Homework 3 - MITM Implant

Introduction

You have gained access to a peripheral machine on a target network, but need to compromise other target machines. You know that users on those machines frequently download files from the Internet; in this homework, you will exploit this attack vector. Your task is to write an implant to MITM a specified target, intercept download requests from the target machine to the gateway, and inject a malicious binary in its place. This malicious binary will be given to you (one version for Windows, one version for Linux). In this manner, once your implant is running, if the target machine tries to blindly download & run an .exe file, it will get and run the malicious file instead. Besides this file injection, no other network operations on the target machine should be affected.

Requirements

First, the implant must achieve a man in the middle between a given target machine the gateway; doing so in IPv4 with an ARP spoof attack is sufficient. Your implant should pass along all unrelated network traffic so that the users are unaware your implant is running or that they have been attacked, while searching for HTTP GET requests. If an HTTP GET request to a URL ending in .exe or .sh is made, your implant should spoof a response request and send the given binaries instead. You will have to manipulate and/or build TCP packets in order to accomplish this. You may assume the .exe/.sh file will fit within one TCP packet.
Other requirements:

- Your implant should be written in Python. You must provide a singular self-contained executable named `implant.py` that can be invoked directly.
- Unlike the previous implant, you are allowed to use outside libraries. Scapy and NetfilterQueue are *highly* recommended.
- Implant must re-ARP the targets upon exit.

Usage

It should take 2 command line arguments, the target machine and the gateway (these are the machines to MITM). Two files, `bad.exe` and `bad.sh`, will be in the same directory as your implant to use for download replacement (requests to `*.exe` should get `bad.exe`, requests to `*.sh` should get `bad.sh`). When a SIGINT (KeyboardInterrupt/Control-C/^C) is sent, the program should gracefully reARP the targets so the users can access the Internet after the MITM has shut down.

```
# ./implant <target ip address> <gateway ip address>
```

Example attacking Windows host:

```
# ./implant 10.10.10.13 10.10.10.1
```

```
ARPing targets...
Successfully ARP spoofed
Identified .exe download request from 10.10.10.13, replacing binary
Identified .exe download request from 10.10.10.13, replacing binary
^C
reARPing targets…
Successfully reARPed targets.
#
```

Example attacking Linux host:

```
# ./implant 10.10.10.11 10.10.10.1
ARPing targets...
Successfully ARP spoofed
Identified .sh download request from 10.10.10.11, replacing script
^C
reARPing targets…
Successfully reARPed targets.
#
```

Extra Credit

Extra Credit 1 - 3 points
Extra credit is available on this assignment if you add a third, optional command line argument for `.sh` file injection. In this mode, your implant will inject the given bad.sh script into the requested .sh file (instead of replacing it entirely) so that the original functionality of the downloaded file remains unaffected. If you attempt the extra credit, create a brief `.pdf` report explaining how you accomplished the task and screenshots/proof of it working.

Extra Credit 2 - 10 points
Additional extra credit is available on this assignment if you can demonstrate in a `.pdf` report (with screenshots) or demo to us that your implant can function like evilgrade: injecting an .exe to attack a regular normal program in the wild (such as CCleaner, Notepad++, etc) that is attempting to update itself. For suggestions of vulnerable applications to try, see the modules supported by evilgrade.

Testing

You are encouraged to use the environment on aceslab used in class for MITM, though this is not required. This environment is functionally identical to what we will use when grading your implant (a simple environment that has Linux and Windows Targets, Kali Attacker, and a server. The target machines will download `.sh/.exe` files from the server and run the file it downloads. You can run a server that serves up files on the server machine with:

```
$ sudo python -m SimpleHTTPServer 80
```

Submission

Please submit your singular, self-contained implant in a .zip file to the ELMS submit link. If you attempt the extra credit, say so in the submission comments, and attach a `.pdf` report of what you did and how you did it.