|  |  |
| --- | --- |
| Cover Image | *Violent Python, A Cookbook for Hackers, Forensic Analysts, Penetration Testers and Security Engineers* TJ O'Connor  Newnes  [Get it on Google Play](https://play.google.com/store/books/details?id=2XliiK7FKoEC&source=books-notes-export) |

|  |
| --- |
| This document is overwritten when you make changes in Play Books.  You should make a copy of this document before you edit it. |

# *988 notes/highlights • 12 bookmarks*

*Created by Tom Goodheart*  – Last synced August 6, 2018

## *Introduction*

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Highlighted text cannot be displayed*  [Why can't I see my highlighted text?](https://support.google.com/googleplay/?p=books_notes_drive)  August 3, 2018 | [24](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PR23.w.0.0.0.0.1) | |

## *Chapter 1. Introduction*

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *A small Python script granted him access to over one thousand workstations. Another small script allowed him to triage*  July 7, 2018 | [29](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA2.w.4.0.63) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Highlighted text cannot be displayed*  August 3, 2018 | [29](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA2.w.8.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Highlighted text cannot be displayed*  August 3, 2018 | [29](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA2.w.8.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Highlighted text cannot be displayed*  August 3, 2018 | [29](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA2.w.8.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Highlighted text cannot be displayed*  August 3, 2018 | [29](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA2.w.8.0.62) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Python is an interpreted language. At runtime an interpreter processes the code and executes it*  July 7, 2018 | [32](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA5.w.10.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Python provides interactive capability. A programmer can invoke the Python interpreter and interact with the interpreter directly*  July 7, 2018 | [33](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA6) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we will build our scripts out of several functional blocks of code known as methods or functions. As we finalize each script, we will show how to reassemble these methods and invoke them from the main() method*  July 7, 2018 | [33](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA6.w.5.0.11) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *variable points to data stored in a memory location*  July 7, 2018 | [34](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA7) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *can store different values such as integers, real numbers, Booleans, strings, or more complex data such as lists or dictionaries*  July 7, 2018 | [34](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA7) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *To combine the two variables together into one string, we must explicitly cast the port as a string*  July 7, 2018 | [34](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA7) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Python reserves memory space for variables when the programmer declares them*  July 7, 2018 | [34](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA7.w.3.0.65) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the Python interpreter decides the type of the variable and how much space in the memory to reserve*  July 7, 2018 | [34](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA7.w.3.0.65) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Python string module provides a very robust series of methods for string*  July 7, 2018 | [34](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA7.w.6.4.1) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Replace(old,new) replaces the old occurrence of the substring old with the substring new. Find() reports the offset where the first occurrence of the substring occurs.*  July 7, 2018 | [35](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA8) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *list data structure in Python provides an excellent method for storing arrays of objects*  July 7, 2018 | [35](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA8.w.2.1.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Python dictionary data structure provides a hash table that can store any number of Python objects. The dictionary consists of pairs of items that contain a key and value.*  July 7, 2018 | [35](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA8.w.5.4.26.0.16) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *When scanning specific TCP ports, it may prove useful to have a dictionary that contains the common service names for each port.*  July 7, 2018 | [36](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA9) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *socket module provides a library for making network connections using Python*  July 8, 2018 | [36](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA9.w.5.2.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Our script will print the banner after connecting to a specific IP address and TCP port*  July 8, 2018 | [36](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA9.w.5.2.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *importing the socket module, we instantiate a new variable s from the class socket*  July 8, 2018 | [36](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA9.w.7.0.18) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we use the connect() method to make a network connection to the IP address and port*  July 8, 2018 | [36](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA9.w.7.0.18) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *successfully connected, we can read and write from the socket*  July 8, 2018 | [36](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA9.w.7.0.18) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *recv(1024) method will read the next 1024 bytes on the socket*  July 8, 2018 | [36](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA9.w.7.0.18.0.1) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Python provides a method for conditional select statements*  July 8, 2018 | [37](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA10) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *IF statement evaluates a logical expression in order to make a decision based on the result of the evaluation*  July 8, 2018 | [37](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA10.w.3.0.10) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *when a programmer writes a syntactically correct program, the program may still error at runtime or execution*  July 8, 2018 | [37](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA10.w.3.0.127) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *We use try/except statements to provide exception handling*  July 8, 2018 | [38](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA11) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *It might be useful to provide the user with an error message about the specific error that occurred. To do this, we will store the exception in a variable e to print the exception*  July 8, 2018 | [38](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA11.w.5.0.5) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *functions provide organized blocks of reusable code. Typically, this allows a programmer to write a block of code to perform a single, related action.*  July 8, 2018 | [39](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA12) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *The keyword def() begins a function. The programmer can place any variables inside the parenthesis. These variables are then passed by reference, meaning that any changes to these variables inside the function will affect their value from the calling function.*  July 8, 2018 | [39](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA12.w.8.0.8) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *our script needs to check this banner against some known vulnerable program*  July 8, 2018 | [39](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA12.w.9.2.10) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *reflects a single, related function. The function checkVulns() takes the variable banner as a parameter*  July 8, 2018 | [40](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA13) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *you might have found it repetitive to write almost the same exact code three times to check the three different IP addresses*  July 8, 2018 | [41](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA14.w.0.0.0.0.18) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we might find it easier to use a for-loop to iterate through*  July 8, 2018 | [41](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA14.w.6.0.25) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *for example: if we wanted to iterate through the entire /24 subnet of IP addresses for 192.168.95.1 through 192.168.95.254*  July 8, 2018 | [41](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA14.w.6.0.25) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *a for-loop with the range from 1 to 255 allows us to print out the entire subnet*  July 8, 2018 | [41](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA14.w.6.0.25) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we may want to iterate through a known list of ports to check for vulnerabilities*  July 8, 2018 | [41](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA14.w.8.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Nesting our two for-loops, we can now print out each IP address and the ports for each address*  July 8, 2018 | [42](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA15.w.0.0.0.0.16) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the ability to iterate through IP addresses and ports, we will update our vulnerability-checking script*  July 8, 2018 | [42](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA15.w.7.0.9) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *it would be nice to occasionally add a new list of vulnerable banners*  July 8, 2018 | [43](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA16.w.1.0.80.0.22) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *let’s assume we have a text file called vuln\_banners.txt*  July 8, 2018 | [43](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA16.w.4.0.19) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *this file lists a specific service version with a previous vulnerability*  July 8, 2018 | [43](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA16.w.4.0.19) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *place our updated code in the checkVulns function*  July 8, 2018 | [44](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA17.w.0.0.0.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we will open the text file in read-only mode (‘r’). We*  July 8, 2018 | [44](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA17.w.0.0.0.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *iterate through each line in*  July 8, 2018 | [44](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA17.w.0.0.0.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *built-in sys module provides access to objects used or maintained by the Python interpreter*  July 8, 2018 | [44](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA17.w.4.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *includes flags, version, max sizes of integers, available modules, path hooks, location of standard error/in/out, and command line arguments called*  July 8, 2018 | [44](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA17.w.4.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Consider our vulnerability scanner: what if we wanted to pass the name of a text file as a command line argument? The list sys.argv contains all the command line argument*  July 8, 2018 | [44](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA17.w.7.0.30) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Take the time to examine the entire sys module for the wealth of capabilities it provides to the programmer*  July 9, 2018 | [45](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA18) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *built-in OS module provides a wealth of OS routines for Mac, NT, or Posix operating systems*  July 9, 2018 | [45](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA18.w.5.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *independently interact with the OS environment, file-system, user database, and permissions*  July 9, 2018 | [45](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA18.w.5.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the user passed the name of a text file as a command line argument. It might prove valuable to check to see if that file exists and the current user has read permissions to that file*  July 9, 2018 | [45](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA18.w.5.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *verify our code, we initially try to read a file that does not exist, which causes our script to print an error. Next, we create the specific filename and successfully read it. Finally, we restrict permission and see that our script correctly prints*  July 9, 2018 | [45](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA18.w.15.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Clifford Stoll, documented his personal hunt for a hacker (and KGB informant) who broke into various United States national research laboratories, army bases, defense contractors, and academic institutions in The Cuckoo’s Egg: Tracking a Spy Through the Maze of Computer Espionage*  July 9, 2018 | [47](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA20.w.5.0.6) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *also published a May 1988 article in Communications of the ACM describing*  July 9, 2018 | [47](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA20.w.5.0.6) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Stoll noticed something interesting (at least in 1988). Almost immediately after compromising a victim, the attacker downloaded the encrypted password file*  July 9, 2018 | [47](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA20.w.6.0.53) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *week of stealing the encrypted password files, Stoll saw the attacker log on with the stolen accounts. Confronting some of the victim users, he learned they had used common words*  July 9, 2018 | [48](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA21) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the following encrypted password file. The victim used a plaintext password egg and salt equal to the first two bytes or HX. The UNIX Crypt function calculates the encrypted password with crypt(‘egg’,’HX’) = HX9LLTdc/jiDE.*  July 9, 2018 | [48](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA21.w.3.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *real strength of the Python programming language lies in the wide array of standard and third-party libraries*  July 9, 2018 | [48](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA21.w.5.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *our UNIX password cracker, we will need to use the crypt() algorithm that hashes UNIX passwords*  July 9, 2018 | [48](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA21.w.5.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *To calculate an encrypted UNIX password hash, we simply call the function crypt.crypt() and pass it the password and salt as parameters*  July 9, 2018 | [48](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA21.w.7.0.39) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *try hashing a password using the crypt() function*  July 9, 2018 | [49](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA22.w.0.0.0.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *To write our program, we will create two functions-main and testpass. It proves a good programming practice*  July 9, 2018 | [49](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA22.w.3.0.46) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *allows us to reuse code and makes the program easier to read*  July 9, 2018 | [49](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA22.w.3.0.46) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *main function opens the encrypted password file “passwords.txt” and reads the contents of each line in the password file*  July 9, 2018 | [49](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA22.w.5.0.36) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *For each line*  July 9, 2018 | [49](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA22.w.5.0.36) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *splits out the username*  July 9, 2018 | [49](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA22.w.5.0.36) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *password.*  July 9, 2018 | [49](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA22.w.5.0.36) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *For each*  July 9, 2018 | [49](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA22.w.5.0.36) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the main function calls the testPass() function*  July 9, 2018 | [49](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA22.w.5.0.36) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *testPass(), takes the encrypted password as a parameter and returns either after finding the password or exhausting the words in the dictionary*  July 9, 2018 | [49](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA22.w.5.0.36) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the function first strips out the salt from the first two characters of the encrypted password hash*  July 9, 2018 | [49](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA22.w.6.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *If the result matches our encrypted password hash, the function prints a message indicating the found password and returns*  July 9, 2018 | [49](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA22.w.6.0.68) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *explained that there are really no offensive or defensive tools-instead there are simply tools*  July 9, 2018 | [50](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA23.w.7.0.42.0.8) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *The police department brought in city programmer Albert Castillo to search the contents of Zimmerman’s computer (McCullagh, 2008). Castillo’s initial investigation found several adult pornographic images but*  July 10, 2018 | [51](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA24.w.3.0.46) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *examining the zipfile library*  July 10, 2018 | [51](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA24.w.5.0.31) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *issue the command help(‘zipfile’)*  July 10, 2018 | [51](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA24.w.5.0.31) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *see the class ZipFile with a method extractall(). This class and method will prove useful in writing our program to crack password-protected zip files*  July 10, 2018 | [51](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA24.w.5.0.31) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *a quick script to test the use of the zipfile library. After importing the library, we instantiate a new ZipFile class by specifying the filename of the password-protected zip file. To extract the zip file, we utilize the extractall() method and specify the optional parameter for the password*  July 10, 2018 | [52](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA25.w.7.0.1) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *prior to execution, only the script and the zip file exist in our current working directory. We execute our script, which extracts the contents of evil.zip to a newly created directory called evil/. This directory contains the files from the previously password-protected zip file*  July 10, 2018 | [52](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA25.w.12.0.34) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *what happens if we execute the script with an incorrect password? Let’s add some exception handling to catch and display the error message*  July 10, 2018 | [52](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA25.w.16.1.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Executing our script*  July 10, 2018 | [52](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA25.w.16.1.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *indicating that the user specified an incorrect password to decrypt*  July 10, 2018 | [53](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA26) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *use the fact that an incorrect password throws an exception to test our zip file against a dictionary file*  July 10, 2018 | [53](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA26) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *instantiating a ZipFile class, we open a dictionary file and iterate through and test each word in the dictionary*  July 10, 2018 | [53](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA26) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *extractall() throws a bad password exception, we ignore the exception and continue trying passwords*  July 10, 2018 | [53](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA26.w.6.0.30) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Executing our script*  July 12, 2018 | [53](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA26.w.7.0.10) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *it correctly identifies the password for the password-protected zip file*  July 12, 2018 | [53](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA26.w.7.0.10) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Instead of having a linear program, we will modularize our script with functions*  July 13, 2018 | [54](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA27) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *modularized into separate functions, we can now increase our performance*  July 13, 2018 | [54](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA27.w.5.0.16) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *utilize threads of execution to allow simultaneous testing of multiple passwords*  July 13, 2018 | [54](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA27.w.5.0.16) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *each word in the dictionary, we will spawn a new thread*  July 13, 2018 | [54](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA27.w.9.0.38) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *let’s modify our script to allow the user to specify the name of the zip file to crack and the name of the dictionary*  July 13, 2018 | [55](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA28.w.0.0.0.0.10) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *import the optparse library*  July 13, 2018 | [55](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA28.w.0.0.0.0.10) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *it parses flags and optional parameters*  July 13, 2018 | [55](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA28.w.4.0.47) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | July 13, 2018 | [55](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA28.w.5.3.35.0.28) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *CyberFastTrack.m4v>*  July 13, 2018 | [56](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA29.w.11.2.6.0.2) | |

## *Chapter 2. Penetration Testing with Python*

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Highlighted text cannot be displayed*  August 4, 2018 | [58](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA31) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Twenty-two years before the StuxNet worm crippled the Iranian nuclear power plants in Bushehr and Natantz*  July 14, 2018 | [58](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA31.w.0.0.0.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *graduate student at Cornell launched the first digital munitions. Robert Tappen Morris Jr., son of the head of the NSA’s National Computer Security Center, infected six thousand workstations*  July 14, 2018 | [58](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA31.w.7.0.58) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *this figure represents ten percent of all computers that were connected to the Internet in 1988*  July 14, 2018 | [58](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA31.w.7.0.58) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *put the cost somewhere between $10 and $100 million*  July 14, 2018 | [58](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA31.w.9.1.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Morris’s worm used a three-pronged attack in order to compromise systems*  July 14, 2018 | [58](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA31.w.9.1.2.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *first took advantage of vulnerability in the Unix sendmail program*  July 14, 2018 | [58](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA31.w.9.1.2.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Second, it exploited a separate vulnerability in the finger daemon used by Unix systems*  July 14, 2018 | [58](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA31.w.9.1.2.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Finally, it attempted to connect to targets using the remote shell (RSH) protocol*  July 14, 2018 | [59](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA32) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *If any of the three attack vectors succeeded, the worm would use a small program as a grappling hook to pull over the rest of the virus*  July 14, 2018 | [59](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA32) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Morris wrote the majority of his attack in the C programming*  July 14, 2018 | [59](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA32.w.6.0.6) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Reconnaissance serves as the first step*  July 14, 2018 | [59](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA32.w.6.0.76.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *attacker must discover where the vulnerabilities*  July 14, 2018 | [59](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA32.w.6.0.76.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we will build a small reconnaissance script that scans a target host for open TCP*  July 14, 2018 | [59](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA32.w.8.0.17) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we will need to first construct TCP sockets*  July 14, 2018 | [59](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA32.w.8.0.17) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Python*  July 14, 2018 | [59](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA32.w.8.0.17) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *provides access to the BSD socket interface. BSD sockets provide an application-programming interface that allows coders to write applications in order to perform network communications between hosts*  July 14, 2018 | [59](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA32.w.8.0.17) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *a series of socket API functions, we can create, bind, listen, connect, or send traffic on TCP/IP sockets*  July 14, 2018 | [59](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA32.w.9.0.18) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *The majority of Internet accessible applications reside on the TCP*  July 14, 2018 | [59](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA32.w.9.0.18) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *in a target organization, the web server might reside on TCP port 80, the email server on TCP port 25, and the file transfer server on TCP port 21*  July 14, 2018 | [59](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA32.w.10.0.7) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *an attacker must know both the Internet Protocol Address and the TCP port associated with the service*  July 14, 2018 | [59](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA32.w.10.0.7) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *attacker routinely performs a port scan in the opening salvo of any successful cyber assault.*  July 14, 2018 | [59](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA32.w.10.0.77) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *One type of port scan includes sending a TCP SYN packet to a series of common ports and waiting for a TCP ACK response that will result in signaling an open port.*  July 14, 2018 | [59](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA32.w.10.0.77) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *a TCP Connect Scan uses the full three-way handshake to determine the availability of the service or port*  July 14, 2018 | [59](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA32.w.10.0.77.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *TCP Full Connect Scan*  July 14, 2018 | [60](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA33) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *begin by writing our own TCP port scanner that utilizes a TCP full connect scan to identify hosts*  July 14, 2018 | [60](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA33) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *import the Python implementation of BSD socket API.*  July 14, 2018 | [60](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA33) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *to better understand how our TCP Port Scanner works, we will break our script into five unique steps and write Python code for each of them*  July 14, 2018 | [60](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA33.w.6.0.48) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *First, we will input a hostname and a comma separated list of ports to scan*  July 14, 2018 | [60](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA33.w.8.0.22) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Next, we will translate the hostname into an IPv4 Internet*  July 14, 2018 | [60](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA33.w.8.0.22) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *For each port in the list, we will also connect to the target address and specific port*  July 14, 2018 | [60](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA33.w.8.0.22) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *to determine the specific service running on the port, we will send garbage data and read the banner results sent back*  July 14, 2018 | [60](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA33.w.8.0.22) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *The following example shows a quick method for parsing the target hostname and port to scan*  July 14, 2018 | [60](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA33.w.8.0.96.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *build two functions connScan and portScan*  July 14, 2018 | [61](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA34.w.2.0.33) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *portScan function takes the hostname and target ports as arguments. It will first attempt to resolve an IP address to a friendly hostname using the gethostbyname() function*  July 14, 2018 | [61](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA34.w.2.0.33) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *it will print the hostname (or IP address) and enumerate through each individual port attempting to connect using the connScan*  July 14, 2018 | [61](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA34.w.2.0.33) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *connScan function will take two arguments: tgtHost and tgtPort and attempt to create a connection to the target host and por*  July 14, 2018 | [61](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA34.w.3.0.49) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Application Banner Grabbing*  July 21, 2018 | [62](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA35) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *to grab the application banner from our target host, we must first insert additional code into the connScan function*  July 21, 2018 | [62](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA35.w.0.0.0.0.18) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *discovering an open port, we send a string of data to the port and wait for the response*  July 21, 2018 | [62](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA35.w.6.0.21) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *this response might give us an indication of the application running*  July 21, 2018 | [62](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA35.w.6.0.21) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *knowing that the server runs FreeFloat FTP (Version 1.00) this will prove to be useful for targeting*  July 21, 2018 | [63](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA36.w.9.6.0.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Depending on the timeout variable for a socket, a scan of each socket can take several seconds*  July 21, 2018 | [63](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA36.w.14.0.8.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *quickly adds up if we are scanning multiple hosts or ports*  July 21, 2018 | [63](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA36.w.14.0.8.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Threading provides a way to perform these kinds of executions simultaneously*  July 21, 2018 | [63](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA36.w.14.0.8.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *To utilize this in our scan, we will modify the iteration loop in our portScan() function*  July 21, 2018 | [64](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA37) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we call the connScan function as a thread. Each thread created in the iteration*  July 21, 2018 | [64](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA37) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *provides us with a significant advantage in speed, it does present one disadvantage. Our function connScan() prints an output to the screen. If multiple threads print an output at the same time, it could appear garbled and*  July 21, 2018 | [64](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA37.w.4.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *order to allow a function to have complete control of the screen, we will use a semaphore*  July 21, 2018 | [64](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA37.w.4.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *a lock to prevent other threads from proceeding. Notice that prior to printing an output, we grabbed a hold of the lock using screenLock.acquire*  July 21, 2018 | [64](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA37.w.4.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *open, the semaphore will grant us access to proceed and we will print to the screen. If locked, we will have to wait until the thread holding the semaphore releases the lock*  July 21, 2018 | [64](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA37.w.4.0.70) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we now ensure only one thread can print to the screen at any given point in time*  July 21, 2018 | [64](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA37.w.4.0.70) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *preceding example provides a quick script for performing a TCP connect scan. This might prove limited as we may require the ability to perform additional scan*  July 21, 2018 | [66](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA39.w.4.0.3) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *as ACK, RST, FIN, or SYN-ACK scans provided by the Nmap toolkit*  July 21, 2018 | [66](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA39.w.4.0.3) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Nmap, delivers a rather extensive amount of functionality. This begs the question, why not just use Nmap? Enter the true beauty of Python. While Fyodor Vaskovich wrote Nmap and its associated scripts in the C and LUA programming*  July 21, 2018 | [66](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA39.w.6.0.37) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Nmap produces XML based output. Steve Milner and Brian Bustin wrote a Python library that parses this XML based output. This provides us with the ability to utilize the full functionality of Nmap within a Python script. Before starting, you must install Python-Nmap*  July 21, 2018 | [66](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA39.w.6.0.37) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *a few other types of scans. While we lack the tools to craft packets with TCP options*  July 21, 2018 | [66](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA39.w.6.0.161) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *TCP SYN SCAN*  July 21, 2018 | [66](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA39.w.6.0.161) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *known as a half-open scan, this type of scan initiates a TCP connection with a SYN packet and waits for a response*  July 21, 2018 | [66](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA39.w.6.0.161) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *A reset packet indicates the port is closed while a SYN/ACK indicates the port is open*  July 21, 2018 | [67](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA40) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *TCP NULL SCAN—A null scan sets the TCP flag header to zero. If a RST is received, it indicates the port is closed*  July 21, 2018 | [67](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA40) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *TCP FIN SCAN—A TCP FIN Scan sends the FIN to tear down an active TCP connection and wait for a graceful termination. If a RST is received, it indicates the port is closed*  July 21, 2018 | [67](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA40) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *TCP XMAS SCAN—An XMAS Scan sets the PSH, FIN, and URG TCP Flags*  July 21, 2018 | [67](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA40.w.3.0.142) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we can now import Nmap into existing scripts and perform Nmap scans inline with your Python scripts*  July 21, 2018 | [67](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA40.w.3.0.142.0.3) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Creating a PortScanner() class object will allow us the capability to perform a scan on that object*  July 21, 2018 | [67](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA40.w.3.0.142.0.3) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *PortScanner class has a function scan() that takes a list of targets and ports as input and performs a basic Nmap scan*  July 21, 2018 | [67](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA40.w.3.0.142.0.3) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we can now index the object by target hosts and ports and print the status of the port*  July 21, 2018 | [67](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA40.w.4.0.43) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Running our script that utilizes Nmap, we notice something interesting about TCP port 1720. The server or a firewall is actually filtering access to TCP port 1720*  July 21, 2018 | [67](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA40.w.4.0.140.0.22) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *port is not necessarily closed as we initially thought*  July 21, 2018 | [68](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA41) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Building an SSH BotNet with Python*  July 21, 2018 | [68](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA41.w.6.0.14.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *In 1988, RSH provided an excellent (although not very secure) method for a system administrator to remotely connect to a machine and manage it by performing a series of terminal commands on the host*  July 21, 2018 | [68](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA41.w.6.0.14.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Secure Shell (SSH) protocol has since replaced RSH by combining RSH with a public-key cryptographic scheme*  July 21, 2018 | [68](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA41.w.8.0.56) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *this does very little to stop the same attack vector by forcing out common user names and passwords*  July 21, 2018 | [68](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA41.w.8.0.56) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *SSH Worms have proven to be very successful and common attack vectors*  July 21, 2018 | [68](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA41.w.8.0.56) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the attacker has attempted to connect to the machine using the accounts ucla, oxford, and matrix. These are interesting choices*  July 21, 2018 | [68](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA41.w.8.0.125) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Lets implement our own automated SSH Worm that brute forces user credentials against a target*  July 21, 2018 | [69](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA42.w.5.0.0.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *our script must be able to wait and match for an expected output before sending further input commands*  July 21, 2018 | [69](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA42.w.5.0.0.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *to connect to our SSH machine at IP Address, 127.0.0.1, the application first asks us to confirm the RSA key fingerprint. In this case, we must answer, “yes” before*  July 21, 2018 | [69](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA42.w.5.0.0.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *execute our command uname –v to determine the kernel version running on our target*  July 21, 2018 | [69](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA42.w.6.0.58) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Pexpect has the ability to interact with programs, watch for expected outputs, and then respond based on expected outputs*  July 21, 2018 | [69](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA42.w.12.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *an excellent tool of choice for automating the process of brute forcing SSH user credentials*  July 21, 2018 | [69](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA42.w.12.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Examine the function connect(). This function takes a username, hostname, and password and returns an SSH connection*  July 21, 2018 | [70](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA43.w.0.0.0.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the pexpect library, it then waits for an expected output*  July 21, 2018 | [70](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA43.w.0.0.0.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Three possible expected outputs can occur—a timeout, a message indicating that the host has a new public key, or a password prompt*  July 21, 2018 | [70](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA43.w.0.0.0.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *timeout occurs, then the session.expect() method returns to zero*  July 21, 2018 | [70](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA43.w.0.0.0.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *If the child.expect() method catches the ssh\_newkey message, it returns a 1. This forces the function to send a message ‘yes*  July 21, 2018 | [70](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA43.w.1.0.62) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *use a separate function command() to send commands to the SSH session. The function command() takes an SSH session and command string as input*  July 21, 2018 | [70](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA43.w.2.0.57) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *After catching the command prompt, it prints this output from the SSH session.*  July 21, 2018 | [70](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA43.w.5.0.19.0.1) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we now have a script that can connect and control the SSH session interactively*  July 21, 2018 | [71](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA44) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *While we ran the simple command to displaying the hashed password for the root user from /etc/shadow file, we could use the tool to something more devious like using wget*  July 21, 2018 | [72](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA45.w.0.0.0.0.14) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | August 5, 2018 | [72](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA45.w.5.0.37) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Pxssh is a specialized script included the pexpect library. It contains the ability to directly interact with SSH sessions*  July 21, 2018 | [72](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA45.w.5.4.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *only have a few minor modifications to get the script to automate the task of brute forcing SSH credentials.*  July 21, 2018 | [72](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA45.w.9.5.0.0.16) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *If the exception indicates that the password was ‘refused’, we know the password failed and we just return. However, if the exception indicates that the socket is ‘read\_nonblocking’, then we will assume the SSH server is maxed out*  July 21, 2018 | [73](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA46) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Additionally, if the exception indicates that pxssh is having difficulty obtaining a command prompt, we will sleep for a second to allow it to do so. Note that we include a Boolean release included in the connect() function*  July 21, 2018 | [73](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA46.w.3.0.101) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *connect() can recursively call another connect(), we only want the caller to be able to release our connection\_lock semaphore*  July 21, 2018 | [73](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA46.w.3.0.172) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | August 5, 2018 | [74](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA47.w.1.0.168) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Trying the SSH password brute force against a device provides*  July 22, 2018 | [74](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA47.w.1.0.168.0.27) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *interesting to note the password found is ‘alpine’. This is the default root password on iPhone devices. In late 2009, a SSH worm attacked jail-broken iPhones. Often*  July 22, 2018 | [75](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA48) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *jail-breaking the device, users enabled an OpenSSH server on the iPhone. While this proved extremely useful for some, several users were unaware*  July 22, 2018 | [75](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA48) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *The worm iKee took advantage this new capability by trying the default password against devices*  July 22, 2018 | [75](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA48.w.3.0.69) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *they changed the background image of the phone to a picture of Rick Astley with the words “ikee never gonna give you up*  July 22, 2018 | [75](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA48.w.3.0.69) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Additionally, SSH provides the means to authenticate using public key cryptography*  July 22, 2018 | [75](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA48.w.7.1.5) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the server knows the public key and the user knows the private key*  July 22, 2018 | [75](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA48.w.7.1.5) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *either RSA or DSA algorithms, the server produces these keys for logging into SSH. Typically, this provides an excellent method for authentication*  July 22, 2018 | [75](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA48.w.7.1.5) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *1024-bit, 2048-bit, or 4096-bit keys, this authentication process makes it difficult to use brute force*  July 22, 2018 | [75](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA48.w.7.1.69) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *in 2006 something interesting happened with the Debian Linux*  July 22, 2018 | [75](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA48.w.7.1.69.0.1) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *developer commented on a line of code found by an automated software analysis toolkit*  July 22, 2018 | [75](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA48.w.7.1.69.0.1) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *line of code ensured entropy in the creation of SSH*  July 22, 2018 | [75](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA48.w.7.1.69.0.1) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *commenting on the particular line of code, the size of the searchable key space dropped to 15-bits of entropy*  July 22, 2018 | [76](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA49) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *only 15-bits of entropy, this meant only 32,767 keys existed for*  July 22, 2018 | [76](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA49) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Moore, CSO and Chief Architect at Rapid7, generated all of the 1024-bit and 2048 bit keys in under two hours*  July 22, 2018 | [76](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA49) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *he made them available for download at: http://digitaloffense.net/tools/debian-openssl*  July 22, 2018 | [76](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA49) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | August 5, 2018 | [76](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA49.w.2.0.108) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *After downloading and extracting the keys, go ahead and delete the public keys, since we will only need the private keys to test our connection*  July 22, 2018 | [76](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA49.w.2.0.108) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *mistake lasted for 2 years before it was discovered*  July 23, 2018 | [77](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA50) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *accurate to state that quite a few servers were built with a weakened SSH service*  July 23, 2018 | [77](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA50) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *possible to write a small Python script to brute force through each of the 32,767*  July 23, 2018 | [77](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA50) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *to authenticate to a passwordless SSH server that relies upon a public-key cryptograph*  July 23, 2018 | [77](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA50.w.2.0.74) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the Warcat Team wrote such a script*  July 23, 2018 | [77](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA50.w.2.0.74) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Exploit-DB archived the Warcat Team script at: http://www.exploit-db.com/exploits/5720/*  July 23, 2018 | [77](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA50.w.2.0.74) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *script to test weak keys proves nearly very similar to our brute force password*  July 23, 2018 | [77](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA50.w.2.0.134) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *For the following script, we loop through the set of generated keys*  July 23, 2018 | [77](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA50.w.2.0.134) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *connection succeeds, we print the name of the keyfile to the screen*  July 23, 2018 | [77](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA50.w.2.0.134) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we will use two global variables Stop and Fails*  July 23, 2018 | [77](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA50.w.3.0.65) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Fails will keep count of the number of failed connection we have had due to the remote host closing the connection*  July 23, 2018 | [77](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA50.w.3.0.65) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *If this number is greater than 5, we will terminate our script*  July 23, 2018 | [77](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA50.w.3.0.65) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Our Stop global variable is a Boolean that lets us known that we have a found a key and the main() function does not need to start any new connection threads. import*  July 23, 2018 | [77](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA50.w.3.0.65) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | August 5, 2018 | [79](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA52.w.0.0.0.0.17) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Testing this against a target, we see that we can gain access to a vulnerable system*  July 23, 2018 | [79](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA52.w.0.0.0.0.17) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *downloading the 2048 keys as well and using them*  July 23, 2018 | [79](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA52.w.0.0.0.0.17) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we have demonstrated we can control a host via SSH*  July 23, 2018 | [79](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA52.w.4.0.97.0.6) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *let us expand it to control multiple hosts simultaneously. Attackers often use collections of compromised computers*  July 23, 2018 | [79](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA52.w.4.0.97.0.6) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *botnet because the compromised computers act like bots to carry out instructions*  July 23, 2018 | [79](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA52.w.4.0.97.0.6) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *The hacker group, Anonymous, routinely employs the use of a voluntary botnet against their adversaries*  July 23, 2018 | [80](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA53) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *hacker group asks its members to download a tool known as Low Orbit*  July 23, 2018 | [80](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA53) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Ion Cannon (LOIC*  July 23, 2018 | [80](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA53) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the members of Anonymous launch a distributed botnet attack against sites they deem adversaries*  July 23, 2018 | [80](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA53) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *In a recent operation, Operation #Darknet, Anonymous used its voluntary botnet to overwhelm the hosting resources of a site dedicated to distributing child pornography*  July 23, 2018 | [80](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA53.w.3.0.9) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *introduce a new concept—a class. The concept of a class serves as the basis for a programming model named, object oriented programming*  July 23, 2018 | [80](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA53.w.3.0.9) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we instantiate individual objects with associated methods*  July 23, 2018 | [80](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA53.w.3.0.23) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *each individual bot or client will require the ability to connect, and issue a command*  July 23, 2018 | [80](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA53.w.3.0.23) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Examine the code to produce the class object Client(). To build the client requires the hostname, username, and password or key*  July 23, 2018 | [80](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA53.w.4.0.17) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the class contains the methods required to sustain a client—connect(), send\_command(), alive(). Notice that when we reference a variable belonging to a class, we call it self-followed by the variable name.*  July 23, 2018 | [80](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA53.w.6.0.6) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *construct the botnet, we build a global array named botnet and this array contains the individual client*  July 23, 2018 | [80](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA53.w.6.0.6) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we build a function named addClient() that takes a host, user, and password as input to instantiates a client object and add it to the botnet array*  July 23, 2018 | [80](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA53.w.6.0.69) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the botnetCommand() function takes an argument of a command. This*  July 23, 2018 | [80](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA53.w.6.0.69.0.1) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *iterates through the entire array and sends the command to each client*  July 23, 2018 | [80](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA53.w.6.0.69.0.1) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *By wrapping everything up, we have our final SSH botnet script. This proves an excellent method for mass controlling targets*  July 23, 2018 | [82](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA55.w.0.0.0.0.6) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we make three copies of our current Backtrack 5 virtual machine and assign*  July 23, 2018 | [82](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA55.w.0.0.0.0.6) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *script iterate through these three hosts and issue simultaneous commands*  July 23, 2018 | [82](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA55.w.0.0.0.0.6) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the SSH Botnet creation script attacked servers directly, the next section will focus on an indirect attack vector to target clients through vulnerable servers and an alternate approach*  July 23, 2018 | [82](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA55.w.4.0.54) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *massive compromise, dubbed k985ytv, attackers used anonymous and stolen FTP credentials to gain access to 22,400 unique domains and 536,000 infected pages (Huang, 2011). With access granted*  July 23, 2018 | [82](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA55.w.6.9.5.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *attackers injected javascript to redirect benign pages to a malicious domain*  July 23, 2018 | [82](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA55.w.6.9.5.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the malicious Ukrainian host exploited victims in order to install a fake antivirus program that stole credit card information from the clients*  July 23, 2018 | [83](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA56) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Examining the FTP logs of the infected servers, we can see exactly what happened*  July 24, 2018 | [83](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA56) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *automated script connected to the target host in order to determine if it contained*  July 24, 2018 | [83](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA56) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *default page named index.htm*  July 24, 2018 | [83](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA56.w.3.0.17) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the attacker uploaded a new index.htm, presumably containing the malicious redirection script*  July 24, 2018 | [83](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA56.w.3.0.17) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *to better understand the initial vector of this attack, let’s briefly talk about the characteristics of FTP. The File Transfer Protocol (FTP) service*  July 24, 2018 | [83](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA56.w.10.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *allows users to transfer files between hosts in a TCP*  July 24, 2018 | [83](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA56.w.10.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *users authenticate to FTP servers using a combination of a username and password. However, some sites provide the ability to authenticate anonymously*  July 24, 2018 | [83](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA56.w.12.0.35) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *a user enters the username “anonymous” and submits an email address*  July 24, 2018 | [83](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA56.w.12.0.35) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *it seems insane that any sites would offer anonymous FTP*  July 24, 2018 | [83](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA56.w.12.0.35.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *many sites surprisingly provide legitimate reasons for this kind of FTP access such as promoting the idea that this enables a more enhanced means of accessing software updates*  July 24, 2018 | [84](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA57) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *utilize the ftplib library in Python in order to build a small script to determine if a server offers anonymous logins*  July 24, 2018 | [84](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA57) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *function anonLogin() takes a hostname and returns a Boolean that describes the availability of anonymous logins*  July 24, 2018 | [84](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA57) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *to determine this Boolean, the function attempts to create an FTP connection with anonymous*  July 24, 2018 | [84](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA57.w.2.0.77) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | August 5, 2018 | [84](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA57.w.5.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *attackers also have been quite successful with using stolen credentials to gain access to legitimate FTP servers*  July 24, 2018 | [84](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA57.w.5.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *programs, such as FileZilla, often store passwords in plaintext configuration files*  July 24, 2018 | [84](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA57.w.5.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *HD Moore even included the get\_filezilla\_creds.rb script in a recent Metasploit*  July 24, 2018 | [84](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA57.w.7.0.42) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Imagine a text file of a username/password combination we wanted*  July 24, 2018 | [84](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA57.w.7.0.42.0.1) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *imagine the username/password combinations stored in a flat text file. administrator:password*  July 24, 2018 | [85](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA58) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *expand upon our early anonLogin() function to build one called bruteLogin(*  July 24, 2018 | [85](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA58) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *function will take a host and password file as input and return the credentials that allow access to the host*  July 24, 2018 | [85](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA58) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *function iterates through each line of the file, splitting each line at the colon*  July 24, 2018 | [85](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA58.w.4.0.28) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *and attempts to login to the FTP server. If it succeeds, it returns a tuple of a username, password*  July 24, 2018 | [85](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA58.w.4.0.28) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *If the function exhausted all lines and failed to successfully login, it returns a tuple of None,None. import*  July 24, 2018 | [85](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA58.w.4.0.28) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *credentials on the FTP server, we must now test if the server also provides web access*  July 24, 2018 | [86](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA59) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *first list the contents of the FTP server’s directory and search for default web pages. The function returnDefault() takes*  July 24, 2018 | [86](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA59.w.4.0.14) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *The function checks each file returned by NLST against default web page file names*  July 24, 2018 | [86](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA59.w.4.0.14) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *also appends any discovered default pages to an array called retList. After*  July 24, 2018 | [86](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA59.w.4.0.83) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | August 5, 2018 | [86](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA59.w.5.4.1.0.24) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *see it has three webpages in the base directory*  July 24, 2018 | [86](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA59.w.5.4.1.0.24) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *We’ll now move on to infecting these pages with our client side attack vector*  July 24, 2018 | [87](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA60) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we have found web page files, we must infect them with a malicious redirect*  July 24, 2018 | [87](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA60) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *use the Metasploit framework in order to quickly create a malicious server and page hosted at http://10.10.10.112:8080/exploit. Notice*  July 24, 2018 | [87](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA60.w.4.0.24) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *choose the exploit ms10\_002\_aurora, the very same exploit used during Operation Aurora against Google*  July 24, 2018 | [87](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA60.w.4.0.24) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *exploit redirected victims, which will provide a call back to our command and control server*  July 24, 2018 | [87](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA60.w.4.0.24) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *connects to our server at http://10.10.10.112:8080/exploit will now fall prey to our exploit*  July 24, 2018 | [87](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA60.w.6.4.5) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *succeeds, it will create a reverse TCP shell and grant us access to the Windows command prompt*  July 24, 2018 | [87](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA60.w.6.4.5) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we must add a redirect from the benign infected servers to our malicious exploit server*  July 24, 2018 | [88](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA61) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we can download the default pages found on the benign server, inject an iframe, and upload the malicious pages*  July 24, 2018 | [88](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA61) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *The function injectPage() takes an FTP connection, a page name, and a redirect iframe string as the input. It then downloads a temporary copy of*  July 24, 2018 | [88](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA61.w.6.0.33) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *it appends the iframe*  July 24, 2018 | [88](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA61.w.6.0.33) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the function uploads the infected page back to the benign server*  July 24, 2018 | [88](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA61.w.6.0.33) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | August 5, 2018 | [88](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA61.w.7.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Running our code, we see it download the index.html*  July 24, 2018 | [88](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA61.w.11.1.1.0.14) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *and inject it with our malicious content*  July 24, 2018 | [88](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA61.w.11.1.1.0.14) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *wrap up our entire attack in the attack() function. The attack() function takes a username, password, hostname, and redirect location as input*  July 24, 2018 | [89](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA62) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *The function first logs onto the FTP server with the credentials*  July 24, 2018 | [89](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA62) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we have the script search for default web pages*  July 24, 2018 | [89](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA62) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *For each of these pages, the script downloads a copy and adds a malicious redirection*  July 24, 2018 | [89](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA62) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *then uploads the infected page back to the FTP server*  July 24, 2018 | [89](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA62.w.5.0.50) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Adding some option parsing, we wrap up the entire script*  July 24, 2018 | [89](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA62.w.6.0.18) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we first try to gain anonymous access to the FTP server. If this fails, we then brute force credentials*  July 24, 2018 | [89](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA62.w.6.0.18) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *only a hundred lines of code, this attack fully replicates the original attack vector of the k985ytv infection*  July 24, 2018 | [89](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA62.w.6.0.18) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | August 5, 2018 | [92](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA65) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Running our script against a vulnerable FTP server, we see it brute attempt anonymous logon*  July 24, 2018 | [92](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA65.w.0.0.0.0.24) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *fail, enumerate the password guest/guest, and then download and inject every*  July 24, 2018 | [92](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA65.w.3.0.9) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | August 5, 2018 | [92](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA65.w.5.2.47) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *ensure our client side attack vector is running and wait for a victim to connect the now infected webserver*  July 24, 2018 | [92](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA65.w.5.2.47) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *km985ytv did successfully compromise 2220 of the 11,000*  July 24, 2018 | [93](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA66.w.2.0.62.0.10) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Fake Antivirus captured the credit cards of over 43 million people by 2009*  July 24, 2018 | [93](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA66.w.3.0.29) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *November of 2008, computer security experts woke up to an interesting and game-changing worm*  July 24, 2018 | [93](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA66.w.3.0.29.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *The Conficker or W32DownandUp Worm spread so rapidly that it infected five million computers*  July 24, 2018 | [93](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA66.w.5.0.15) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *some of the advanced methods (digital signatures, encrypted payloads, and alternative propagation schemes) aided in the attack, Conficker at its very heart, holds some similarities*  July 24, 2018 | [93](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA66.w.5.0.15) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *to the Morris Worm*  July 24, 2018 | [93](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA66.w.5.0.15) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we will recreate the primary attack vectors*  July 24, 2018 | [93](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA66.w.5.0.75) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *its base infection routine, Conficker utilized two separate attack vectors*  July 24, 2018 | [93](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA66.w.5.0.75) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *First, it utilized a zero-day exploit for the Windows server service vulnerability*  July 24, 2018 | [93](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA66.w.5.0.75.0.3) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *this vulnerability allowed the worm to cause a stack corruption that executed shellcode and downloaded a copy of it to the infected host*  July 24, 2018 | [93](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA66.w.5.0.75.0.3) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *failed, Conficker attempted to gain access to a victim by brute forcing credentials to the default administrative network share (ADMIN$)*  July 24, 2018 | [94](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA67) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Conficker utilized a password list of over 250 common passwords. The Morris Worm used a password list of 432 passwords. These*  July 24, 2018 | [94](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA67) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *share 11 common passwords on the list. When*  July 24, 2018 | [94](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA67) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *eleven passwords*  July 24, 2018 | [94](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA67.w.3.0.34) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *these password attempts are undoubtedly illegal, these passwords dumps have proven interesting research for security experts. DARPA Cyber Fast Track Project Manager, Peiter Zatko*  July 24, 2018 | [94](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA67.w.3.0.34) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *room full of Army Brass blush when he asked them if they constructed their passwords using a combination of two capitalized words following by two special character and two numbers*  July 24, 2018 | [94](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA67.w.5.0.25) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the hacker group LulzSec released 26,000 passwords and personal information*  July 24, 2018 | [94](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA67.w.5.0.25) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *coordinated strike, several of these passwords were reused to attack the social networking sites of the same individuals*  July 24, 2018 | [94](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA67.w.5.0.86) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the most prolific attack was the release of over 1 million usernames and passwords for Gawker*  July 24, 2018 | [94](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA67.w.5.0.86) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *simplify our attack we will utilize the Metasploit Framework*  July 24, 2018 | [94](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA67.w.8.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *open source computer security project, Metasploit, has risen to quick popularity to become the de facto exploitation toolkit over the last eight years*  July 24, 2018 | [94](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA67.w.8.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Championed and developed by the legendary exploit writer, HD Moore*  July 24, 2018 | [94](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA67.w.8.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *While attacks can be interactively driven using Metasploit, it also has the capability to read in a resource batch*  July 24, 2018 | [94](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA67.w.8.0.56.0.1) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Metasploit sequentially processes the commands for the batch file in order to execute an attack. Consider*  July 24, 2018 | [94](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA67.w.8.0.56.0.1) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *if we want to attack a target at our victim host 192.168.13.37*  July 24, 2018 | [95](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA68) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *using the ms08\_067\_netapi (Conficker) exploit in order to deliver a shell*  July 24, 2018 | [95](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA68) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *utilize Metasploit’s attack, we first chose our exploit*  July 24, 2018 | [95](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA68.w.3.0.8) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *then set the target*  July 24, 2018 | [95](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA68.w.3.0.8) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Following target selection, we indicated the payload as*  July 24, 2018 | [95](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA68.w.3.0.8) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *and selected a reverse connection to our host at*  July 24, 2018 | [95](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA68.w.3.0.8) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Finally, we told Metasploit to exploit the system*  July 24, 2018 | [95](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA68.w.3.0.8) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Saving the configuration file to the filename conficker.rc, we can now launch our attack by issuing the command msfconsole –r conficker.rc*  July 24, 2018 | [95](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA68.w.5.0.45) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *successful, our attack returns a Windows command shell to control the machine*  July 24, 2018 | [95](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA68.w.5.0.45) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | August 5, 2018 | [95](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA68.w.6.0.8) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *We built a configuration file, exploited a machine and gained a shell*  July 24, 2018 | [95](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA68.w.6.1.7.0.20) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *process for 254 hosts might take us quite a bit of time in order to type out a configuration file, but if we use Python again, we can generate a quick script to scan for hosts that have TCP port 445 open and then build a Metasploit*  July 24, 2018 | [96](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA69) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *file to attack all the vulnerable hosts*  July 24, 2018 | [96](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA69) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *First, lets use the Nmap-Python module from our previous portscanner*  July 24, 2018 | [96](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA69) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the function findTgts,() takes an input of potential target hosts and returns all the hosts*  July 24, 2018 | [96](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA69.w.4.0.9) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *TCP port 445 open*  July 24, 2018 | [96](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA69.w.4.0.9) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *TCP port 445 serves as a primary port for the SMB protocol*  July 24, 2018 | [96](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA69.w.4.0.9) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *filtering only the hosts that have a TCP port 445 open, our attack script can now target only valid ones*  July 24, 2018 | [96](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA69.w.4.0.9) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *If the function finds a host with a TCP open, it appends that host to an array*  July 24, 2018 | [96](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA69.w.4.0.80) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *completing the iteration, the function returns this array*  July 24, 2018 | [96](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA69.w.4.0.80) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we will set up a listener for our exploited targets. This listener, or command and control channel*  July 24, 2018 | [96](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA69.w.5.0.18) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *allow us to remotely interact with our target hosts once they are exploited*  July 24, 2018 | [96](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA69.w.5.0.18) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *an advanced and dynamic payload known as the Meterpreter*  July 24, 2018 | [96](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA69.w.5.0.18) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the Metasploit Meterpreter, calls back to our command and control host and provides a wealth of functionality to analyze and control the infected target*  July 24, 2018 | [96](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA69.w.6.0.39) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *extensions provide the ability to look for forensic objects, issue commands, route traffic through the infected host, install a key-logger, or dump the password hashes*  July 24, 2018 | [96](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA69.w.6.0.39) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *a Meterpreter process connects back to the attacker for command and control it a Metasploit module called the multi/handler*  July 24, 2018 | [97](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA70) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *multi/handler listener on our machine, we will first need to write the instructions to our Metasploit resource configuration*  July 24, 2018 | [97](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA70) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we set the payload as a reverse\_tcp connection and then indicate our local host address and port we wish to receive the connection on*  July 24, 2018 | [97](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA70) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *set a global configuration DisablePayloadHandler to indicate that all future hosts do not need to set up a handler*  July 25, 2018 | [97](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA70.w.2.0.64) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the script has reached the point of being able to launch exploits against the target*  July 25, 2018 | [97](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA70.w.3.0.11) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *function will input a Metasploit configuration file, a target, and the local address and ports for the exploit*  July 25, 2018 | [97](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA70.w.3.0.11) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *function will write the particular exploit settings to the configuration file*  July 25, 2018 | [97](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA70.w.3.0.11) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *first selects the particular exploit, ms08\_067\_netapi, used in the Conficker attack against the target or RHOST. Additionally, it chooses the Meterpreter payload and the local address (LHOST) and port (LPORT) required*  July 25, 2018 | [97](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA70.w.5.0.37) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *it sends an instruction to exploit the machine under the context of a job (-j) and to not interact with the job immediately (-z*  July 25, 2018 | [97](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA70.w.5.0.37) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *ms08\_067\_netapi exploit against victims around the world, a defender can easily prevent it with current security patches*  July 25, 2018 | [97](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA70.w.6.0.14.0.12) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the script will require the second attack vector used in the Conficker Worm. It will need to brute force through SMB username/password*  July 25, 2018 | [98](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA71) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *function smbBrute takes the Metasploit configuration file, the target host*  July 25, 2018 | [98](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA71) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *second file*  July 25, 2018 | [98](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA71) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *containing a list of passwords, and the local address and port for the listener*  July 25, 2018 | [98](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA71) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *sets the username as the default windows Administrator and then opens the password file.*  July 25, 2018 | [98](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA71.w.2.0.85) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *each password in the file, the function builds a Metasploit resource configuration in order to use the remote process execution (psexec) exploit*  July 25, 2018 | [98](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA71.w.2.0.85) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *script now has the ability to scan for possible targets and exploit them using the MS08\_067 vulnerability and/or brute force through a list*  July 25, 2018 | [98](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA71.w.4.1.27) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *add some option parsing back to the main() function of the script and then call the previous written functions as required to wrap up the entire script*  July 25, 2018 | [98](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA71.w.7.0.18) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *what happens when you encounter a target with no known exploit? How do you build your own zero-day attack*  July 25, 2018 | [100](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA73.w.4.0.49) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we will construct our own zero-day attack*  July 25, 2018 | [100](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA73.w.4.0.49) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *preceding section and the Conficker worm made use of a stack corruption vulnerability.*  July 25, 2018 | [101](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA74.w.5.0.7.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *you may encounter a time when you have to write your own*  July 25, 2018 | [101](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA74.w.5.0.7.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *In order to do so, lets begin by understanding stack-based buffer overflows.*  July 25, 2018 | [101](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA74.w.7.0.51) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Morris Worm succeeded in part because of a stack-based buffer overflow against the Finger service*  July 25, 2018 | [101](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA74.w.7.0.51) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *class of exploits succeeds because a program fails to sanitize or validate a user input.*  July 25, 2018 | [101](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA74.w.7.0.51) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *made use of a stack-based buffer overflow attack in 1988, it was not until 1996 that Elias Levy (a.k.a. Aleph One) published the seminal paper,*  July 25, 2018 | [101](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA74.w.7.0.51) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the case of a stack-based buffer overflow, unchecked user data overwrites the next instruction pointer [EIP]*  July 25, 2018 | [102](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA75) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *The exploit directs the EIP register to point to a location containing shellcode inserted by the attacker.*  July 25, 2018 | [102](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA75) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Endless shellcode possibilities exist, solely depending on the size of available space in memory*  July 25, 2018 | [102](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA75.w.3.0.55) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Essential elements of stack-based buffer overflow exploit*  July 25, 2018 | [102](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA75.w.3.0.55) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Overflow: user input that exceeds the expected value allotted in the stack.*  July 25, 2018 | [102](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA75.w.3.0.55) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Return Address: The 4-byte address used to jump directly to the top of the stack. In the following exploit, we use a 4-byte address that points to a JMP ESP*  July 25, 2018 | [102](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA75.w.3.0.55) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Padding: A series of NOP (no operation) instructions that precedes the shellcode, allowing an attacker to guestimate*  July 25, 2018 | [102](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA75.w.3.0.78) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *address location to jump directly to. If an attacker lands anywhere in the NOP-sled, he slides directly in*  July 25, 2018 | [102](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA75.w.3.0.78) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Shellcode: A small piece of code written in assembly machine code*  July 25, 2018 | [102](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA75.w.4.0.13) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *stack-based buffer overflows provided the original exploit vector.*  July 25, 2018 | [102](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA75.w.4.0.13) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *an exploit for a vulnerable FTP server to packetstorm (Freyman, 2011). Although the development of the exploit may appear to be a complex task, the actual attack contains less than eighty lines*  July 25, 2018 | [102](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA75.w.4.0.13) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *begin by building the key elements of our exploit. First we set our shellcode variable to contain the hexadecimal*  July 25, 2018 | [102](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA75.w.4.0.38) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *for a payload we created with the Metasploit*  July 25, 2018 | [102](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA75.w.6.0.20) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *set our overflow variable to contain 246 instances of the letter “A” (\x41 in hex)*  July 25, 2018 | [102](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA75.w.6.0.20) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Our return address variable points to an address location in kernel32.dll containing an instruction that jumps directly to the top of the stack*  July 25, 2018 | [102](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA75.w.6.0.20) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Our padding variable contains a series of 150 NOP instructions*  July 25, 2018 | [102](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA75.w.6.0.20) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *assemble all of these variables together into a variable we call crash.*  July 25, 2018 | [102](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA75.w.6.0.85) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the Berkeley Socket API, we will create a connection*  July 25, 2018 | [103](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA76.w.2.0.24) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *TCP port 21 on our target host. If this connection succeeds, we will then authenticate to the host by sending an anonymous username and password.*  July 25, 2018 | [103](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA76.w.2.0.24) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *send the FTP command “RETR” followed by our crash variable. Since the affected program does not properly sanitize user input, this will result in a stack-based buffer overflow that overwrites the EIP register*  July 25, 2018 | [103](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA76.w.4.0.29) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *)After downloading a copy of a FreeFloat FTP to either a Windows XP SP2 or SP3 machine, we can test Craig Freyman’s exploit*  July 25, 2018 | [106](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA79) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we will run our exploit script and use the netcat utility to connect to port 4444 on the target host*  July 25, 2018 | [106](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA79.w.5.0.3) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | August 5, 2018 | [106](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA79.w.6.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *let’s examine how we can write some scripts to aid us in Forensic investigations*  July 25, 2018 | [106](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA79.w.9.0.23) | |

## *Chapter 3. Forensic Investigations with Python*

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Information in this chapter: Geo-Location through the Windows Registry Recycle Bin Investigation Examining Metadata in PDFs and Microsoft Documents*  July 25, 2018 | [108](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA81.w.0.0.0.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Let’s begin by using some unique Windows Registry keys to physically track a user*  July 25, 2018 | [109](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA82.w.6.0.31) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Where Have You Been?—Analysis of Wireless Access Points in the Registry*  July 25, 2018 | [109](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA82.w.7.0.0.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Windows registry contains a hierarchical database that stores the configuration settings*  July 25, 2018 | [109](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA82.w.7.0.0.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *advent of wireless networking, the Windows Registry stores information related to the wireless connection*  July 25, 2018 | [109](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA82.w.7.0.0.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *these registry keys can provide us with geo-location information about where a laptop has been*  July 25, 2018 | [109](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA82.w.7.0.0.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *From Windows Vista on, the Registry stores each of the networks in subkey under HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\NetworkList\Signatures\Unmanaged*  July 25, 2018 | [109](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA82.w.8.0.46) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *list each of the networks, showing the profile Guid, network description, network name, and gateway MAC address*  July 25, 2018 | [109](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA82.w.8.0.46) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *For each profile, it contains the following sub-keys: ProfileGuid, Description, Source, DnsSuffix, FirstNetwork, DefaultGatewayMac*  July 25, 2018 | [110](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA83.w.6.0.17) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we now have a script that will print out the previously connected wireless networks stored in the Windows Registry*  July 25, 2018 | [110](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA83.w.7.0.15) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *script against a target laptop, we see the previously connected wireless networks along with their MAC addresses*  July 26, 2018 | [111](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA84.w.6.3.12.0.10) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *running from inside an Administrator console or you will be unable to read the keys*  July 26, 2018 | [111](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA84.w.6.3.12.0.10) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *With the MAC address of a wireless access point, we can now also print out the physical location of the access point as well*  July 26, 2018 | [112](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA85) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *enormous listings of wireless access points correlated to their physical locations*  July 26, 2018 | [112](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA85.w.4.0.35) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *products such as cell phones use these databases to geo-locate without the use of GPS*  July 26, 2018 | [112](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA85.w.4.0.35) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *SkyHook database*  July 26, 2018 | [112](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA85.w.4.0.35) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *a software developer kit to geo-locate based off of Wi-Fi positioning*  July 26, 2018 | [112](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA85.w.4.0.35) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *open-source project developed by Ian McCracken provided access to this database*  July 26, 2018 | [112](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA85.w.4.0.35) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Google also maintained a similarly large database for the purpose of correlating access-point MAC addresses to physical locations*  July 26, 2018 | [112](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA85.w.5.0.22) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Gorjan Petrovski developed an NMAP NSE script to interact with it, Google deprecated open source*  July 26, 2018 | [112](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA85.w.5.0.22) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Microsoft locked down a similar Wi-Fi geo-location database*  July 26, 2018 | [112](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA85.w.5.0.79) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *remaining database and open-source project, wigle.net, continues to allow users*  July 26, 2018 | [112](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA85.w.5.0.79) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *After registering for an account, a user can interact with wigle.net with a little creative Python scripting*  July 26, 2018 | [112](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA85.w.5.0.79) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *a user will quickly realize that he or she must interact with three separate pages in order to return a Wigle*  July 26, 2018 | [112](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA85.w.6.0.23) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *initial page at http://wigle.net; next the user must log in to*  July 26, 2018 | [112](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA85.w.6.0.23) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the user can query a specific wireless SSID MAC address*  July 26, 2018 | [112](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA85.w.7.0.40) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Capturing the MAC address query, we see that the netid parameter contains the MAC address in*  July 26, 2018 | [112](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA85.w.7.0.40) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we see the response from the page includes the GPS coordinates*  July 26, 2018 | [113](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA86) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *have enough to build a simple function that will return the latitude and longitude of a wireless access point*  July 26, 2018 | [113](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA86.w.4.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *mechanize allows stateful web programming in Python*  July 26, 2018 | [113](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA86.w.7.0.27) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *correctly log on to the Wigle service, it will store and reuse the authentication cookie for us*  July 26, 2018 | [113](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA86.w.7.0.27) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *let’s quickly walk through it together. First, we create an instance of a mechanize browser. Next, we open the initial wigle.net page. We then encode our username and password as parameters and request a login*  July 26, 2018 | [113](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA86.w.7.0.27) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *successfully logged in, we create an HTTP post using the parameter netid*  July 26, 2018 | [113](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA86.w.8.0.16) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *to search the database*  July 26, 2018 | [113](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA86.w.8.0.16) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *search the result of our HTTP post for the terms maplat= and maplon= for our latitude and longitude*  July 26, 2018 | [113](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA86.w.8.0.16) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *return these coordinates as a tuple*  July 26, 2018 | [113](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA86.w.8.0.88) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Adding the Wigle MAC address functionality to our original script*  July 26, 2018 | [114](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA87.w.5.0.2.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the ability to examine a registry for previously connected wireless access points*  July 26, 2018 | [114](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA87.w.5.0.2.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *then look up their physical locations*  July 26, 2018 | [114](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA87.w.5.0.2.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *script with the new functionality, we now see the previously connected wireless networks*  July 26, 2018 | [116](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA89.w.2.0.0.0.5) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *With the knowledge of where a computer has been, let’s now use the next section to examine the trash*  July 26, 2018 | [116](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA89.w.2.0.0.0.5) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Microsoft Operating Systems, the Recycle Bin serves as a special folder that contains deleted files*  July 26, 2018 | [116](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA89.w.4.0.12.0.10) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *marking them for deletion but not actually removing them*  July 26, 2018 | [116](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA89.w.6.0.23) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *with a FAT file system, the C:\Recycled*  July 26, 2018 | [116](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA89.w.6.0.23) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *support NTFS, including Windows NT, 2000, and XP, store*  July 26, 2018 | [116](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA89.w.6.0.23) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *C:\Recycler\ directory*  July 26, 2018 | [116](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA89.w.6.0.23) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Vista and 7 store the directory at C:\$Recycle.Bin.*  July 26, 2018 | [116](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA89.w.6.0.23) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *let’s write a function to test each of the possible candidate directories and return the first one that exists*  July 26, 2018 | [116](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA89.w.6.0.89.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we will need to inspect its contents. Notice the two subdirectories*  July 26, 2018 | [117](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA90) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *They both contain the string*  July 26, 2018 | [117](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA90) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *and terminate with*  July 26, 2018 | [117](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA90) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *This string represents the user SID, corresponding to a unique user account on the machine*  July 26, 2018 | [117](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA90) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *use the windows registry to translate this SID into an exact username*  July 26, 2018 | [117](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA90.w.9.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *inspecting the windows registry key HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList\\ProfileImagePath*  July 26, 2018 | [117](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA90.w.9.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *it return a value of %SystemDrive%\Documents and Settings\*  July 26, 2018 | [117](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA90.w.9.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *write a small function to translate each SID into a username. This*  July 26, 2018 | [118](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA91.w.6.0.1) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *function will open the registry to examine the ProfileImagePath Key, find the value and return the name located after the last backward slash*  July 26, 2018 | [118](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA91.w.6.0.1) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we will put all of our code together to create a script that will print the deleted files*  July 26, 2018 | [118](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA91.w.7.0.67) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *our code inside a target, we see that the script discovers two users: alex and Administrator*  July 26, 2018 | [119](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA92.w.1.0.52) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *next section, we will examine a method for examining some of the content inside of those files*  July 26, 2018 | [119](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA92.w.1.0.52) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *will write some scripts to extract metadata from some files*  July 26, 2018 | [119](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA92.w.3.2.23.0.8) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *authoring application may store details such as the file’s authors, creation and modification times, potential revisions, and comments*  July 26, 2018 | [119](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA92.w.3.2.23.0.8) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *From The TrenchesAnonymous’ Metadata*  July 26, 2018 | [120](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA93) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *December 10, 2010, the hacker group Anonymous posted a press release outlining the motivations behind a recent attack named Operation Payback*  July 26, 2018 | [120](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA93) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *companies that had dropped support for the Web site WikiLeaks, Anonymous called for retaliation by performing a distributed denial of service*  July 26, 2018 | [120](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA93.w.11.0.14) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *posted the press release unsigned and without attribution*  July 26, 2018 | [120](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA93.w.11.0.14) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *addition to the program used to create the document, the PDF metadata contained the name of the author, Mr. Alex Tapanaris. Within days, Greek police*  July 26, 2018 | [120](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA93.w.11.0.75) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *recreate the forensic investigation of a document*  July 26, 2018 | [120](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA93.w.11.0.75) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *We can start by downloading the document using the wget utility*  July 26, 2018 | [120](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA93.w.11.0.95) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *PYPDF is an excellent third-party utility for managing PDF*  July 26, 2018 | [121](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA94) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *ability to extract document information, split, merge, crop, encrypt and decrypt documents*  July 26, 2018 | [121](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA94) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we utilize the method .getDocumentInfo(). This method returns an array of tuples*  July 26, 2018 | [121](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA94) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *description of the metadata element and its value*  July 26, 2018 | [121](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA94) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Adding an option parser to identify a specific file, we have a tool that can identify the metadata embedded in a PDF document*  July 26, 2018 | [121](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA94.w.5.0.6) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *modify our script to test for specific metadata, such as a specific user*  July 26, 2018 | [121](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA94.w.5.0.6) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *exchange image file format (Exif) standard defines the specifications for how to store image and audio file*  July 26, 2018 | [122](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA95) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *this standard to save audio or image files*  July 26, 2018 | [122](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA95.w.7.1.15) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Exif standard contains several useful tags for a forensic investigation*  July 26, 2018 | [122](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA95.w.7.1.15) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Phil Harvey wrote a tool aptly named exiftool*  July 26, 2018 | [122](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA95.w.7.1.15) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Examining all the Exif tags in a photo could result in several pages of information*  July 26, 2018 | [122](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA95.w.7.1.15) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *this information also has plenty of malicious uses. Imagine a soldier placing Exif-tagged photos on a blog or a Web site: the enemy could download entire sets of photos and know all of that soldier’s movements in seconds*  July 26, 2018 | [122](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA95.w.7.1.140) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we will build a script to connect to a Web site, download all the images on the site, and then check them for Exif metadata*  July 26, 2018 | [122](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA95.w.7.1.140) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Beautiful Soup allows us to quickly parse HTML and XML documents. Leonard Richardson*  July 26, 2018 | [123](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA96.w.0.0.0.0.14) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Beautiful Soup to scrape the contents of an HTML document for all the images found on the document*  July 26, 2018 | [123](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA96.w.4.0.5) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *using the urllib2 library to open the contents of a document and read it*  July 26, 2018 | [123](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA96.w.4.0.5) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we can create a Beautiful Soup object*  July 26, 2018 | [123](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA96.w.4.0.5) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *or a parse tree that contains the different objects of the HTML document. In that object, we will extract*  July 26, 2018 | [123](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA96.w.4.0.5) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *image tags by searching using the method .findall(‘img’). This*  July 26, 2018 | [123](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA96.w.6.0.60) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we need to download each image from the site in order to examine them in a separate function*  July 26, 2018 | [124](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA97.w.0.0.0.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we will use the functionality in*  July 26, 2018 | [124](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA97.w.0.0.0.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *urllib2, urlparse, and os libraries*  July 26, 2018 | [124](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA97.w.0.0.0.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we will extract the source address from the image tag*  July 26, 2018 | [124](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA97.w.0.0.0.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we will read the binary contents of the image into a variable*  July 26, 2018 | [124](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA97.w.0.0.0.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we will open a file in write-binary mode and write the contents of the image*  July 26, 2018 | [124](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA97.w.0.0.0.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *test the contents of an image file for Exif Metadata, we will process the file using*  July 26, 2018 | [124](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA97.w.3.0.110) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Python Imaging Library. PIL, available*  July 26, 2018 | [124](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA97.w.3.0.110) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *adds image-processing capabilities to Python, and allows us to quickly extract the metadata associated with geo-location information*  July 26, 2018 | [124](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA97.w.3.0.110) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *test a file for metadata, we will open the object as a PIL Image and use the method \_getexif()*  July 26, 2018 | [124](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA97.w.6.0.47) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Next, we parse the Exif data into an array, indexed by the metadata type*  July 26, 2018 | [124](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA97.w.6.0.47) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *With the array complete, we can search the array to see if it contains an Exif tag*  July 26, 2018 | [124](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA97.w.6.0.47) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *script is now able to connect to a URL address, parse and download all the images files, and test each file for Exif metadata*  July 26, 2018 | [125](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA98.w.0.0.0.0.6) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *in the main function, we first fetch a list of all the images on the site. Then*  July 26, 2018 | [125](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA98.w.0.0.0.0.6) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *for each image in the array, we will download the file and test it for GPS metadata*  July 26, 2018 | [125](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA98.w.0.0.0.0.6) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Testing the newly created script against a target address, we see that*  July 26, 2018 | [126](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA99.w.2.4.7.0.29) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *images on the target contains GPS metadata information. While*  July 26, 2018 | [127](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA100) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *can be used in an offensive reconnaissance*  July 26, 2018 | [127](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA100) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *in a completely benign way—to identify our own*  July 26, 2018 | [127](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA100) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we will examine application artifacts, namely data stored in SQLite Databases*  July 26, 2018 | [127](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA100.w.1.5.4.0.22) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *SQLite database is a popular choice for local/client storages on several different applications*  July 26, 2018 | [127](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA100.w.3.0.7) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *because of the programming-language-independent bindings*  July 26, 2018 | [127](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA100.w.3.0.7) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *SQLite stores the entire database as a single flat file on the host*  July 26, 2018 | [127](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA100.w.3.0.7) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *next section begins by examining the SQLite database format used in the popular Skype voice-over-ip, chat*  July 26, 2018 | [127](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA100.w.3.0.121) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *version 4.0, the popular chat utility Skype changed its internal database format to use SQLite*  July 26, 2018 | [127](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA100.w.3.0.121) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Skype stores a database named main.db in*  July 26, 2018 | [127](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA100.w.3.0.121) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *SQLite database maintains a table named sqlite\_master; this table contains a column named tbl\_name*  July 26, 2018 | [128](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA101) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Executing this SELECT statement allows us to see tables in the Skype main.db database*  July 26, 2018 | [128](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA101) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *this database holds tables containing information about contacts, calls, accounts, messages, and even SMS messages*  July 26, 2018 | [128](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA101) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Accounts contains information about the Skype account used by the application*  July 26, 2018 | [128](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA101.w.3.13.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Unixepoch time provides a simple measurement for time. It records the date as a simple integer that represents the number of seconds since January 1st, 1970. The SQL method datetime() can convert this value*  July 26, 2018 | [129](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA102) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *connecting to the database and executing a SELECT statement proves easy enough, we would like to be able to automate this process*  July 26, 2018 | [129](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA102.w.2.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *write a small Python program that utilizes the sqlite3 library to do this*  July 26, 2018 | [129](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA102.w.4.0.31) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *our function printProfile(). It creates a connection to the database main.db*  July 26, 2018 | [129](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA102.w.4.0.31) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *it asks for a cursor prompt and executes our previous SELECT statement*  July 26, 2018 | [129](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA102.w.4.0.31) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *result of the SELECT statement returns an array of arrays*  July 26, 2018 | [129](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA102.w.4.0.31) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *each result returned, it contains indexed columns for the user*  July 26, 2018 | [129](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA102.w.4.0.31) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Notice that the table Contacts stores information*  July 26, 2018 | [130](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA103) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *All of this personally identifiable information can prove useful as we investigate or attack a target*  July 26, 2018 | [130](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA103.w.4.0.18) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *what happens when two tables contain information that we want to output togethe*  July 26, 2018 | [130](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA103.w.7.0.12) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *join the database tables with values that uniquely identify the*  July 26, 2018 | [130](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA103.w.7.0.12) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *how to output the call log stored in the skype database*  July 26, 2018 | [130](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA103.w.7.0.12) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *detailed Skype call log, we will need to use both the Calls table and the Conversations*  July 26, 2018 | [130](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA103.w.7.0.12) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we need to issue a SELECT statement with a condition WHERE calls.conv\_dbid = conversations.id. The*  July 26, 2018 | [131](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA104) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *add one final function to our Skype database scrapping*  July 26, 2018 | [131](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA104.w.2.5.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Skype profile database actually contains all the messages*  July 26, 2018 | [131](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA104.w.2.5.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *stores this in a table named Messages*  July 26, 2018 | [131](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA104.w.2.5.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we will SELECT the timestamp, dialog\_partner, author, and body\_xml*  July 26, 2018 | [131](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA104.w.2.5.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *if the author differs from the dialog\_partner, the owner of the database initiated the message*  July 26, 2018 | [131](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA104.w.5.0.52) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *script can print the profile information, address contacts, call log, and even the messages stored in the database*  July 26, 2018 | [132](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA105) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *can add some option parsing*  July 26, 2018 | [132](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA105) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *some of the functionality in the os library*  July 26, 2018 | [132](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA105.w.4.0.36) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *ensure the profile file exists before executing each of the functions to*  July 26, 2018 | [132](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA105.w.4.0.36) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *()Running the script, we add the location of a Skype profile path with the –p opti*  July 26, 2018 | [134](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA107) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *e next section, we will use our knowledge of sqlite3 to examine the artifacts stored by the popular Firefox brows*  July 26, 2018 | [134](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA107.w.0.0.0.0.13) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *More Information…Other Useful Skype Queries*  July 26, 2018 | [135](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA108) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *take the time to examine the Skype database further and make new scripts*  July 26, 2018 | [135](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA108) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *will examine what the Firefox application stores in a series*  July 26, 2018 | [135](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA108.w.3.0.86.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *databases in a default directory located at C:\Documents and Settings\\Application Data\Mozilla\Firefox\Profiles\\ under Windows and /Users//Library/Application\ Support/Firefox/Profiles/ under*  July 26, 2018 | [135](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA108.w.3.0.86.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Firefox stores quite a bit of forensically rich data*  July 26, 2018 | [135](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA108.w.11.1.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *start with the downloads.sqlite database*  July 26, 2018 | [135](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA108.w.11.1.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *contains a single table named moz\_downloads that stores information about the file*  July 26, 2018 | [136](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA109) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Excellent! We now know when a user downloaded specific files using Firefox*  July 27, 2018 | [136](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA109.w.4.0.22) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *investigator wants to log back onto sites that use authentication*  July 27, 2018 | [136](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA109.w.7.0.11) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *investigator (lawfully) would want to log back onto the web-based email, but most likely lacks the password or authentication to the user’s web-based email. Enter cookies*  July 27, 2018 | [136](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA109.w.7.0.11) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the HTTP protocol lacks a stateful design, origin Web sites utilize cookies*  July 27, 2018 | [136](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA109.w.7.0.73) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *a user logs onto a web-based email: if the browser could not maintain cookies, the user would have to log on in order to read every individual email*  July 27, 2018 | [136](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA109.w.7.0.73.0.1) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Firefox stores these cookies in a database named cookies.sqlite*  July 27, 2018 | [136](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA109.w.7.0.73.0.1) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Dealing With Encrypted Database ErrorUpdating*  July 27, 2018 | [137](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA110) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *open the cookies.sqlite database with the default Sqlite3 installation from Backtrack 5 R2, that it reports file is encrypted or is not a database*  July 27, 2018 | [137](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA110) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Recent versions of Firefox use the PRAGMA journal\_mode=WAL in their cookies.sqlite and places.sqlite databases*  July 27, 2018 | [137](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA110.w.2.0.38) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *libraries to a version > 3.7, you should be able to open the newer Firefox databases*  July 27, 2018 | [137](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA110.w.3.1.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *avoid our script crashing on this unhandled error, with the cookies.sqlite and places.sqlite databases, we put exceptions to catch the encrypted database error message*  July 27, 2018 | [137](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA110.w.6.0.1) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *write a quick Python script to extract cookies from a user under investigation*  July 27, 2018 | [137](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA110.w.6.0.46) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the database, the moz\_cookies maintains the data regarding cookies*  July 27, 2018 | [137](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA110.w.6.0.46.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we will query the column values for host, name, and cookie value, and print them to the screen*  July 27, 2018 | [138](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA111) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *enumerate the browser history. Firefox stores this data in a database named places.sqlite*  July 27, 2018 | [138](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA111.w.3.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the moz\_places table gives us valuable columns that include information about when (date) and where (address) a user visited a site*  July 27, 2018 | [138](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA111.w.6.0.16) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *our script for printHistory() only takes into account the moz\_places table*  July 27, 2018 | [138](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA111.w.6.0.16) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the ForensicWiki Web site recommends using data from both the moz\_places table and the moz\_historyvisits*  July 27, 2018 | [138](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA111.w.6.0.16) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *use the last example and our knowledge of regular expressions to expand the previous function*  July 27, 2018 | [139](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA112.w.0.0.0.0.14) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *history is infinitely valuable, it would be useful to look deeper into some of the specific URLs visited*  July 27, 2018 | [139](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA112.w.0.0.0.0.14) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Google search queries contain the search terms right inside of the URL, for*  July 27, 2018 | [139](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA112.w.2.0.27) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *If we spot a URL in our history that contains Google, we will search it for the characters q= followed by an &*  July 27, 2018 | [139](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA112.w.2.0.27) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *clean up the output by replacing some of the characters used in URLs to pad whitespace*  July 27, 2018 | [139](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA112.w.2.0.100) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *will print out the corrected output to the screen*  July 27, 2018 | [139](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA112.w.2.0.100) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we have a function that can search the places.sqlite file for and print out Google search queries*  July 27, 2018 | [139](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA112.w.2.0.100) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Wrapping it all together, we now have functions to print downloaded files, cookies, the history of a profile, and even print out the terms a user googled*  July 27, 2018 | [139](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA112.w.2.0.213) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *option parsing should look very similar to our script to investigate the Skype profile*  July 27, 2018 | [139](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA112.w.2.0.213) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *may notice the use of the function os.path.join when*  July 27, 2018 | [140](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA113) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *What prevents us from using an example such asdownloadDB = pathName + “\\downloads.sqlite” instead ofdownloadDB = os.path.join(pathName,“downloads.sqlite”)Consider this:*  July 27, 2018 | [140](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA113) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *. The slashes that indicate directories*  July 27, 2018 | [140](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA113.w.4.0.1) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *. The os library allows us to create an operating-system-independent script that will work on Windows, Linux and Mac*  July 27, 2018 | [140](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA113.w.4.0.1) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we have a complete working script to do some serious investigations into a Firefox profile*  July 27, 2018 | [140](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA113.w.4.0.66) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *try adding some addition functions to this script and modify it for your own investigations*  July 27, 2018 | [140](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA113.w.4.0.66) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *our script against a Firefox user profile under investigation, we see the results*  July 27, 2018 | [142](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA115.w.5.0.52.0.20) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *next section, we will use the skills learned in the two previous sections*  July 27, 2018 | [143](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA116) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *knowledge of SQLite by searching through a haystack of databases to find a needle*  July 27, 2018 | [143](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA116) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *April 2011, security researcher and former Apple employee Pete Warden disclosed a privacy issue with the popular Apple iPhone/Ipad iOS operating*  July 27, 2018 | [143](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA116.w.5.1.4.0.10) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *revealed proof that the Apple iOS operating system*  July 27, 2018 | [143](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA116.w.7.0.11) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *actually tracked and recorded the GPS coordinates of the device and stored them in a database on the phone called consolidated.db*  July 27, 2018 | [143](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA116.w.7.0.11) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *a table named CellLocation contained the GPS points the phone had collected. The device determined the location information by triangulating off the nearest cell-phone towers in order to provide the best service*  July 27, 2018 | [143](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA116.w.7.0.11) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *this same data could be used maliciously to track the entire movements*  July 27, 2018 | [143](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA116.w.7.0.72) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the process used to backup and store a copy of the mobile device to a computer also recorded this information*  July 27, 2018 | [143](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA116.w.7.0.72) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the location-recording information has been removed from the Apple iOS*  July 27, 2018 | [143](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA116.w.7.0.139) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the process Mr. Warden used to discover the data remains*  July 27, 2018 | [143](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA116.w.7.0.139) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *backup of his iPhone or iPad device, it stores files in a special directory on*  July 27, 2018 | [144](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA117) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the Windows operating*  July 27, 2018 | [144](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA117) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *that mobile device backup directory under the user’s profile directory at C:\Documents and Settings\\Application Data\AppleComputer\MobileSync\Backup*  July 27, 2018 | [144](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA117) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Mac OS X, this directory exists at /Users//Library/Application Support/MobileSync/Backup*  July 27, 2018 | [144](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA117) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the directory that stores our mobile directory backup, we see it contains over 1000 unhelpfully named files*  July 27, 2018 | [144](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA117.w.2.0.57) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *file contains a unique sequence of 40 characters that provide absolutely no description of the material*  July 27, 2018 | [144](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA117.w.2.0.57) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *get a little more information about each file, we will use the UNIX command file to extract the file type of*  July 27, 2018 | [144](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA117.w.4.0.10) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *command uses the first identifying bytes of a file header and footer*  July 27, 2018 | [144](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA117.w.4.0.10) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *provides us slightly more information, as we see that the mobile backup directory contains some sqlite3 databases, JPEG images, raw data, and ASCII text files*  July 27, 2018 | [144](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA117.w.4.0.10) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the file command does let us know that some of the files contain SQLite databases, it does very little to describe the content in each database*  July 27, 2018 | [144](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA117.w.6.0.18.0.10) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *use a Python script to quickly enumerate all the tables in each database found in the entire mobile backup directory*  July 27, 2018 | [144](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA117.w.6.0.18.0.10) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *script lists the contents of the working directory*  July 27, 2018 | [145](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA118) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *then attempts to make a database connection to each file*  July 27, 2018 | [145](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA118) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *in making a connection, the script executes the commandSELECT tbl\_name FROM sqlite\_master WHERE type==‘table’Each*  July 27, 2018 | [145](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA118) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *The previous command allows us to enumerate out the database schema*  July 27, 2018 | [145](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA118.w.2.0.97) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *our script, we enumerate the schema of all the databases in our mobile backup directory*  July 27, 2018 | [145](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA118.w.4.0.12) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *snipped the output to show a specific database of concern*  July 27, 2018 | [145](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA118.w.6.0.20) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *database contains a listing of the text messages stored in the iPhone backup*  July 27, 2018 | [145](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA118.w.6.0.20) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we want to be able to automate the investigation on different backups. To execute this, we write a simple function named isMessageTable()*  July 27, 2018 | [146](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA119.w.1.0.36.0.8) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *function will connect to a database and enumerate the information schema of the database*  July 27, 2018 | [146](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA119.w.1.0.36.0.8) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we can locate the text message database, we want to be able to print the data contained in the database—specifically the date, address, and text messages*  July 27, 2018 | [146](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA119.w.3.0.9) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *connect to the database and execute the command‘select datetime(date,\‘unixepoch\’), address, text from message WHERE address>0;’We can then print the results*  July 27, 2018 | [146](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA119.w.3.0.9) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *In the event that isMessageTable() returned a database that is not our actual text message database*  July 27, 2018 | [146](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA119.w.5.0.39.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *If we grabbed the wrong database by mistake, we will allow the script to catch the exception and continue executing*  July 27, 2018 | [146](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA119.w.5.0.39.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Packaging the functions isMessageTable() and printMessage() together*  July 27, 2018 | [147](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA120.w.2.0.101) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *construct the final script. We will add some option parsing to the script to include parsing the iPhone backup directory*  July 27, 2018 | [147](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA120.w.2.0.101) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we will list the contents of this directory and test each file until we find the text message database*  July 27, 2018 | [147](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA120.w.2.0.101) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Running the script against an iPhone backup directory, we can see the results against some recent text messages*  July 27, 2018 | [148](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA121.w.5.1.0.0.23) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *We have written quite a few tools in this chapter to investigate digital artifacts*  July 27, 2018 | [149](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA122.w.4.0.42.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *investigating the Windows Registry, the Recycle Bin, artifacts left inside metadata, or application-stored databases, we have added quite a*  July 27, 2018 | [149](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA122.w.4.0.42.0.4) | |

## *Chapter 4. Network Traffic Analysis with Python*

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Information in this chapter: Geo-Locate Internet Protocol (IP) Traffic Discover Malicious DDoS Toolkits*  July 27, 2018 | [152](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA125.w.0.0.0.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Uncover Decoy Network Scans Analyze Storm’s Fast-Flux and Conficker’s Domain Flux Understand the TCP Sequence Prediction Attack Foil Intrusion Detection Systems With Crafted Packets*  July 27, 2018 | [152](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA125.w.0.0.0.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *January 14, 2010, the United States learned of*  July 27, 2018 | [152](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA125.w.6.0.60.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *computer attack that targeted Google, Adobe and over 30 Fortune*  July 27, 2018 | [152](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA125.w.6.0.60.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Dubbed Operation Aurora after a folder found in an infected machine, the attack used a novel exploit unseen*  July 27, 2018 | [152](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA125.w.6.0.60.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Microsoft knew of the vulnerability exploited in the attack, they falsely assumed that nobody else knew of the vulnerability and therefore no mechanisms existed to detect*  July 27, 2018 | [152](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA125.w.8.0.51) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *exploit their victims, the attackers initiated the attack by sending the victims an email*  July 27, 2018 | [152](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA125.w.8.0.51.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *When users clicked on the link, they would download a piece of malware that connected back to a command-and-control server*  July 27, 2018 | [152](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA125.w.8.0.51.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the attackers used their newly gained access to hunt for proprietary information*  July 27, 2018 | [153](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA126) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *obvious as this attack appears, it went undetected for several months and succeeded*  July 27, 2018 | [153](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA126) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Even a rudimentary piece of network visualization software could have identified this behavior*  July 27, 2018 | [153](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA126.w.7.0.17) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *A visual map that showed users connecting to both Taiwan and China with significant frequency could have allowed network administrators to investigate*  July 27, 2018 | [153](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA126.w.7.0.17) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Let’s begin the investigation by building a script to visually analyze network traffic, something the administrators at the victimized Fortune 100 companies could have used during Operation Aurora*  July 27, 2018 | [153](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA126.w.7.0.80) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *begin with, we must how to correlate an Internet Protocol (IP) address to a physical location*  July 27, 2018 | [153](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA126.w.8.0.39.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we will rely on a freely available database from MaxMind*  July 27, 2018 | [153](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA126.w.8.0.39.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *open-source GeoLiteCity database available*  July 27, 2018 | [153](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA126.w.8.0.39.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the GeoCityLite database, we can correlate an IP address to a state, postal code, country name, and general latitude and longitude coordinates*  July 27, 2018 | [154](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA127.w.0.0.0.0.10) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Jennifer Ennis produced a pure Python library to query the GeoLiteCity*  July 27, 2018 | [154](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA127.w.5.0.30) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *that we will first instantiate a GeoIP class with the location of our uncompressed database*  July 27, 2018 | [154](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA127.w.5.0.30) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we will query the database for a specific record, specifying the IP address*  July 27, 2018 | [154](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA127.w.5.0.30) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Running the script, we see that it produces output showing the target IP’s physical location in Jersey City, NJ, US*  July 27, 2018 | [154](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA127.w.9.0.6) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we are able to correlate an IP to a physical address, let’s begin writing our analysis script*  July 27, 2018 | [154](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA127.w.12.0.21) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we will primarily use the Scapy packet manipulation toolkit analyze and craft packets*  July 27, 2018 | [154](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA127.w.12.0.21.0.12) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *will use a separate toolkit, dpkt, to analyze packet*  July 27, 2018 | [155](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA128) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Scapy offer*  July 27, 2018 | [155](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA128) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *tremendous capabilities, novice users often find the directions for installing it on Mac OS X and Windows extremel*  July 27, 2018 | [155](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA128) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *contrast, dpkt is fairly simple: it can be downloaded fro*  July 27, 2018 | [155](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA128) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *similar capabilities, but it always proves useful to keep an arsenal of similar tool*  July 27, 2018 | [155](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA128) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *After Dug Song initially created dpkt, Jon Oberheide added a lot of additional capabilities to parse different protocols, such as FTP, H.225, SCTP, BPG, and IPv6.*  July 27, 2018 | [155](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA128.w.4.0.73) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *assume we recorded a pcap network capture that we would like to analyze*  July 27, 2018 | [155](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA128.w.4.0.73) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Dpkt allows us to iterate through each individual packet in the capture and examine each protocol layer of the packet*  July 27, 2018 | [155](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA128.w.4.0.73) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we simply read a pre-captured PCAP in this example, we could just as easily analyze live traffic by using pypcap*  July 27, 2018 | [155](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA128.w.6.0.28) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *To read a pcap file, we instantiate the file, create a pcap.reader class object and then pass that object to our function*  July 27, 2018 | [155](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA128.w.6.0.28) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *printPcap(). The object pcap contains an array of records containing the [timestamp, packet]*  July 27, 2018 | [155](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA128.w.6.0.28) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *then break each packet down by into Ethernet and IP layers*  July 27, 2018 | [155](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA128.w.6.0.92) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *because we may capture layer-2 frames that do not contain the IP layer, it’s possible to throw an exception*  July 27, 2018 | [155](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA128.w.6.0.92) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we use exception handling to catch the exception and continue on to the next packet*  July 27, 2018 | [155](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA128.w.6.0.92) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *use the socket library to resolve IP addresses stored in inet notation*  July 27, 2018 | [155](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA128.w.6.0.92) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we print the*  July 27, 2018 | [155](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA128.w.6.0.163) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *source and destination to the screen for each individual packet*  July 27, 2018 | [155](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA128.w.6.0.163) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we see the source IP and destination IP address printed to the screen. While this provides us some level of analysis, let’s now correlate this to physical locations*  July 27, 2018 | [156](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA129.w.0.0.0.0.18) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Improving our script, let’s add an additional function called retGeoStr(), which returns a physical location for an IP address. For*  July 27, 2018 | [156](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA129.w.3.0.84) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we will simply resolve the city and three-digit country code and print these to the screen*  July 27, 2018 | [156](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA129.w.3.0.84) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the function raises an exception, we will return a message indicating the address is unregistered*  July 27, 2018 | [156](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA129.w.3.0.84) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *instances of addresses not in the GeoLiteCity database or private IP addresses, such as 192.168.1.3*  July 27, 2018 | [156](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA129.w.4.0.53) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Adding the retGeoStr function to our original script, we now have a pretty powerful packet analysis toolkit*  July 27, 2018 | [157](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA130) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we see several of our packets headed to Korea, London, Japan, and even Australia*  July 27, 2018 | [158](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA131) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Google Earth may prove a better way of visualizing this same information*  July 27, 2018 | [158](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA131) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Google Earth provides a virtual globe, map, and geographical information, shown on a proprietary viewer*  July 27, 2018 | [158](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA131.w.4.0.113) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Creating a text file with the extension KML allows a user to integrate various place marks into Google Earth. KML files contain a specific XML*  July 27, 2018 | [158](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA131.w.4.0.113) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we show how to plot two specific place marks on the map with a name and specific coordinates*  July 27, 2018 | [159](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA132) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *this should prove easy to integrate into our existing script to produce a KML file*  July 27, 2018 | [159](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA132) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *build a quick function, retKML(), that takes an IP as input and returns the specific KML structure*  July 27, 2018 | [159](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA132.w.2.1.3) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we are first resolving the IP address to a latitude and longitude using pygeoip*  July 27, 2018 | [159](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA132.w.3.0.9) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *then build our KML for a place mark. If we encounter an exception, such as “location not found,”*  July 27, 2018 | [159](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA132.w.3.0.9) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *our original script, we now also add the specific KML header*  July 27, 2018 | [159](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA132.w.3.0.73) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *For each packet, we produce KML place marks for the source IP and destination IP and plot them on our globe*  July 27, 2018 | [159](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA132.w.3.0.73) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *expanding this that could prove useful for an organization’s specific purpose. You may wish to use different icons for the types of traffic, specified by the source and destination TCP ports*  July 27, 2018 | [160](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA133) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *a .kml extension. Opening this file with Google Earth, we see a visual depiction our packet destinations*  July 27, 2018 | [161](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA134.w.8.0.0.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *December 2010, Dutch police arrested a teenager for participating in distributed denial-of-service*  July 28, 2018 | [161](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA134.w.8.0.0.0.6) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Visa, MasterCard, and PayPal as part of an operation to target companies opposed to WikiLeaks*  July 28, 2018 | [162](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA135) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the FBI issued forty search warrants and British police made five arrests as well. Loosely connected to the hacker group Anonymous*  July 28, 2018 | [162](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA135) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *alleged criminals downloaded and used the Low Orbit Ion Cannon (LOIC) distributed denial-of-service toolkit*  July 28, 2018 | [162](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA135) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *LOIC floods a target with large volumes of UDP and TCP traffic. A single instance of LOIC will do very little*  July 28, 2018 | [162](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA135.w.2.0.68) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *LOIC offers two modes of operation. In the first mode, a user can enter a target address. In the second mode, dubbed HIVEMIND, the user connects LOIC to an IRC server where users can nominate targets that the IRC-connected users will automatically attack*  July 28, 2018 | [162](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA135.w.2.0.68) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *their toolkit, LOIC. The Frequently Asked Questions (FAQ) states: “Will I get caught/arrested for using it? Chances are next to zero. Just blame you have a virus, or simply deny any knowledge of it.” In this section, let’s debunk that*  July 28, 2018 | [162](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA135.w.4.0.9) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *some are more credible than others. As sourceforge hosts a copy at http://sourceforge.net/projects/loic/, let’s download a copy from there. Before downloading, open up a tcpdump session, filter on port 80, and print the results in ASCII format.*  July 28, 2018 | [162](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA135.w.7.0.9) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the first part of our LOIC discovery toolkit, we will write a Python script to parse HTTP traffic and examine it for HTTP GETs for the zipped LOIC*  July 28, 2018 | [162](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA135.w.8.2.3.0.6) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we will again use Dug Song’s dpkt library. To examine the HTTP traffic, we must extract the Ethernet, IP,*  July 28, 2018 | [162](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA135.w.8.2.3.0.6) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *and TCP layers. Finally, the HTTP protocol rides on top of the TCP protocol layer.*  July 28, 2018 | [163](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA136) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *layer utilizes the GET method, we parse out the specific uniform resource identifier (URI) that the HTTP GET request*  July 28, 2018 | [163](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA136) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *this URI contains .zip and LOIC in the name, we print a message to the screen with the IP that downloaded LOIC.*  July 28, 2018 | [163](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA136) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *a clever administrator prove that a user downloaded LOIC as opposed to being infected by a virus*  July 28, 2018 | [163](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA136.w.3.0.114) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Simply downloading LOIC is not necessarily illegal (or the author of this book might be in some trouble); however, connecting to the Anonymous HIVE and launching*  July 28, 2018 | [163](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA136.w.8.0.3.0.12) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *does violate several state, federal, and national laws.*  July 28, 2018 | [164](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA137) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Any member of Anonymous connected to the IRC with LOIC connected in HIVEMIND mode can immediately start an attack against the target*  July 28, 2018 | [164](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA137.w.3.0.96) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the IP address 66.211.169.66 refers to the address of paypal.com, targeted during Operation Payback.*  July 28, 2018 | [164](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA137.w.3.0.96) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the specific attack message traffic in tcpdump, we see that a specific user—anonOps—issued a command to start an attack*  July 28, 2018 | [164](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA137.w.3.0.96) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *While this proves easily looking at these two specific packets, imagine trying to find this in a lengthy PCAP file containing hours or days of network traffic*  July 28, 2018 | [164](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA137.w.4.0.12) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *most cases, the IRC server uses TCP port 6667. Messages headed to the IRC server will have the destination TCP port 6667*  July 28, 2018 | [164](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA137.w.7.0.12) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *use this knowledge when we write our HIVEMIND parsing function, findHivemind(). This time, we will extract the Ethernet, IP, and TCP layers.*  July 28, 2018 | [164](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA137.w.9.0.26) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we examine it for the specific source and destination ports.*  July 28, 2018 | [164](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA137.w.9.0.26.0.1) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *With functions to locate a user downloading LOIC and to find the hive commands, one last mission remains: identifying the attack*  July 28, 2018 | [165](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA138.w.3.0.9) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *starts a LOIC attack, it fires a massive amount of TCP packets towards a target. These packets, combined with the collective packets*  July 28, 2018 | [165](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA138.w.6.0.3) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *We start a tcpdump session and see several small (length 12) TCP packets sent every 0.00005 seconds.*  July 28, 2018 | [165](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA138.w.6.0.3) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *quickly write a function that finds a DDoS attack in progress. To detect an attack, we will set a threshold of packets*  July 28, 2018 | [166](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA139.w.12.0.0.0.12) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the number of packets from a user to a specific address exceeds this threshold*  July 28, 2018 | [166](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA139.w.18.0.11) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we might want to investigate further as an attack*  July 28, 2018 | [166](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA139.w.18.0.11) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *this does not definitively prove a user has initiated an attack*  July 28, 2018 | [166](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA139.w.18.0.11) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *correlating this to a user downloading LOIC*  July 28, 2018 | [166](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA139.w.18.0.11) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *followed by acceptance of a HIVE command, followed by the actual attack, does provide overwhelming evidence to prove a user participated in an Anonymous-sponsored DDoS attack. import*  July 28, 2018 | [166](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA139.w.18.0.11) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *adding some option parsing, our script now detects the download, overhears the HIVE commands*  July 28, 2018 | [167](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA140.w.7.0.31.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Running the code, we see the results. Four users downloaded the toolkit. Next, a different user issued the attack command to two other connected attackers*  July 28, 2018 | [169](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA142.w.1.0.84.0.15) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *these two attackers actually participated in the attack. Thus, the script now identifies an entire DDoS in action. While*  July 28, 2018 | [169](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA142.w.1.0.84.0.15) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *In the following section, we will look at a custom script that a seventeen-year-old wrote to defend the Pentagon*  July 28, 2018 | [169](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA142.w.3.0.40.0.1) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *late 1999, the US Pentagon faced a serious crisis against its computer networks. The headquarters of the US Department of Defense*  July 28, 2018 | [170](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA143.w.2.0.57.0.8) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *announced it was under a coordinated series of sophisticated attacks*  July 28, 2018 | [170](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA143.w.4.0.10) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *newly released tool, Nmap, made it rather easy for anyone to scan networks for services and vulnerabilities. The*  July 28, 2018 | [170](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA143.w.4.0.10) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Nmap scan proves rather easy to detect, correlate to the attacker’s address and then geo-locate that IP address*  July 28, 2018 | [170](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA143.w.4.0.72) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Instead of just sending scans from their specific attacker addresses, they included decoy scans that appeared to originate from many places around the world*  July 28, 2018 | [170](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA143.w.4.0.72) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *seventeen-year-old from Austin, TX finally presented a working solution. H.D. Moore, legendary creator of the attack framework Metasploit, met with Stephen Northcutt from the NAVY Shadow project*  July 28, 2018 | [170](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA143.w.5.0.53) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *suggested using the TTL fields for all incoming packets from Nmap scans (Verton, 2002). The time-to-live (TTL) field of an IP packet determines*  July 28, 2018 | [171](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA144) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *how many hops*  July 28, 2018 | [171](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA144) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *take before reaching its destination. Every time a packet crosses a routing device, the router decrements the TTL field*  July 28, 2018 | [171](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA144) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *For each source address used in the logged Nmap scans, he sent a single ICMP packet to determine the number of hops between the source address and the scanned machine*  July 28, 2018 | [171](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA144) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *then used this information to distinguish the attacker from the decoys*  July 28, 2018 | [171](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA144.w.1.0.65) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Moore to present his toolkit and research at a SANS conference in 1999 (Verton, 2002). Moore dubbed his tool Nlog because it logged various bits of information from Nmap scans*  July 28, 2018 | [171](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA144.w.1.0.131) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we will use Python to recreate Moore’s analysis and construction of the Nlog toolkit*  July 28, 2018 | [171](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA144.w.1.0.131) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *explain the TTL field of an IP packet. The TTL field contains 8 bits, making valid values 0 through 255*  July 28, 2018 | [171](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA144.w.2.0.32) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *a computer sends an IP packet, it sets the TTL field as the upper bound of hops a packet can take before reaching a destination*  July 28, 2018 | [171](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA144.w.2.0.32) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *If the field reaches zero, then the router discards the packet to prevent infinite routing loops*  July 28, 2018 | [171](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA144.w.4.0.57) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *if I ping the address 8.8.8.8 with an initial TTL of 64 and it returns with a TTL of 53, I see the packet crossed 11 routing devices*  July 28, 2018 | [171](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA144.w.4.0.57) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Nmap initially introduced decoy scans in version 1.60, the TTL was neither randomized nor calculated correctly for the decoy packets*  July 28, 2018 | [171](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA144.w.5.0.17) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Failing to correctly calculate the TTL allowed Moore to identify these packets*  July 28, 2018 | [171](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA144.w.5.0.17) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *randomizes the TTL using the following algorithm. This algorithm produces a random TTL, averaging about 48 per packet*  July 28, 2018 | [171](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA144.w.6.0.43) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *user can also hard-code the TTL*  July 28, 2018 | [171](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA144.w.6.0.43.0.1) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *run an Nmap decoy scan, we will utilize the –D flag*  July 28, 2018 | [171](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA144.w.6.0.43.0.1) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we will use the address 8.8.8.8 as a decoy address. Furthermore, we will hard code a TTL of 13 using the –ttl*  July 28, 2018 | [172](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA145) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *On the target, 192.168.1.7, we fire up tcpdump in verbose mode (-v), disable name resolution (-nn) and filter on the specific address 8.8.8.8 (‘host 8.8.8.8’). We*  July 28, 2018 | [172](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA145.w.5.0.12) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *begin writing our script by printing out the source IP address and TTL of incoming packets*  July 28, 2018 | [172](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA145.w.7.7.0.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we will return to using Scapy for the rest of the chapter*  July 28, 2018 | [172](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA145.w.7.7.0.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *We will set up a function to sniff and pass each individual packet to the function testTTL(), which*  July 28, 2018 | [172](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA145.w.7.7.0.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *examines the packet for the IP layer, extracting the IP source address and TTL fields*  July 28, 2018 | [173](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA146) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we see that we have received quite a few packets from different source addresses, with varying TTLs. These*  July 28, 2018 | [173](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA146.w.6.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *also include the decoy scans from 8.8.8.8 with a TTL of 13. As we know that the TTL should be*  July 28, 2018 | [173](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA146.w.6.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *53 hops away, we can argue that somebody spoofed these*  July 28, 2018 | [173](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA146.w.6.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *important to note at this point that while Linux/Unix systems usually start with an initial TTL of 64, Windows-based systems start with a TTL of 128*  July 28, 2018 | [173](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA146.w.6.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *our script here, we’ll assume we are only dissecting IP packets from Linux workstations*  July 28, 2018 | [173](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA146.w.11.0.58) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *function checkTTL() takes an IP source address with its respective received TTL as input and prints out a message for invalid TTLs*  July 28, 2018 | [173](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA146.w.15.1.3.0.14) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *use a quick conditional statement to eliminate packets from private IP addresses (10.0.0.0–10.255.255.255, 172.16.0.0–172.31.255.255, and 192.168.0.0–192.168.255.255)*  July 28, 2018 | [174](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA147) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we import the IPy library. To avoid the class IP conflicting with the Scapy class IP, we reclassify it as IPTEST*  July 28, 2018 | [174](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA147) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *IPTEST(ipsrc).iptype() returns ‘PRIVATE’, we return from our checkTTL function*  July 28, 2018 | [174](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA147) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *only want to check the source address once. If we have not seen the source address previously, let’s build an IP packet with a destination address equal to the source*  July 28, 2018 | [174](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA147.w.1.0.70) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we make the packet an ICMP echo request so that the destination will respond. Once the destination address responds, we place the TTL value in a dictionary, indexed by the IP source address*  July 29, 2018 | [174](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA147.w.1.0.70) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *check to see if that difference between the actual received TTL and the TTL on the original packet exceeds a threshold*  July 29, 2018 | [174](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA147.w.2.0.55) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *however, if thehops distance differs byfive hops, we can assume it may be a*  July 29, 2018 | [174](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA147.w.2.0.55) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *add some option parsing for the specific address to listen in on, followed by an option to set the threshold*  July 29, 2018 | [174](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA147.w.5.0.12) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Running our code, we can see that it correctly identifies the decoy Nmap scan from 8.8.8.8 because of the TTL of 13 compared to the actual TTL of*  July 29, 2018 | [175](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA148.w.3.0.0.0.17) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *It is important to note that our value is generated off an initial default TTL for Linux of 64. Although RFC 1700 recommends the default TTL as 64*  July 29, 2018 | [175](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA148.w.3.0.0.0.17) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *some other Unix variants have different TTLs such as Solaris 2.x with a default TTL of 255*  July 29, 2018 | [176](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA149) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *2007, security researchers identified a new technique used by the infamous Storm botnet*  July 29, 2018 | [176](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA149.w.5.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *technique, named fast-flux, used domain name service (DNS) records to hide the command and control servers*  July 29, 2018 | [176](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA149.w.5.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *DNS records typically translate a domain name to an IP address. When a DNS server returns a result*  July 29, 2018 | [176](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA149.w.5.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the TTL that the IP address remains valid for before the host should check again*  July 29, 2018 | [176](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA149.w.6.0.55) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *attackers behind the Storm botnet changed the DNS records for the command-and-control server rather frequently*  July 29, 2018 | [176](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA149.w.6.0.55) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *they used 2,000 redundant hosts spread amongst 384 providers in more than 50 countries*  July 29, 2018 | [176](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA149.w.6.0.55) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *attackers swapped the IP addresses for the command-and-control server frequently and ensured the DNS results returned with a very short TTL*  July 29, 2018 | [176](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA149.w.6.0.55.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *fast-flux of IP addresses made*  July 29, 2018 | [177](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA150) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *difficult for security researchers to identify the command-and-control servers*  July 29, 2018 | [177](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA150) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *fast-flux proved difficult in the takedown of the Storm botnet, a similar techniq*  July 29, 2018 | [177](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA150) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *following year aided in the infection of seven million computers in over two hundred countries*  July 29, 2018 | [177](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA150) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Conficker, the most successful computer worm to date*  July 29, 2018 | [177](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA150.w.2.0.23) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *by attacking a vulnerability in the Windows Service Message Block (SMB) protocol*  July 29, 2018 | [177](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA150.w.2.0.23) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the vulnerable machines contacted a command-and-control server for further instructions*  July 29, 2018 | [177](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA150.w.2.0.23) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *However, Conficker generated different domain names every three hours, using the current date and time at UTC*  July 29, 2018 | [177](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA150.w.2.0.77) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the third iteration of Conficker, this meant 50,000 domains were generated every three hours*  July 29, 2018 | [177](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA150.w.2.0.77) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Attackers registered only a handful of these domains to*  July 29, 2018 | [177](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA150.w.2.0.77) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *actual IP addresses for the command-and-control servers*  July 29, 2018 | [177](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA150.w.2.0.77) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Because the technique rotated domain names, researchers named it domain-flux*  July 30, 2018 | [177](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA150.w.2.0.136) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we will write some Python scripts to detect fast-flux and domain-flux in the wild to identify attacks.*  July 30, 2018 | [177](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA150.w.2.0.136) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *let’s quickly review the DNS by looking at the traffic generated during a domain name request.*  July 30, 2018 | [177](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA150.w.2.0.136) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *let’s perform a domain-name lookup on the address whitehouse.com. Note that our DNS server at 192.168.1.1, translates whitehouse.com into the IP address 74.117.114.119.*  July 30, 2018 | [177](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA150.w.5.0.8) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Examining a DNS lookup with tcpdump, we see that our client (192.168.13.37) sends a request to the DNS server at 192.168.1.1.*  July 30, 2018 | [177](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA150.w.7.0.6) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the client generates a DNS Question Record (DNSQR) asking for the IPv4 address of whitehouse.com. The*  July 30, 2018 | [177](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA150.w.7.0.6) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *server responds by appending a DNS Resource Record (DNSRR) that provides the IP address*  July 30, 2018 | [177](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA150.w.7.0.6.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *When we examine these DNS protocol requests in Scapy, we see the fields included*  July 30, 2018 | [178](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA151) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *DNSQR contains the question name (qname), the question type*  July 30, 2018 | [178](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA151) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *and question class (qclass*  July 30, 2018 | [178](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA151.w.5.0.22) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we ask for the IPv4 address for whitehouse.com to be resolved, making the qname field equal to whitehouse.com*  July 30, 2018 | [178](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA151.w.5.0.22) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *The DNS server responds by appending a DNSRR that contains the resource record name (rrname), the type (type), resource record class (rclass), and TTL*  July 30, 2018 | [178](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA151.w.5.0.22) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *The European Network and Information Security Agency provides*  July 30, 2018 | [178](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA151.w.7.1.22) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *resource for analyzing network traffic. They*  July 30, 2018 | [178](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA151.w.7.1.22) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *a live DVD ISO image that contains several network captures and exercises. You can download a copy from http://www.enisa.europa.eu/activities/cert/support/exercise/live-dvd-iso-images*  July 30, 2018 | [178](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA151.w.7.1.22) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Exercise #7 provides a Pcap that demonstrates fast-flux behavior. Additionally*  July 30, 2018 | [178](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA151.w.7.1.22) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *you may wish to infect a virtual machine with spyware or malware and examine the traffic safely*  July 30, 2018 | [178](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA151.w.8.0.44) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *assume you now have a network captured named fastFlux.pcap that contains some DNS traffic you would like to analyze*  July 30, 2018 | [178](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA151.w.8.0.44.0.1) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *write a Python script that reads in this pcap and that parses out all the packets that contain DNSRR*  July 30, 2018 | [178](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA151.w.8.0.44.0.1) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Scapy contains a powerful function,. haslayer(),which takes a protocol type as input and returns a Boolean*  July 30, 2018 | [179](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA152) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the packet contains a DNSRR, we will extract*  July 30, 2018 | [179](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA152) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *rrname and rdata variables that contain the appropriate domain name and IP address*  July 30, 2018 | [179](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA152) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *check the domain name against a dictionary we maintain, indexed by the domain names*  July 30, 2018 | [179](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA152) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we have seen the domain name before, we will check to see if it had a previous IP address associated*  July 30, 2018 | [179](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA152.w.3.0.72) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *a different previous IP address, we add our new address to the array maintained in the value of our dictionary*  July 30, 2018 | [179](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA152.w.3.0.72) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *if we identify a new domain, we add it to our dictionary. We add the IP address for the domain*  July 30, 2018 | [179](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA152.w.3.0.72) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *does seem a little complex, but we want to be able to store all the domain names and the various IP addresses associated with them*  July 30, 2018 | [179](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA152.w.3.0.150) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *To detect fast flux, we will need to know which domain names have multiple addresses*  July 30, 2018 | [179](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA152.w.3.0.150) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we examine all the packets, we print out all the domain names and how many unique IP addresses exist*  July 30, 2018 | [179](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA152.w.3.0.150) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Running our code, we see that at least four domain names have a multitude of IP addresses associated with them*  July 30, 2018 | [179](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA152.w.4.0.101.0.14) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *let’s begin by analyzing a machine infected with Conficker*  July 30, 2018 | [180](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA153) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Many third-party sites contain various Conficker network captures. As Conficker utilized domain-flux,*  July 30, 2018 | [180](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA153.w.7.0.24) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *need to look at the server responses that contain error messages for unknown domain names*  July 30, 2018 | [180](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA153.w.7.0.24) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Different versions of Conficker generated several DNS names hourly*  July 30, 2018 | [180](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA153.w.7.0.24) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *several of the domain names proved bogus and were meant to mask the actual command-and-control server, most DNS servers lacked the ability to translate the domain*  July 30, 2018 | [180](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA153.w.7.0.24) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *and instead generated error messages*  July 30, 2018 | [180](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA153.w.7.0.82) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *identify domain-flux in action by identifying all the DNS responses that contain an error code for name-error. For*  July 30, 2018 | [180](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA153.w.7.0.82) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *full listing of the domains used in the Conficker Worm, see http://www.cert.at/downloads/data/conficker\_en.html*  July 30, 2018 | [180](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA153.w.7.0.82) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *read in a network capture and enumerate through all the packets in the capture. We will test only packets originating from the server source port 53 that contain resource records.*  July 30, 2018 | [180](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA153.w.7.0.82) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *DNS packet contains anrcode field. When the rcode equals 3, it indicates that the domain name does not exist*  July 30, 2018 | [180](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA153.w.8.0.9) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *print the domain name to the screen and update a hit counter*  July 30, 2018 | [180](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA153.w.8.0.9) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *when we run our script, we see several of the actual domain names used in Conficker for domain-flux*  July 30, 2018 | [181](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA154.w.0.0.0.0.6) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *February 16, 1995 ended the reign of a notorious hacker, whose crime spree included the theft of corporate trade secrets worth millions of dollars. For over 15 years, Kevin Mitnick gained unauthorized access to computers, stole proprietary information, and harassed anyone who tried to catch him*  July 30, 2018 | [181](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA154.w.4.0.51.0.8) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Tsutomu Shimomura, a long-haired computational physicist from San Diego, aided in capturing Mitnick (Markoff, 1995). After testifying before Congress about cellular phone security in 1992, Shimomura became a target for Mitnick*  July 30, 2018 | [181](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA154.w.6.0.38.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *December 1994, someone broke into Shimomura’s home computer system*  July 30, 2018 | [181](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA154.w.6.0.38.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Convinced the attacker was Mitnick and fascinated by the novel attack method*  July 30, 2018 | [182](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA155) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Shimomura essentially led the technical team that tracked Mitnick*  July 30, 2018 | [182](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA155) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Never seen before in the wild, Mitnick used a method of hijacking TCP sessions*  July 30, 2018 | [182](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA155) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *known as TCP sequence prediction, exploited the lack of randomness in the sequence numbers used to track individual network connections*  July 30, 2018 | [182](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA155) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *combined with IP address spoofing, allowed Mitnick to hijack a connection to Shimomura’s home computer*  July 30, 2018 | [182](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA155.w.2.0.27) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *machine that Mitnick attacked had a trusted agreement with a remote server. The remote server could access Mitnick’s victim via the remote login (rlogin) protocol that runs on TCP port 513*  July 30, 2018 | [182](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA155.w.3.0.5) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *rlogin used an insecure means of authentication—by checking the source IP address*  July 30, 2018 | [182](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA155.w.3.0.5) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *to attack Shimomura’s machine, Mitnick had to 1) find a server that it trusted; 2) silence that trusted server; 3) spoof a connection from that server; and 4) blindly spoof a proper acknowledgement of the TCP three-way handshake*  July 30, 2018 | [182](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA155.w.3.0.5) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *On January 25, 1994,Shimomura posted details about the attack to a USENET blog (Shimomura, 1994). Analyzing*  July 30, 2018 | [182](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA155.w.4.0.63) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Mitnick identified a remote server that had a trusted agreement with Shimomura’s personal machine, he needed to silence that machine. If the machine noticed the spoofed connection attempt using its IP address, it would then send TCP reset packets to close the connection*  July 30, 2018 | [182](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA155.w.4.0.129) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *silence the machine, Mitnick sent a series of TCP SYN packets to the rlogin port on the server*  July 30, 2018 | [182](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA155.w.4.0.129) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *SYN Flood, this attack filled up the connection queue of the server and kept it from responding*  July 30, 2018 | [182](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA155.w.5.0.53) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Replicating a TCP SYN flood attack in Scapy proves simple*  July 30, 2018 | [183](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA156.w.0.0.0.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *will craft some IP packets with a TCP protocol layer with an incrementing TCP source port and constant TCP destination port of 513*  July 30, 2018 | [183](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA156.w.0.0.0.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Running the attack sends TCP SYNs to exhaust the resources of the target, filling up its connection queue*  July 30, 2018 | [183](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA156.w.7.0.53) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *silencing the target’s ability to send TCP-reset packets*  July 30, 2018 | [183](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA156.w.7.0.53) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the attack gets a little more interesting. With the remote server silenced, Mitnick could spoof a TCP connection*  July 30, 2018 | [183](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA156.w.12.0.0.0.12) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *this depended upon his ability to send a spoofed SYN, followed by Shimomura’s machine acknowledging the TCP connection with a TCP SYN-ACK*  July 30, 2018 | [183](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA156.w.12.0.0.0.12) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Mitnick needed to correctly guess the TCP sequence number in the SYN-ACK*  July 30, 2018 | [183](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA156.w.12.0.0.0.12) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *then send back an ACK of that correctly guessed TCP sequence number. To calculate the correct TCP sequence number, Mitnick sent a series of SYNs from*  July 30, 2018 | [184](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA157) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *After receiving the SYN, Shimomura’s machine x-terminal responded with an SYN-ACK with a TCP sequence number. Notice the sequence numbers*  July 30, 2018 | [184](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA157) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *technical details: 2022080000, 2022208000, 2022336000, 2022464000. Each incrementing SYN-ACK differs by 128,000 digits. This made calculating the correct TCP sequence number rather easy*  July 30, 2018 | [184](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA157.w.5.0.115) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *in Python, we will send a TCP SYN and wait for a TCP SYN-ACK. Once*  July 30, 2018 | [184](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA157.w.6.8.3) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we will strip off the TCP sequence number from the acknowledgement and print it to the screen. We*  July 30, 2018 | [185](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA158) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *for four packets to confirm that a pattern exists*  July 30, 2018 | [185](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA158) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Scapy, we don’t need to complete all the TCP and IP fields: Scapy will fill them in with values*  July 30, 2018 | [185](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA158) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Our new function calTSN will take a target IP address and return the next sequence number to be acknowledged (the current sequence number plus the difference)*  July 30, 2018 | [185](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA158) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Running our code against a vulnerable target, we see that TCP sequence randomization does not exist*  July 30, 2018 | [185](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA158.w.12.0.3) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Note, that by default Scapy will use default destination TCP Port 80. The destination target must have a service listening on whatever port you attempt to spoof*  July 30, 2018 | [185](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA158.w.17.0.27) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the correct TCP sequence number in hand, Mitnick was able to attack*  July 30, 2018 | [186](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA159.w.0.0.0.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *First he spoofed a connection from the now-silent server. Next he sent a blind ACK with the sequence number of 2024371201, indicating that the connection was established correctly*  July 30, 2018 | [186](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA159.w.0.0.0.0.4) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *First we create a SYN with a TCP source port of 513 and destination of 514 with the IP source address of the spoofed server and the destination IP address as the target*  July 30, 2018 | [186](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA159.w.4.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Next, we create a similar acknowledgement packet, add the calculated sequence number as an additional field, and send*  July 30, 2018 | [186](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA159.w.6.0.45) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the entire codebase back together, we’ll add option parsing to add command line options for the spoofed address for the connection, the target server*  July 30, 2018 | [186](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA159.w.7.1.28) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *and the spoofed address for the initial SYN flood*  July 30, 2018 | [186](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA159.w.7.1.28) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *our final script, we have successfully replicated Mitnick’s almost two-decade-old attack. What was once thought of as one of the most sophisticated attacks in history can now be replicated with exactly 65 lines*  July 30, 2018 | [188](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA161.w.5.1.14) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *let’s use the next section to describe a method*  July 30, 2018 | [188](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA161.w.9.0.27) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *for complicating analysis of network attacks, specifically targeting intrusion detection systems*  July 30, 2018 | [188](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA161.w.9.0.27) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Intrusion Detection System (IDS) is a very valuable tool in the hands of a competent analyst*  July 30, 2018 | [188](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA161.w.15.0.0.0.32) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *network-based intrusion detection system (NIDS) can analyze traffic in real time by logging packets on IP networks*  July 30, 2018 | [189](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA162) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *matching packets against a known set of malicious signatures, an IDS can alert the network analyst to an attack before it succeeds*  July 30, 2018 | [189](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA162) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the SNORT IDS system comes pre-packaged with a variety of different rules capable of detecting different types of reconnaissance, exploits, and denial*  July 30, 2018 | [189](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA162) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *denial-of-service attacks amongst*  July 30, 2018 | [189](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA162.w.8.0.71) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Examining the contents of one of these rule configurations, we see four alerts to detect the TFN, tfn2k, and Trin00 distributed denial-of-service attack toolkits. When*  July 30, 2018 | [189](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA162.w.8.0.71) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *what happens when analysts receive more alerts than they can reasonably correlate to an event*  July 30, 2018 | [189](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA162.w.8.0.133) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *order to hide a legitimate attack from the analyst, we will write a toolkit that generates an overwhelming number of alerts*  July 30, 2018 | [189](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA162.w.9.5.16.0.6) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *an analyst could use this tool to verify an IDS can correctly identify malicious traffic. Writing the script will not prove difficult*  July 30, 2018 | [189](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA162.w.9.5.16.0.6) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the first rule for the DDOS TFN Probe:here we must generate an ICMP packet with an ICMP ID of 678 and ICMP TYPE 8 that contains the raw contents “1234*  July 30, 2018 | [190](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA163) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *With Scapy, we craft a packet with these variables and send it to our destination*  July 30, 2018 | [190](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA163.w.1.0.104) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Running the script, we see that four packets were sent to our destination. The IDS will analyze these packets and generate alerts if they match the signatures correctly*  July 30, 2018 | [190](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA163.w.2.1.12) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Examining the alert log for SNORT, we find that we have succeeded*  July 30, 2018 | [190](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA163.w.4.0.14.0.18) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Let’s look at some more slightly complicated rules in the exploit.rules signature file for SNORT*  July 30, 2018 | [191](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA164.w.5.0.14) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *a sequence of specific bytes will generate alerts for the ntalkd x86 Linux overflow and the Linux mountd overflow*  July 30, 2018 | [191](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA164.w.5.0.14) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *To generate packets containing the raw bytes, we will use the notation \x followed by the hexadecimal encoding of the byte*  July 30, 2018 | [191](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA164.w.8.1.16) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the first alert, this generates a packet that will trip the signature for the ntalkd Linux overflow exploit*  July 30, 2018 | [191](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA164.w.8.1.16) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the second packet, we will use a combination of raw bytes encoded as hex plus standard ASCII characters. Notice how 89|F| encodes as \x89F to indicate it contains raw bytes*  July 30, 2018 | [191](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA164.w.8.1.16) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *The following packets will generate alerts for exploit attempts*  July 30, 2018 | [191](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA164.w.10.0.43) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Finally, it would be nice to spoof some reconnaissance or scans. We examine the SNORT rules for scans and find two rules that we can craft packets for. Both*  July 30, 2018 | [191](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA164.w.11.0.25.0.10) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *detect malicious behavior on the UDP protocol on specific ports with specific raw content*  July 30, 2018 | [191](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA164.w.11.0.25.0.10) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *We generate the two packets for the scan rules for cybercop and Amanda reconnaissance tools*  July 30, 2018 | [192](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA165.w.3.0.15) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we have packets to generate alerts for denial-of-service attacks, exploits, and reconnaissance*  July 30, 2018 | [192](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA165.w.5.0.19) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *put our script back together and add some option parsing*  July 30, 2018 | [192](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA165.w.5.0.19) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the user must enter the target address or the program will exit: if the user fails to enter a source address, we will generate a random source address*  July 30, 2018 | [192](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA165.w.5.0.19) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *If the user does not specify how many times to send the crafted packets, we will only send them once*  July 30, 2018 | [192](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA165.w.5.0.19) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *script uses the default adapter eth0 unless otherwise specified*  July 30, 2018 | [192](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA165.w.7.4.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *you could continue to add to this script to generate and test alerts*  July 30, 2018 | [192](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA165.w.7.4.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we see it correctly sends eight packets to the target address and spoofs the source address as 1.3.3.7. For testing purposes, ensure the target is different than the attacker machine*  July 30, 2018 | [194](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA167) | |

## *Chapter 5. Wireless Mayhem with Python*

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Sniffing Wireless Networks for Personal Information Listening for Preferred Networks and Identifying Hidden Wireless Networks*  July 30, 2018 | [197](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA170.w.0.0.0.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Taking Control of Wireless Unmanned Aerial Vehicles Identifying Firesheep in Use Stalking Bluetooth Radios Exploiting Bluetooth Vulnerabilities*  July 30, 2018 | [197](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA170.w.0.0.0.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *September 5, 2007, the US Secret Service arrested a wireless hacker named Max Ray Butler*  July 30, 2018 | [198](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA171.w.0.0.0.0.8) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *as the Iceman, Mr. Butler sold tens of thousands of credit card accounts through a Website.*  July 30, 2018 | [198](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA171.w.8.0.3) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *how did he collect this private information? Sniffing unencrypted wireless Internet connection*  July 30, 2018 | [198](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA171.w.8.0.3) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *one of the methods he used to gain access to credit card information. The Iceman rented hotel rooms and apartments using false identities. He then used high-power antennae to intercept communications to the hotel’s and nearby apartments’ wireless access points to capture the personal information*  July 31, 2018 | [198](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA171.w.8.0.3) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *media experts classify this type of attack “sophisticated and complex.” Such a statement proves dangerous, as we can execute several of these attacks in short Python scripts*  July 31, 2018 | [198](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA171.w.8.0.70.0.1) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we can sniff for credit card information in less than 25 lines of code*  July 31, 2018 | [199](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA172) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *let’s ensure we have our environment setup correctly*  July 31, 2018 | [199](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA172) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *In the following sections, we will write code to sniff wireless traffic and send raw 802.11 frames*  July 31, 2018 | [199](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA172.w.0.0.0.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *We will use a Hawking Hi-Gain USB Wireless-150N Network Adapter with Range Amplifier*  July 31, 2018 | [199](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA172.w.0.0.0.0.2) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *to create and test the scripts in this chapter. The default drivers for this card on Backtrack 5 allow a user to place it into monitor mode as well as transmit raw frames*  July 31, 2018 | [199](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA172.w.8.0.21) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *it contains an external antenna connection that allows us to attach a high-powered antenna to the card*  July 31, 2018 | [199](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA172.w.8.0.21) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *scripts require the ability to place the card into a monitor in order to passively listen for all wireless traffic*  July 31, 2018 | [199](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA172.w.8.0.21) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Monitor mode allows you to receive raw wireless frames rather than 802.11 Ethernet frames*  July 31, 2018 | [199](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA172.w.9.0.5) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *allows you to see beacons and the wireless management frames even if you are not associated*  July 31, 2018 | [199](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA172.w.9.0.5) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *place the card into monitor mode, we use the aircrack-ng suite of tools written by Thomas d’Otreppe*  July 31, 2018 | [199](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA172.w.9.0.5) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Iwconfig lists our wireless adapter as wlan0*  July 31, 2018 | [199](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA172.w.11.0.6) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *run the command airmon-ng start wlan0 to start it into monitor mode*  July 31, 2018 | [199](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA172.w.11.0.6) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *This creates a new adapter known as mon0*  July 31, 2018 | [199](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA172.w.11.0.6) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *quickly test that we can capture wireless traffic after placing the card into monitor mode*  July 31, 2018 | [199](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA172.w.15.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we set our conf.iface to the newly created monitoring interface*  July 31, 2018 | [199](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA172.w.15.0.0) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Upon hearing each packet, the script runs the procedure pktPrint()*  July 31, 2018 | [199](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA172.w.15.0.0.0.12) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *prints a message if the packet contains an 802.11 Beacon, an 802.11 Probe Response, a TCP Packet, or DNS traffic*  July 31, 2018 | [200](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA173) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *firing up the script we see quite a bit of traffic*  July 31, 2018 | [200](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA173.w.1.0.58) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the traffic includes the 802.11 Probe Requests looking for networks, 802.11 Beacon Frames indicating traffic, and a DNS and TCP packet*  July 31, 2018 | [200](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA173.w.1.0.58) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *will cover some Bluetooth attacks in this chapter. To write Python Bluetooth scripts*  July 31, 2018 | [200](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA173.w.3.0.38) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *utilize the Python bindings to the Linux Bluez Application Programming Interface*  July 31, 2018 | [200](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA173.w.4.0.8) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the obexftp API*  July 31, 2018 | [200](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA173.w.4.0.8) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *you will need access to a Bluetooth device. Most Cambridge Silicon Radio (CSR) chipsets work fine under Linux*  August 1, 2018 | [201](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA174.w.0.0.0.0.14) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the scripts in this chapter, we will use a SENA Parani UD100 Bluetooth USB Adapter*  August 1, 2018 | [201](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA174.w.4.0.10) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *test if this operating system recognizes the device*  August 1, 2018 | [201](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA174.w.4.0.10) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *run the hciconfig config command. This prints out the configuration*  August 1, 2018 | [201](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA174.w.4.0.10) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *following sections will prove exciting. We will sniff credit cards, user credentials, takeover a UAV remotely, identify wireless hackers, and stalk and exploit Bluetooth devices*  August 1, 2018 | [201](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA174.w.6.0.48) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *always check the applicable*  August 1, 2018 | [201](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA174.w.6.0.48) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *laws concerning the passive and active interception of wireless and Bluetooth transmissions*  August 1, 2018 | [201](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA174.w.6.0.48) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *2001, the Wall of Sheep team has set up a booth at the annual DEFCON security conference*  August 1, 2018 | [201](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA174.w.8.0.1) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Passively, the team listens for users logging onto email, Web sites, or other network services without any protection or encryption*  August 1, 2018 | [201](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA174.w.8.0.1) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *team detects any of these credentials, they display the credentials on a big screen overlooking the conference floor*  August 1, 2018 | [201](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA174.w.8.0.1) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *recent years the team added a project called Peekaboo*  August 1, 2018 | [201](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA174.w.9.0.56) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *carves images right out of the wireless traffic as well*  August 1, 2018 | [201](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA174.w.9.0.56) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *benign in nature, the team excellently demonstrates how an attacker might capture the same information*  August 1, 2018 | [201](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA174.w.9.0.56.0.1) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Before sniffing a wireless network for credit card information, a quick review of regular expressions will prove useful*  August 1, 2018 | [202](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA175) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Regular expressions provide a means of matching specific strings of text*  August 1, 2018 | [202](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA175) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *access to regular expressions as part of the regular expression (re) library*  August 1, 2018 | [202](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA175) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *attacker can use regular expressions to match strings for credit card numbers*  August 1, 2018 | [202](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA175.w.9.2.3) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *use the top three credit cards: Visa, MasterCard, and American Express. If you would like to learn more about writing regular expressions for credit cards, visit http://www.regular-expressions.info/creditcard.html*  August 1, 2018 | [202](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA175.w.9.2.3) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Let’s write a small function to check a string to determine if it contains an American Express Credit Card*  August 1, 2018 | [202](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA175.w.10.0.51) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Notice the following regular expression; it ensures the credit card must begin with 3, followed by either a 4 or 7. Next, the regular expression matches13 more digits to ensure a total length of 15 digits*  August 1, 2018 | [202](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA175.w.10.0.51) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *examine the regular expressions necessary to find MasterCards and Visa credit cards. MasterCard credit cards begin with any number between 51 and 55 and are 16 digits long*  August 1, 2018 | [203](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA176.w.0.0.0.0.10) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Visa credit cards start with the number 4, and are either 13 or 16 digits long*  August 1, 2018 | [203](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA176.w.6.0.11) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *expand our findCreditCard() function to find MasterCard and Visa credit card numbers*  August 1, 2018 | [203](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA176.w.6.0.11) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Now we must match these regular expressions inside of sniffed wireless packets*  August 1, 2018 | [203](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA176.w.7.0.15) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *remember to use monitor mode for sniffing purposes, as it allows us to observe both frames intended and not intended for us as*  August 1, 2018 | [203](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA176.w.7.0.15) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *For parsing packets intercepted on our wireless interface, we will use the Scapy library*  August 1, 2018 | [203](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA176.w.10.0.30) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Notice the use of the sniff() function. Sniff() passes each TCP packet as a parameter to the findCreditCard() function*  August 1, 2018 | [203](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA176.w.10.0.30) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we do not intend not for anybody to steal credit card data. In fact, this very attack landed a wireless hacker and thief named Albert Gonzalez in jail for over twenty years*  August 1, 2018 | [204](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA177.w.3.0.60) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *hopefully you realize this attack is relatively simple and not as sophisticated as generally believed*  August 1, 2018 | [204](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA177.w.3.0.60) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we will examine a separate scenario where we attack an unencrypted wireless network to steal personal information*  August 1, 2018 | [204](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA177.w.5.0.39) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *From The TrenchesThe Demise of the Shadow Crew*  August 1, 2018 | [204](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA177.w.5.0.39.0.1) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *September of 2008, the US District Attorney of Massachusetts indicted Albert Gonzalez for wire fraud, damage to computer systems, access device fraud and aggravated*  August 1, 2018 | [204](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA177.w.5.0.39.0.1) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *identity theft (Heymann, 2008). Albert*  August 1, 2018 | [205](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA178) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *used a wireless sniffer to gain access to the computer systems of the TJX Corporation*  August 1, 2018 | [205](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA178) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Corporation encrypted their traffic with the flawed and less secure WEP encryption*  August 1, 2018 | [205](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA178) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *oversight allowed Gonzalez’s Shadow Crew to intercept and decrypt the wireless traffic*  August 1, 2018 | [205](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA178) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *wireless sniffer, along with a variety of other techniques, gained access to over 45.7 million customer cards*  August 1, 2018 | [205](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA178) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Seven feet tall and a veteran hacker, Steven Watt conspired with the Shadow Crew in their activities. At the time Watt had a budding career writing real-time trading software (Zetter, 2009). For his role in writing the wireless sniffer, the state sentenced Watt to two years in prison and forced him to pay restitution to TJX in the amount of $171.5 million*  August 1, 2018 | [205](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA178.w.3.0.87) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Often these networks fail to encrypt traffic and lack any enterprise authentication or encryption controls. This section examines a scenario where a few lines of Python can exploit this situation*  August 1, 2018 | [205](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA178.w.3.0.151) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *I stayed in a hotel that offered wireless connectivity to guests. After connecting to the wireless network, my web browser directed me to a web page to log on to the network*  August 1, 2018 | [205](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA178.w.6.0.28) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *After providing this information, my browser posted an unencrypted HTTP page back to the server to receive an authentication cookie*  August 1, 2018 | [205](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA178.w.6.0.28) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Examining this initial HTTP post revealed something interesting. I noticed a string similar to PROVIDED\_LAST\_NAME=OCONNOR&PROVIDED\_ROOM\_NUMBER=1337*  August 1, 2018 | [205](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA178.w.7.0.50) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *plaintext transmission to the hotel server contained both my last name and hotel room number. The server made no attempt to protect this*  August 1, 2018 | [205](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA178.w.7.0.50) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *a customer’s last name and room number provided the credentials required to eat a steak dinner in the guest restaurant, receive an expensive massage, or even buy items at the gift shop*  August 1, 2018 | [205](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA178.w.8.0.25) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *imagine that hotel guests would not want an attacker to get a hold of this personal information*  August 1, 2018 | [205](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA178.w.8.0.25) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *now use Python to capture this information from other hotel guests*  August 1, 2018 | [206](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA179.w.3.2.5.0.10) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Starting a wireless sniffer in Python is rather simple*  August 1, 2018 | [206](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA179.w.3.2.5.0.10) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *identify our interface to capture traffic*  August 1, 2018 | [206](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA179.w.3.2.5.0.10) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Next, our sniffer listens for traffic using the sniff() function—notice this function filters only TCP traffic*  August 1, 2018 | [206](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA179.w.5.0.29) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *and forwards all packets to a procedure named findGuest()*  August 1, 2018 | [206](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA179.w.5.0.29) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *the function findGuest receives the packet, it determines if the intercepted packet contains any personal information*  August 1, 2018 | [206](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA179.w.5.0.29) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *First, it copies the raw contents of the payload to a variable named raw*  August 1, 2018 | [206](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA179.w.7.0.7) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *then build a regular expression to parse the last name and room number of the guests*  August 1, 2018 | [206](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA179.w.7.0.7) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *our regular expression for last names accepts any string that begins with LAST\_NAME and terminates with an ampersand symbol (&)*  August 1, 2018 | [206](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA179.w.7.0.7) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *regular expression for the hotel guest’s room number captures any string that begins with*  August 1, 2018 | [206](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA179.w.7.0.7) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *ROOM\_NUMBER. def*  August 1, 2018 | [206](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA179.w.7.0.75) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Putting all this together, we now have a wireless hotel guest sniffer to capture the last name*  August 1, 2018 | [206](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA179.w.7.0.75.0.16) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *and hotel room number of any guest who connects to the wireless*  August 1, 2018 | [207](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA180) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *we need to import the scapy library in order to have the capability to sniff traffic and parse it. import*  August 1, 2018 | [207](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA180) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *I cannot emphasize enough at this time that collection of this information potentially violates several state, federal, and national*  August 1, 2018 | [208](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA181.w.0.0.0.0.8) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Highlighted text cannot be displayed*  August 1, 2018 | [208](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA181.w.0.0.0.0.8) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Highlighted text cannot be displayed*  August 1, 2018 | [208](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA181.w.2.0.35) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Highlighted text cannot be displayed*  August 1, 2018 | [208](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA181.w.2.0.35) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Highlighted text cannot be displayed*  August 1, 2018 | [208](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA181.w.2.0.35) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Highlighted text cannot be displayed*  August 1, 2018 | [208](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA181.w.4.0.55) | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | *Highlighted text cannot be displayed*  August 1, 2018 | [208](http://play.google.com/books/reader?printsec=frontcover&output=reader&id=2XliiK7FKoEC&source=books-notes-export&pg=GBS.PA181.w.4.0.55) | |

|  |
| --- |
| Limit reached.  394 notes/highlights or bookmarks not synced. |