Anthony Dust / Adam Wiggenhauser
aldust2 / wiggenh2
CS 460
Final Project
Spring 2016

The Avenger: Network low-hanging fruit grabber

Goals:

We wanted to build a tool that could scan a network for low hanging fruit, exploit that fruit for a foothold and perform common post-exploitation tasks. We set out to build a Metasploit resource file that could automatically dictionary attack common network service logins and perform post-exploitation with the access that we would be granted. We also wanted to perform phishing with client side exploits as that is also considered low hanging fruit.

Challenges:

Future work:

Unfortunately, a lot of system administration time is needed to properly configure all of the services we wanted to set up and test. We did not make it to the phishing nor email server portion of our tool, but we do wish to set up a XEAMS mail server VM in the future when we will further improve our tool. We also wish to automate theHarvester and dump it's output into the metasploit db in the future as well as automating MSFVENOM to generate connect back executable files to our listeners which we can send in phishing emails. We had planned to send phishing emails as post-exploitation upon taking over a mail server, but that will be future work. We had also anticipated possibly importing openvas scans, but the current Metasploit Openvas plugin has known bugs so we chose to use db_nmap as an alternative proof-of-concept of the fact that we could dump scan info to the database and act upon it. Currently, regardless of what ports are open we try the scans, but in the future we could check the nmap output to save us time and only scan what services are there. SMB, SMTP, POP3, and VNC are services we wish to add in the future. Additionally, we would like to improve the sessions management such that once we get a root shell on a given box, we stop scanning for additional means of access. Exception handling for when something goes wrong is also another future to-do. We chose Linux because not much has been done for Linux in Metasploit and Windows will be easier to bring into the tool than Linux would have been had the tool been Windows-centric in the first place. Additionally, many aspects of the tool at it's current proof-of-concept stage are hard-coded but could be improved as user controlled variables in the future.

Finished work:

We completed a resource file, and two modules. The resource file runs nmap against a target host and kicks off SSH, FTP, and Telnet scans. If the SSH and Telnet scans are successful we receive shells back and upgrade them to meterpreters. For SSH post-exploitation we developed a custom post-exploitation module for Linux called upload_single which takes two parameters: a local file and a remote path. A module to upload files was not previously available for Linux and Linux file uploading was only possible through manual interaction with the meterpreter. No more! Also there were tutorials where scripts were used, but scripts (different from resource files) are now deprecated and some of the old global variables that Metasploit used to support for scripts are no longer present. We developed this utilizing the Metasploit API as stated in the rubydoc for Metasploit. We decided to change the wallpaper of an exploited host for SSH post-exploitation. To change the wallpaper on our Ubuntu machine we chose to overwrite the current wallpaper file with our uploaded file because all of the methods for querying and setting the wallpaper from the terminal that we discovered online were not functioning in the latest Ubuntu. For FTP post-exploitation, we modified the FTP auxiliary scanner to send extra commands upon successful login to initiate a file download, the contents of which we print

to the terminal. This involved us again consulting the Metasploit rubydoc, learning FTP commands, and modifying an existing metasploit library called client.rb under the FTP directory in Lib (/usr/share/metasploit-framework). We modified client.rb such that upon a successful FTP data connection we return a handle to that socket to the caller. The caller is our resource script which then queries the handle for any data that the socket has received. We then print that data to the terminal. We found the needed methods in the rubydoc. Telnet is much like SSH so we did not feel the need to demonstrate any post-exploitation maneuvers as long as we performed SSH post-exploitation.

Results:

Although we did not complete everything we set out to do, we still learned a lot of helpful things about developing for Metasploit and built a pretty nifty tool that is useful and will serve as a great foundation for the tool that we originally envisioned. We learned how resource files work and how to print out information to the terminal, use variables, query and store to the Metasploit database, utilize the Metasploit API, automate msfconsole commands, and use common ruby programming constructs. We also learned how to build Metasploit modules and how to read the rubydoc to extend our modules beyond what is currently available. We produced a resource file that can automate scanning a network for low hanging SSH, FTP, and Telnet fruit (creds logged to db), gain meterpreter shells on machines with weak passwords, change the background given an Ubuntu machine, and upload and download files to and from the exploited machine. Although the tool is at a proof-of-concept stage currently, it shows promise and could be trivially modified to be a one command network pwn tool for course staff during future attack/defend labs.

Demo notes:

fruit.rc => /usr/share/metasploit-framework/scripts/resource/fruit.rc
upload_single.rb => /usr/share/metasploit-framework/modules/post/linux/manage/upload_single.rb
ftp_download.rb => /usr/share/metasploit-framework/modules/auxiliary/scanner/ftp_download.rb
client.rb => /usr/share/metasploit-framework/lib/metasploit/framework/ftp/client.rb

Additionally, we used Ubuntu 15.10, with openssh-server, vsftpd and telnet via xinetd. We configured each of these services to enable root login and added extra key exchange and encryption algorithms to openssh-server so that it could communicate with Metasploit's supported algorithms which are slightly dated. If you wish to run the demo yourself, you will need to change the hardcoded IP address in the fruit.rc resource file and copy all of the project files to the aforenoted locations. You will also have to make sure that there is a picture in /tmp/uni.jpg on the attacking machine to upload (or you can change the path in fruit.rc) and a file in /tmp/hello.txt on the target machine (or you can change the path in fruit.rc). Also be sure to add your own username and password lists (the default location is /usr/wordlists/user.txt and /usr/wordlists/pass.txt, but again this can be changed in fruit.rc). Please see our demo video for a complete demonstration of our tool.