Network Reconnaissance

Technical Report

Mark Ennis

October 20th, 2022

Introduction

This report covers a variety of active and passive network reconnaissance techniques and tools. Port scanning, network discovery, fingerprinting, web scanning, and enumeration are all tested and discussed. When possible, tools and methods to detect and prevent these reconaissance activities are also outlined. The list of tools and techniques is by no means exhaustive and serves as an introduction. All reconnaissance was performed on a wireless home network.

A detailed list of the files contained within the report submission can be found in Appendix B.

Body

nmap

Nmap is an open-source active reconnaissance tool used to map and explore networks. It is highly focused on performance to work well on large networks. Using a variety of different protocols and techniques, nmap attempts to map hosts to addresses and determine which services are running on each host. Nmap can also perform additional fingerprinting on the host machines and services. By analyzing the responses to requests, nmap attempts to identify a remote host's operating system (OS) by fingerprinting the TCP/IP stack.

Host discovery, without port scanning, is performed on a local network using ARP requests:

```
┌──(mark㉿orangecounty)-[~]
└─$ sudo nmap -PR 192.168.0.0/24 -sn
[sudo] password for mark:
Starting Nmap 7.92 ( https://nmap.org ) at 2022-10-18 19:22 PDT
Nmap scan report for dlinkrouter (192.168.0.1)
Host is up (0.0037s latency).
MAC Address: 40:9B:CD:A1:A4:C4 (D-Link International)
Nmap scan report for 192.168.0.114
Host is up (0.0054s latency).
MAC Address: B8:27:EB:81:88:62 (Raspberry Pi Foundation)
Nmap scan report for 192.168.0.124
Host is up (0.092s latency).
MAC Address: EA:43:FA:E0:AE:4F (Unknown)
Nmap scan report for 192.168.0.132
Host is up (1.5s latency).
MAC Address: F4:4E:E3:C6:E1:84 (Intel Corporate)
Nmap scan report for 192.168.0.174
Host is up (1.5s latency).
MAC Address: EC:B5:FA:07:AC:2A (Philips Lighting BV)
Nmap scan report for 192.168.0.187
Host is up (0.059s latency).
MAC Address: 8E:7E:DC:57:F4:16 (Unknown)
Nmap scan report for 192.168.0.195
Host is up (0.064s latency).
MAC Address: 4C:D5:77:35:2C:1F (Chongqing Fugui Electronics)
Nmap scan report for 192.168.0.162
Host is up.
Nmap done: 256 IP addresses (8 hosts up) scanned in 9.73 seconds

┌──(mark㉿orangecounty)-[~]
└─$
```

Outside of the local network, ARP cannot be used. By default, nmap attempts to scan external networks using ICMP echo requests, and TCP SYN requests are sent to ports 80 and 443. By using ports 80 and 443, nmap can bypass many security measures that allow incoming HTTP/S connections. However, many networks drop ICMP echo requests by default. If a web server is not running on the host, incoming TCP connections to ports 80 and 443 can be dropped. No method of network discovery is fully accurate.

Port scanning can be accomplished using several different protocols and options. Scanned ports are classified into one or more of four states:

- OPEN - An application is listening for traffic on this port
- FILTERED - A network security measure is blocking the port and it cannot be determined if it is open or closed.
- UNFILTERED - The port is responding to the probes but it cannot be determined if the is open or closed.
- CLOSED - No application is listening on this port

Some of the common nmap TCP port scans:

- SYN - TCP requests with the SYN flag are sent out. If a SYN-ACK is returned, the port is listening for incoming TCP connections. Otherwise, depending on the TCP/IP stack implementation, nmap determines if it is closed, filtered, or unfiltered.
- FIN - TCP FIN requests with the FIN flag are sent out and nmap attempts to determine the port state.
- NULL - TCP requests with no flags set are sent out. Many network security measures did not take this edge case into account. This is not a valid combination of flags and can be safely dropped.
- XMAS - The PSH, URG, and FIN flags are all set, lighting up the packet “like a Christmas tree”. Similar to a NULL scan, this flag combination would only be used to evade security measures and can be dropped.
- --scanflags - Custom TCP flags can be set using either a numerical value or chaining keywords. This can help get around network security measures that have been configured against the above combinations

An example TCP SYN port scan can be found below:

```
┌──(mark㉿orangecounty)-[~]
└─$ sudo nmap -sS 192.168.0.114
Starting Nmap 7.92 ( https://nmap.org ) at 2022-10-18 20:16 PDT
Nmap scan report for 192.168.0.114
Host is up (0.012s latency).
Not shown: 995 closed tcp ports (reset)
PORT     STATE SERVICE
21/tcp   open  ftp
22/tcp   open  ssh
80/tcp   open  http
111/tcp  open  rpcbind
2049/tcp open  nfs
MAC Address: B8:27:EB:81:88:62 (Raspberry Pi Foundation)

Nmap done: 1 IP address (1 host up) scanned in 0.69 seconds

┌──(mark㉿orangecounty)-[~]
└─$ 
```

Scanning a specific machine, nmap found several services that were accepting TCP connections. Running the scan with Aggressive scan options allows nmap to perform additional fingerprinting of each service running:

```
┌──(mark㉿orangecounty)-[~]                                        [29/32]
└─$ sudo nmap -sS 192.168.0.114 -A
Starting Nmap 7.92 ( https://nmap.org ) at 2022-10-18 20:16 PDT
Nmap scan report for 192.168.0.114
Host is up (0.0046s latency).
Not shown: 995 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
21/tcp   open  ftp     vsftpd 3.0.3
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_-rw-r--r--    1 ftp      ftp            15 Jan 23  2022 test.txt
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to ::ffff:192.168.0.162
|      Logged in as ftp
|      TYPE: ASCII
|      No session bandwidth limit
|      Session timeout in seconds is 300
|      Control connection is plain text
|      Data connections will be plain text
|      At session startup, client count was 2
|      vsFTPd 3.0.3 - secure, fast, stable
|_End of status
22/tcp   open  ssh     OpenSSH 7.9p1 Raspbian 10+deb10u2+rpt1 (protocol 2.0)
| ssh-hostkey:
|   2048 c8:b0:50:5e:f8:63:b1:d6:30:82:81:2e:45:b9:1a:0b (RSA)
|   256 c5:24:b2:87:8d:7c:ce:13:f1:10:41:40:cd:2f:c5:73 (ECDSA)
|_  256 2e:e1:78:2b:e8:44:be:ea:02:bf:b1:70:fe:7d:8b:80 (ED25519)
80/tcp   open  http    Apache httpd 2.4.38 ((Raspbian))
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache/2.4.38 (Raspbian)
111/tcp  open  rpcbind 2-4 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2,3,4        111/tcp   rpcbind
|   100000  2,3,4        111/udp   rpcbind
```

Nmap also allows for ICMP sweep scanning and UDP port scanning. Many ICMP packet types are considered suspicious and are ignored by modern networks. UDP port scanning is slower than TCP port scanning, despite the additional overhead that TCP connections require. In order to match the responses to each UDP packet sent, nmap can only have one unanswered response on the wire at a given time. The inability to send concurrent requests slows down the scan immensely.

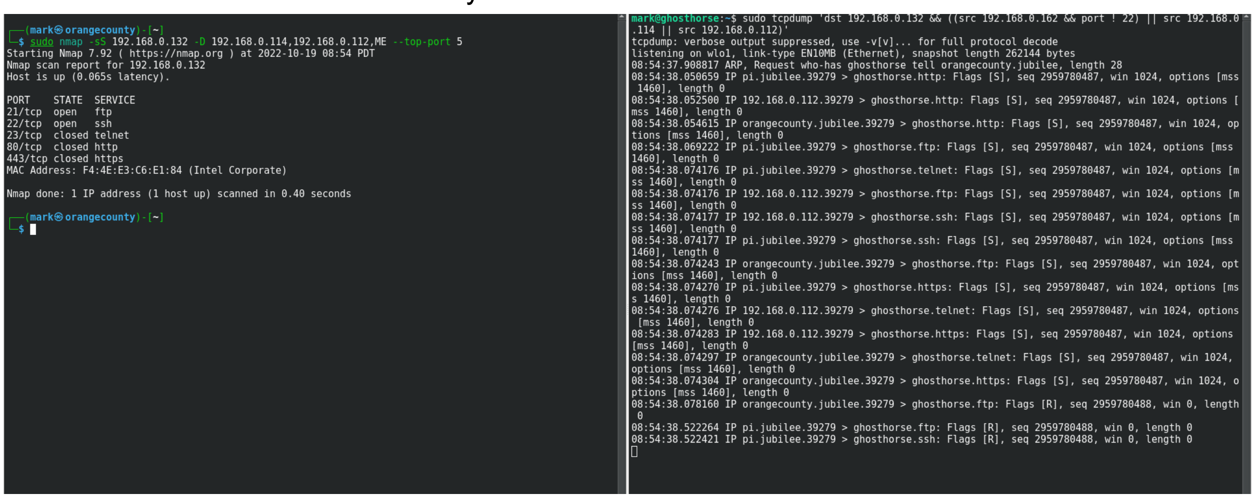
It is slightly harder to create rules that block new incoming connections from UDP requests because they are connectionless. The conntrack iptables module provides the ability to block NEW UDP connections, likely based on ephemeral source-port tracking and application layer protocol headers (i.e. the DNS Identification field).

A more detailed exploration of different default nmap port scans and how to defend against them can be found in Appendix A.

Many nmap scans are easy to discover and counteract using widely available security measures. Network firewalls and NIDS/NIPS solutions can be written to address many out-of-the-box nmap scans. Firewall rules and custom Snort rules can prevent or slow down many of these scans.

Additionally, Snort supports preprocessors such as sfportscan that have been purpose-built to detect network reconnaissance attempts. When possible, an administrator should rely on purpose-built solutions rather than creating something new. A purpose-built solution often has subject-matter experts designing, testing, and using it, and it is more likely to handle edge cases. The maintenance cost of updating is often lower as well.

Nmap has several options available that make it harder to detect. Many of the solutions that detect network scans rely on sudden bursts in traffic. Nmap provides timing templates that can be used to control the speed requests are made. Two of these timing options, *Paranoid* and *Sneaky*, slow down requests to a rate that avoids most IDS rules. It will take far longer to get the results of the scan. Nmap also provides more fine-grained timing options to control the number of concurrent requests, the parallelization of a scan using multiple hosts, timeouts, and rate limiting. Nmap supports several different options to obfuscate the host that is performing a scan. An attacker can spoof their MAC address, use proxies to route their requests, or spoof requests from other hosts using the decoy (`-D`) option. The decoy option accepts a CSV list of hosts and makes it appear as though these hosts are also performing network scans. The goal is to hide the machine that is doing the actual port scanning by forcing defenders to check additional machines if and when they are alerted to the network reconnaissance:



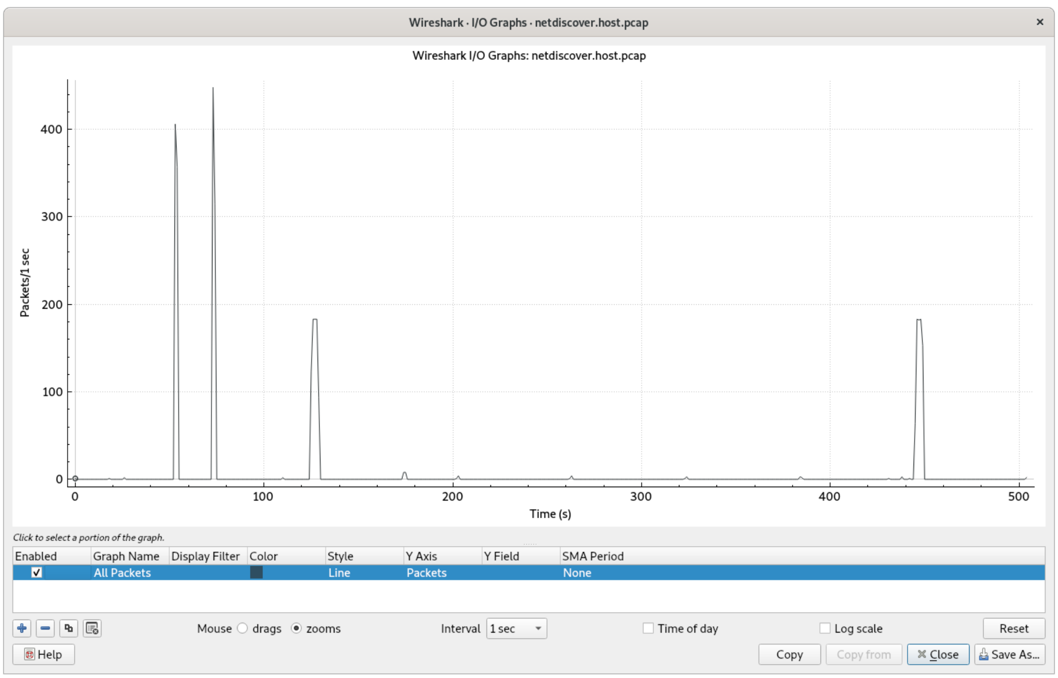
netdiscover

Compared to nmap, netdiscover is a relatively simple tool. Netdiscover can be used to passively or actively map out a local network using ARP requests. In passive mode, netdiscover listens for ARP broadcast messages to learn about devices on a network. In active mode, netdiscover sends out its own ARP Broadcast messages and spoofs its source IP to help disguise the source. Netdiscover attempts to identify the device manufacturer using its Organizationally Unique Identifier (OUI).

```
Currently scanning: Finished!   |   Screen View: Unique Hosts

13 Captured ARP Req/Rep packets, from 5 hosts.   Total size: 546
_____________________________________________________________________________
  IP            At MAC Address     Count     Len  MAC Vendor / Hostname
-----------------------------------------------------------------------------
192.168.0.1     40:9b:cd:a1:a4:c4      3     126  D-Link International
192.168.0.124   ea:43:fa:e0:ae:4f      2      84  Unknown vendor
192.168.0.195   4c:d5:77:35:2c:1f      3     126  CHONGQING FUGUI ELECTRONICS CO.,LTD.
192.168.0.132   f4:4e:e3:c6:e1:84      3     126  Intel Corporate
192.168.0.187   8e:7e:dc:57:f4:16      2      84  Unknown vendor
```

When in passive mode, netdiscover cannot be detected by other machines on the network. It sends out no traffic and does not interact with other hosts. When in active mode, it can be discovered by anyone monitoring the traffic. By default, each ARP request is sent out with only one millisecond between each request. Inspecting the ARP traffic, it is easy to see the number of ARP requests spiking up:



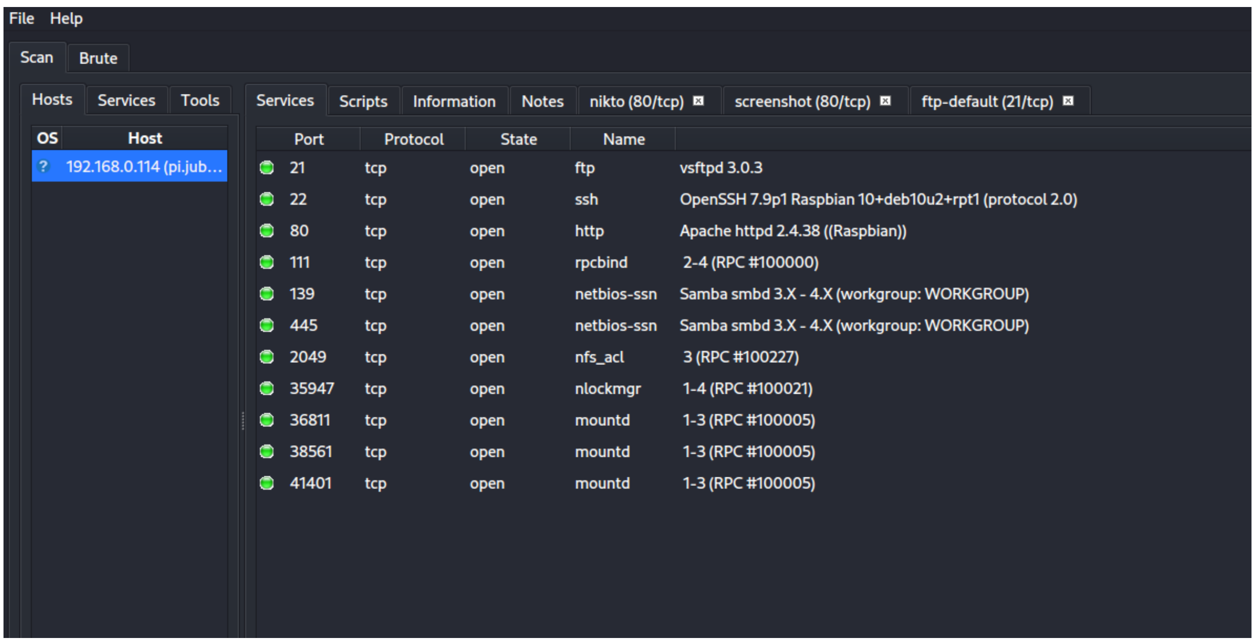
The delay between each ARP request can be controlled. By slowing down the number of ARP requests sent out to a rate similar to normal traffic, the drastic spikes may no longer be visible. However, since netdiscover uses ARP spoofing to hide the host it is running on, it may be possible for Snort to detect and raise an alert. This can theoretically be accomplished using the snort arpspoof preprocessor but was not implemented for this report. If avoiding detection is of the utmost priority, running netdiscover in passive mode for an extended period is the safest option.

While it may be possible to detect when an ARP scan is being run through spikes in traffic or ARP spoof detection, there is less tooling to filter and drop ARP traffic. Many network security measures analyze and control traffic at the Network and Application layers. ARP is Link Layer traffic and is contained to a single network.

A major limitation of netdiscover is that it cannot perform discovery outside of the network that a machine is physically connected to. ARP cannot be used to send requests outside of a network, so netdiscover is unable to see beyond those limitations. Netdiscover is simpler, less powerful, and less flexible than nmap. However, passive mode makes it a worthwhile tool on its own.

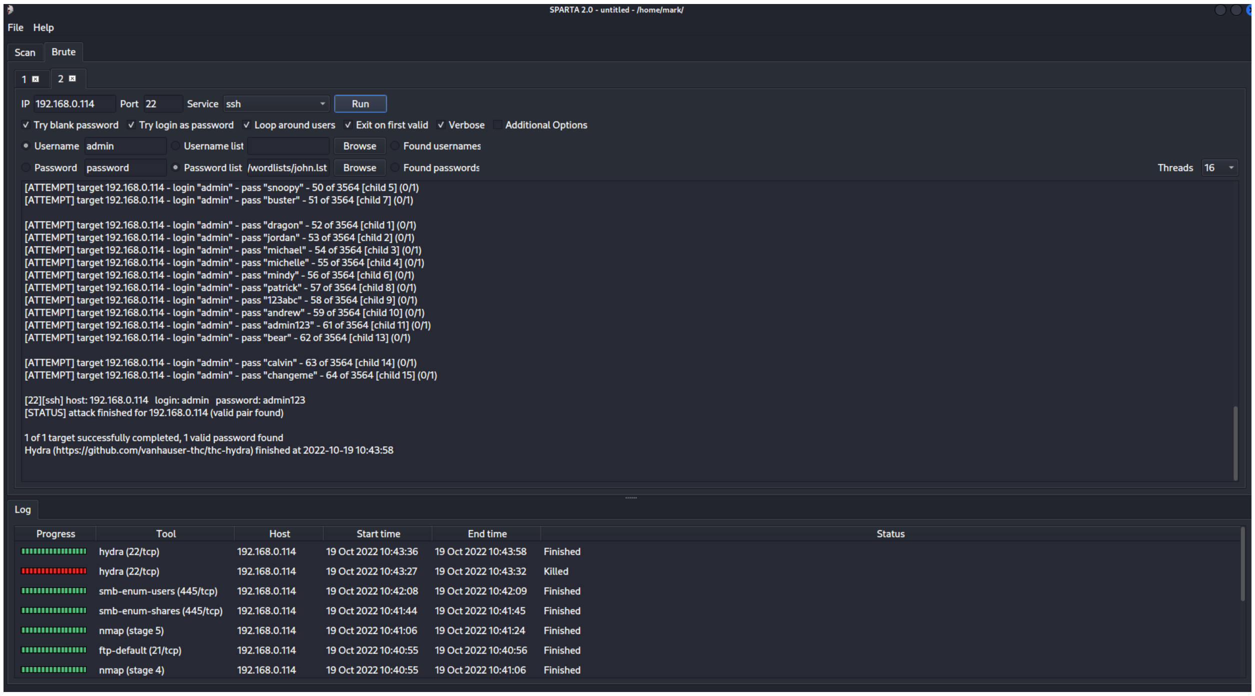
Sparta

Sparta is a GUI built in Python over top of several different security tools. It facilitates network infrastructure penetration testing by allowing a user to perform network scans, brute force attacks, data exfiltration, and service fingerprinting. Using nmap, it can map networks, hosts, or individual machines to discover which services are running and open.



A more detailed list of network discovery features available through nmap can be found in the nmap section.

Using Sparta, once services have been discovered, a variety of fingerprinting and attack tools are available. Non-Windows machines can extract the password policies of remote Windows machines using the polenum tool. SMB Users and Shares can be enumerated using nmap and rpcclient. Websites can be scanned using WhatWeb to determine what Content Management Systems (CMS), platforms, analytics, libraries, and web servers may be used by a domain. Using these tools, Sparta can find several vulnerabilities within a network or machine. User information and available services can be used to help direct a brute-force attack against an open service that is unprotected by rate-limiting. Using hydra, a parallelized login cracker, Sparta can launch automated brute-force password-cracking attempts



IDS rules can be written that make brute-force login attempts take an extremely long time. Using the `detection_filter` option, Snort can define rate limits that drop packets that exceed a certain number of connection attempts. The attacker can of course slow down their attack to avoid hitting the set rate, but an effective brute force attack relies on speed. Using a combination of strong passwords and rate limiting can help prevent successful brute-force attacks. If the option is available for a service, as it is for SSH, using public key authentication is even more effective.

Enumeration Techniques

Enumeration is a technique that extracts data by making connections to a system and performing directed queries. Enumeration techniques are highly dependent on the target OS and the services that are running on the target. As such, they are more effective after performing network discovery of another form of fingerprinting. Sparta uses several enumeration techniques relying on rpcinfo, netcat, and nmap. Below are a description of several enumeration methods and tools.

enum4linux

Enum4linux allows Linux systems to perform enumerations on Windows systems and Samba shares. It is a wrapper around the Samba libraries that allows user lists, machine lists, sharelists, password policy information, operating system, group, and member information to be enumerated.

```
┌──(mark㉿orangecounty)-[~]
└─$ enum4linux 192.168.0.114
Starting enum4linux v0.9.1 ( http://labs.portcullis.co.uk/application/enum4linux/ ) on Wed Oct 19 12:47:13 2022

 =========================================( Target Information )=========================================

Target ........... 192.168.0.114
RID Range ........ 500-550,1000-1050
Username ......... ''
Password ......... ''
Known Usernames .. administrator, guest, krbtgt, domain admins, root, bin, none


 ===========================( Enumerating Workgroup/Domain on 192.168.0.114 )===========================


[+] Got domain/workgroup name: WORKGROUP


 ===============================( Nbtstat Information for 192.168.0.114 )===============================

Looking up status of 192.168.0.114
        PRETTYFLYFORARA <00> -         B <ACTIVE>  Workstation Service
        PRETTYFLYFORARA <03> -         B <ACTIVE>  Messenger Service
        PRETTYFLYFORARA <20> -         B <ACTIVE>  File Server Service
        WORKGROUP       <00> - <GROUP> B <ACTIVE>  Domain/Workgroup Name
        WORKGROUP       <1e> - <GROUP> B <ACTIVE>  Browser Service Elections

        MAC Address = 00-00-00-00-00-00

 ===================================( Session Check on 192.168.0.114 )===================================


[+] Server 192.168.0.114 allows sessions using username '', password ''


 ================================( Getting domain SID for 192.168.0.114 )================================

Domain Name: WORKGROUP
Domain Sid: (NULL SID)

[+] Can't determine if host is part of domain or part of a workgroup


 ==================================( OS information on 192.168.0.114 )==================================


[E] Can't get OS info with smbclient


[+] Got OS info for 192.168.0.114 from srvinfo:
        PRETTYFLYFORARAWk Sv PrQ Unx NT SNT Samba 4.9.5-Debian
        platform_id     :       500
```

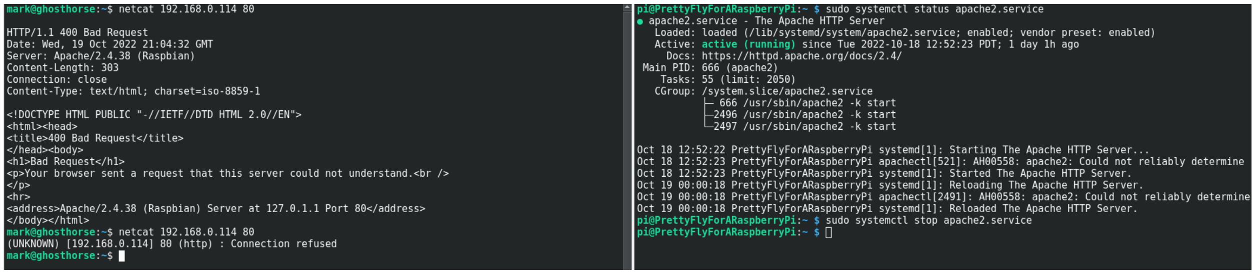
Since enum4linux is a wrapper around smbclient, it relies entirely on communicating with target machines using SMB. If for some reason, SMB traffic needs to be accessible from the wider internet, rate limiting incoming SMB connections by the source using a NIDS may help slow down this form of enumeration. Rate-limiting SMB traffic wholesale will not work if it is used for file transfers, given the volume of traffic that will generate. Setting the RestrictAnonymous registry setting to 1 will require someone to be able to authenticate before performing SMB enumeration. The best defense is to disable Samba wherever possible to avoid leaking data.

Windows 11 now has SMB rate limiting for authentication by default to help combat SMB brute force attacks.

telnet and netcat

Telnet and netcat are two of the most widely used enumeration tools. They come preinstalled in most Linux distributions or are available through the package manager. Telnet is a command-line utility used for communicating over a network using the Telnet protocol. The Telnet protocol is an Application layer protocol that uses TCP. Netcat is another command-line utility used for writing data across network connections. Netcat is far more flexible than telnet.

Netcat allows for UDP messages to be sent, as well as allowing for the transmission of arbitrary binary data. Either of these tools can be used to perform banner grabbing to determine which services are running on a host:



Defending against these utilities is difficult. It may be possible to inspect all TCP packet payloads for Telnet header signatures and drop packets that contain them. This could have serious performance impacts as all non-control TCP packets would have to have their contents inspected. Additionally, it would not prevent netcat from accomplishing the same end. Since these tools can send individually crafted requests that are targeted, it is easy for them to avoid detection.

Instead, an administrator should seek to limit the amount of data that banner grabbing can provide. Disable version and vendor information wherever possible. Changing default HTTP response pages (such as HTTP 400 or 404) to have misleading or no information will limit what attackers can learn. Additionally, firewall rules should be tightened so that only connections to valid services from expected ports are allowed through.

rpcinfo

Remote Procedure Call (RPC) is a network client-server protocol for calling services on remote machines across a network. The command-line utility rpcinfo can return information about RPC applications listening on a remote machine by interrogating the RPC portmapper which binds client requests to ports:

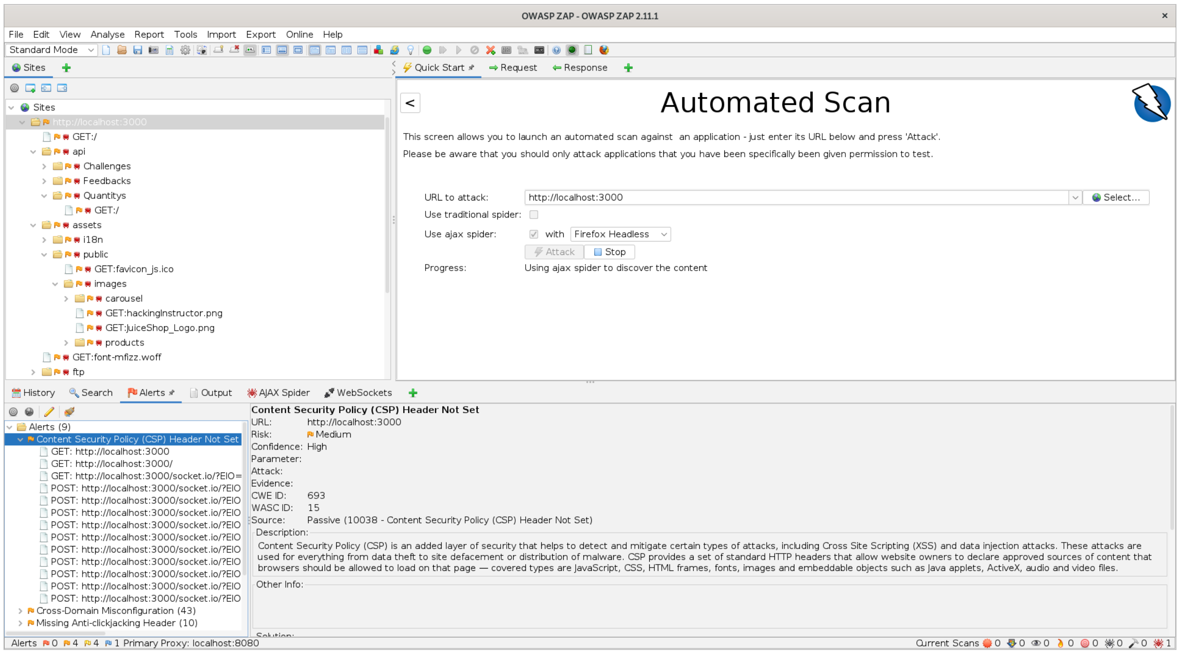
```
┌──(mark㉿orangecounty)-[~]
└─$ rpcinfo -p 192.168.0.114
   program vers proto   port  service
    100000    4   tcp    111  portmapper
    100000    3   tcp    111  portmapper
    100000    2   tcp    111  portmapper
    100000    4   udp    111  portmapper
    100000    3   udp    111  portmapper
    100000    2   udp    111  portmapper
    100005    1   udp  43493  mountd
    100005    1   tcp  38561  mountd
    100005    2   udp  51487  mountd
    100005    2   tcp  41401  mountd
    100005    3   udp  36478  mountd
    100005    3   tcp  36811  mountd
    100003    3   tcp   2049  nfs
    100003    4   tcp   2049  nfs
    100227    3   tcp   2049
    100003    3   udp   2049  nfs
    100227    3   udp   2049
    100021    1   udp  33000  nlockmgr
    100021    3   udp  33000  nlockmgr
    100021    4   udp  33000  nlockmgr
    100021    1   tcp  35947  nlockmgr
    100021    3   tcp  35947  nlockmgr
    100021    4   tcp  35947  nlockmgr
```

It is possible to prevent RPC banner enumeration by closing ports 111 and 32771 if no RPC applications are in use. These are the ports that the portmapper (rpcbind) uses to handle incoming requests. Additionally, it is possible to configure RPC to require authentication which limits who can make requests to the portmapper.

Web-Application Scanners

Web-application scanners are tools that scan web applications to fingerprint various tools, technologies, and services they are built with, as well as discover common vulnerabilities that may be possible. They can be used by developers and administrators to help perform security audits on sites running in production and development environments. WhatWeb is a web-application scanner included in Kali Linux and used by Sparta. Another open-source alternative is OWASP® ZAP.

The Open Web Application Security Project® (OWASP) is a non-profit organization that seeks to improve security in software. OWASP® ZAP, the Zed Attack Proxy, comes with a GUI and allows a user to scan a site, craft and manipulate in-flight HTTP/S requests, and interact with web applications through an embedded browser.



The best defense against a web-application scanner is to use one before, during, and after deployment, on an ongoing basis. While a malicious actor could use one to discover exploits, they primarily serve as a defensive tool that allows teams to proactively fix and remove vulnerabilities from applications.

OSINT

OSINT, or Open Source Intelligence, is the practice of gathering actionable information from open or publicly available sources. In the context of cybersecurity, it is a form of passive reconnaissance that involves learning about organizations, networks, and individuals through a variety of sources including social media, DNS records, general and specialized search engines, data dumps, and other websites.

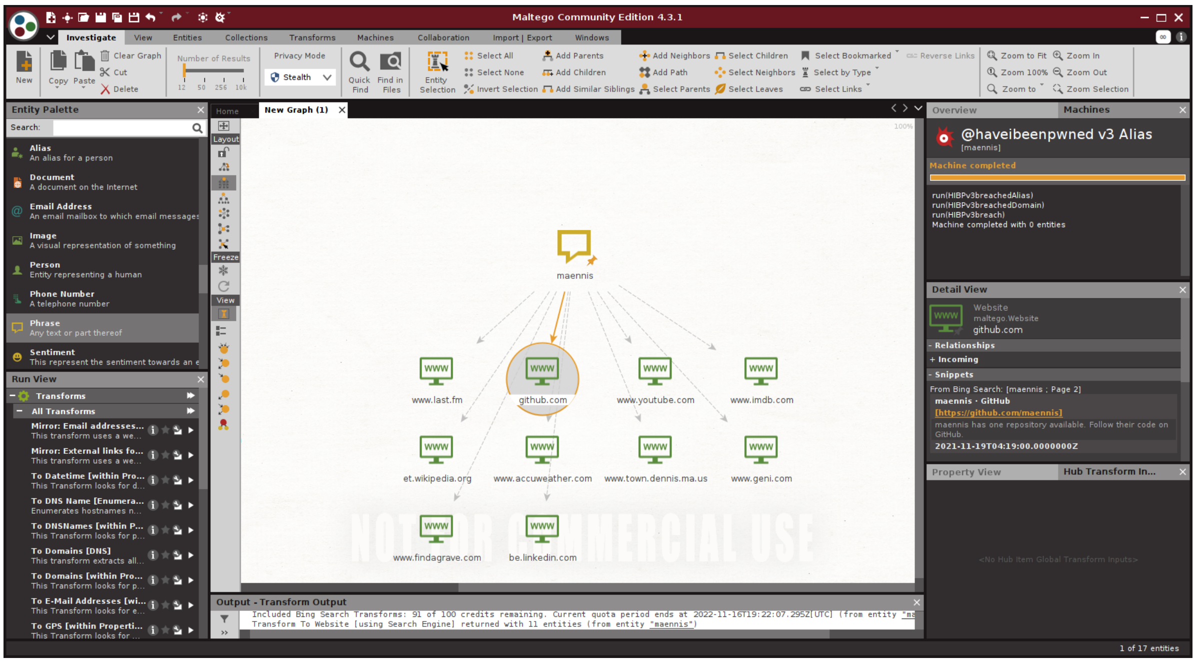
theHarvester

theHarvester is a command-line tool that gathers OSINT about a company or domain name from a broad spectrum of sources. It takes in a company or domain name, and searches its sources to produce a list of IP addresses associated with the company or domain, DNS records, known hosts, emails, and more depending on the source. It can also be configured to draw from closed sources, such as the Shodan. Due to the wide variety of sources it can pull from, theHarvester can produce a wealth of results. It appears to be simple to use, and like other aggregation tools such as Sparta, it is more powerful than some of the tools on their own.

To avoid performing OSINT research on any non-consenting parties, this tool was not used during this report.

Maltego

Maltego is another OSINT tool that also pulls from a variety of sources. Unlike theHarvester, it comes with a GUI and visualizes the results of information into a graph format. It allows a user to create nodes representing emails, search terms, people, or other entities. Then, using scripts called transforms, new nodes can be discovered and attached to existing entities.



On its own, the Community Edition (CE) of Maltego is flexible but not particularly powerful. By adding additional paid transforms, a user can get access to a much wider set of sources and prebuilt scripts that can pull more data from each source.

The representation of data in a graph format allows for easy link analysis and facilitates finding correlations and patterns within large data sets. Link analysis is the practice of analyzing the relationships that connect different nodes.

Conclusion

Reconnaissance is a key stage in attacking and defending a network. OSINT techniques can be first used to help find attack vectors, vulnerabilities, and target networks. After that, host discovery and network mapping reveal possible endpoints to attack. Port scanning will reveal which targets are the most vulnerable, then enumeration, banner grabbing, and fingerprinting can further reveal key information for further attacks. Without employing these techniques, an attacker is more likely to alert administrators to their presence then to gain any actionable information.

Appendix A - Common nmap Scans

Scan	Individual host scan
Description	Attempts host discovery of a single host without performing a port scan. Sends TCP packets with the SYN flag set to 80 and 443. The Firewall and Snort rules that detect and prevent this scan only work for machines that are not running web servers on HTTP / HTTPS.
Protocol	TCP
Command	`nmap -sn $ip_address`

Snort Rule	`alert tcp any any`
	
	`-> $HOME_NET 80,443 (msg: "NMAP TCP HOST SCAN"; flags:S; sid:$sid;)`

Firewall Rule	`sudo iptables -A INPUT -p tcp -m conntrack --ctstate NEW \` `--match multiport --dport 80,443 -j DROP`

Scan	Network scan with no probing
Description	Attempts host discovery of a single host without performing a port scan. Blocking ARP requests is not a valid option for most machines, and Snort is not effective at inspecting Link Layer traffic such as ARP requests.
Protocol	ARP
Command	`nmap -sn $subnet_CIDR_mask` `nmap -PR $subnet_CIDR_mask -sn`
Snort Rule	-
Firewall Rule	-

Scan	TCP SYN Scan
Description	Attempts to discover which services are running on a provided host value by sending TCP SYN requests and checking the responses. A host can be a network, a CIDR range, a domain name, or an IP address. A snort rule can be written to identify and drop packets from a host sends too many SYN requests within a given period of time. A stateful firewall can be used to block incoming SYN requests on any ports that do not have a TCP service running.
Protocol	TCP

Command	`nmap -sS $host`
Snort Rule	`drop tcp any any -> $HOME_NET any ( msg: "NMAP SYN SCAN";flags: S; detection_filter: track by_src, count 10, seconds 5; sid:$sid;)`
Firewall Rule	`sudo iptables -A INPUT -p tcp -m conntrack --ctstate NEW \ --match multiport -j DROP`

Scan	TCP FIN Scan
Description	Attempts to discover which services are running on a provided host value by sending TCP FIN requests and checking the responses. A host can be a network, a CIDR range, a domain name, or an IP address. A snort rule can be written to identify packets from a host sends too many FIN requests within a given period of time. TCP FIN requests should not be blocked by a firewall.
Protocol	TCP
Command	`nmap -sF $host`
Snort Rule	`alert tcp any any -> $HOME_NET any ( msg: "NMAP FIN SCAN";flags: F; detection_filter: track by_src, count 10, seconds 5; sid:$sid;)`
Firewall Rule	-

Scan	TCP XMAS Scan
Description	Attempts to discover which services are running on a provided host value by sending TCP requests with several non-conventional flags set and checking the responses. A host can be a network, a CIDR range, a domain name, or an IP address. Snort and Firewall rules can be written to alert or drop requests with these combinations of flags set. There is no valid usecase where these flags should all be set.
Protocol	TCP
Command	`nmap -sX $host`
Snort Rule	`alert tcp any any -> $HOME_NET any (msg: "NMAP CHRISTMAS TREE SCAN"; flags:FPU; sid:1000002;)`
Firewall Rule	`sudo iptables -A INPUT -p tcp --tcp-flags FIN,PSH,URG FIN,PSH,URG -j DROP`

Scan	UDP Scan
Description	Attempts to discover which services are running on a provided host value by sending UDP requests. A host can be a network, a CIDR range, a domain name, or an IP address. Firewall rules can be written to alert or drop requests to drop incoming UDP requests that are unrelated to existing traffic using the conntrack module. Snort has a harder time classifying UDP traffic as new versus established. A rule can be written to create an alert when several UDP requests are received from the same host in a short period of time. This is likely to trigger false positives, especially because nmap UDP scans are slower than TCP scans. The sfportscan Snort preprocessor can detect UDP port scans.
Protocol	TCP
Command	`nmap -sX $host`
Snort Rule	`udp any any -> $HOME_NET any (msg: "NMAP UDP SCAN"; detection_filter: track by_src, count 5, seconds 5; sid:$sid;)`
Firewall Rule	`sudo iptables -A INPUT -p udp -m conntrack --ctstate NEW -j DROP`

Appendix B - Submission File Structure

Directory	File	Description
./report/	Report.pdf	This report.
./webscanner	zap-host.pcap	Packet capture from the machine running the OWASP ZAP webscanner.
	zap-target-apache.pcap	Packet capture from the machine running an Apache webserver.
	zap-target-node.pcap	Packet capture from the machine running an Express web server, a NodeJS server, and a React Web Application.
	zap.mp4	The demo video of the OWASP ZAP webscanner being run against a default Apache installation and a Web Application.
./sparta/	sparta-target.pcap	Packet capture from a machine that was targeted by different Sparta scans and attacks.
	sparta-host.pcap	Packet capture from the machine running Sparta.
	sparta.mkv	The demo video of Sparta being used to scan, fingerprint, and perform brute force attacks.
./nmap/	nmap-host.pcap	Packet capture from the machine that was running different nmap scans.
	nmap-192.168.0.114.pcap	Packet capture from a machine running a host-based firewall that was dropping some of the packets used by nmap.
	nmap-192.168.0.132.pcap	Packet capture from a machine running a host
	alert	The Snort alert file
	snort.log.1666231709	The snort log file in packet capture format
	nmap.mp4	The demo video of various nmap scans
./netdiscover/	netdiscover.192.168.0.114.pcap	Packet capture from a machine in the same network as one running netdiscover

	netdiscover.host.pcap	Packet capture from a machine running netdiscover
	netdiscover.192.168.0.132.pcap	Packet capture from a machine in the same network as one running netdiscover
	netdiscover.mp4	The demo video of netdiscover
./enumeration	enumeration-host.pcap	Packet capture from a machine running performing various enumeration techniques including rpcinfo, netstat banner grabbing, and enum4linux.
	enumeration-target.pcap	Packet capture from a machine that was the target of enumeration techniques.
	enumeration.mp4	A demo video showing various enumeration techniques including rpcinfo, netstat banner grabbing, and enum4linux.
./enum4linux/	enum4linux-host.pcap	Packet capture from a machine running performing enumeration techniques using enum4linux.
	enum4linux-target.pcap	Packet capture from a machine that was the target of enumeration techniques.
	enum4linux.mp4	The demo video of enum4linux.