let us take a closer look!



Above you can see a screenshot of the Firefox security information for the website www.google.de. Compare the above cipher suite `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256` to the cipher suite `TLS_RSA_RC4_128_MD5`:

- For both suites, (very) briefly describe the methods used for authentication, encryption, and integrity, respectively.
- Give three reasons why the first suite is (most probably) more secure than the second. Justify each reason by briefly describing the known/possible attacks on the weaker method. Don't forget to mention your sources! Hints: *Perfect Forward Secrecy* and the NSA ☺

Task 4.5 (practical): Cipher Suites in the Wild

Write a program/script that is able to list all cipher suites supported by a given webserver. Run your program with a list of at least 100 popular websites (you can, for example, use the Alexa Top 500 list). Create a histogram of the supported suites.