A Survey on Network Attack Surface Mapping

DOUGLAS EVERSON and LONG CHENG, Clemson University School of Computing, Clemson, USA

Network services are processes running on a system with network exposure. A key activity for any network defender, penetration tester, or red team is network attack surface mapping, the act of detecting and categorizing those services through which a threat actor could attempt malicious activity. Many tools have arisen over the years to probe, identify, and classify these services for information and vulnerabilities. In this article, we survey network attack surface mapping by reviewing several prominent tools and their features and then discussing recent works reflecting unique research using those tools. We conclude by covering several promising directions for future research.

CCS Concepts: • **Security and privacy** → **Network security;**

Additional Key Words and Phrases: Network attack surface, port scan, service detection

ACM Reference Format:
Douglas Everson and Long Cheng. 2024. A Survey on Network Attack Surface Mapping. *Digit. Threat. Res. Pract.* 5, 2, Article 18 (June 2024), 25 pages. https://doi.org/10.1145/3640019

1 INTRODUCTION

Network attack surface mapping is the act of identifying a system's running processes with network exposure. In this definition, the term "system" can be taken to mean an individual computer, a network device, an application, or even an entire organization's infrastructure. Identifying services is the first step toward mapping an attack surface, which is the collection of points an attacker could attempt to exploit in order to gain unauthorized access to a system. While there are varying definitions of attack surface, with some limiting the attack surface to points with known vulnerabilities, we choose the more common definition that does not assume that vulnerabilities have already been found [90].

Network attack surface mapping is important regardless of whether one is testing or defending the attack surface. The importance is two-fold: First, it is difficult to test or defend elements of an attack surface if you don't know they exist. If the tool used to detect services is tuned too conservatively, the scan may take too long to be useful; if the tool is tuned quickly enough to return timely results, it could miss services or even entire hosts. Second, it is difficult to design a proper test or defense without knowing detailed information concerning which services are listening for connections (and on which ports they are listening). Some exploitation tools will spray every possible exploit against every possible service, but given the large number of exploits and services now available, this approach will be time-prohibitive [84].

The network attack surface of a single host can have up to 65,535 possible **Transmission Control Protocol (TCP)** ports and 65,535 possible **User Datagram Protocol (UDP)** ports. Research has shown that organizations can have hundreds, even thousands of externally facing hosts [26]. Even services intended for public discovery can be overlooked with a traditional port scan because of issues like network congestion at any point between the

Authors' address: D. Everson and L. Cheng, Clemson University, School of Computing, 821 McMillan Rd Clemson, SC 29631, USA; e-mails: {deverso, lcheng2}@clemson.edu.

ACM 2576-5337/2024/06-ART18
https://doi.org/10.1145/3640019

scanner and target system [62]. Once rate limiting, firewall blocks, and other defensive measures are accounted for, it becomes obvious quickly that developing a true picture can be a challenge.

In this article we survey network attack surface mapping from the external security tester point of view and make the following contributions:

— We present a taxonomy of active network attack surface mapping, which is composed of host discovery, port scanning, service/version detection, operating system detection, and script-based scanning. We also provide an overview of several port scanning tools, discussing their capabilities, strengths, and limitations.
— We provide a thorough survey of the current network attack surface mapping literature, classifying each work by its relevance to different steps in our taxonomy as shown in Figure 1. We discuss the features, limitations, and findings of each work.
— We discuss legal and ethical issues relevant to the use of these tools, using the Menlo Report as an ethical framework and United States law as an example to illustrate potential legal fallout.

Tools were selected based on the authors' professional knowledge of attack surface mapping tools used in the field, combined with a review of GitHub stars and qualitative review of search results for academic papers to identify the most common tools. In turn, we discuss related papers that used these tools for network attack surface mapping and closely related activities.

Related Surveys. Based on our review of all related surveys, there are several works that discuss port scanning in detail, but none survey the full spectrum of activities involved in attack surface mapping as presented in this article. Bou-Harb et al. [13] coined the term "cyber scanning" to describe port scanning conducted by someone with malicious intent as the first step in a cyber attack. They categorized cyber scanning as active or passive, by source and destination, and by the user's approach to scanning. As the authors themselves note, this article focuses on network scanning and does not describe service detection and other post-discovery enumeration. Bhuyan et al. [12] surveyed the literature concerned with detecting port scans. The detection methodologies are categorized as single-source or distributed. The authors ultimately classified detection approaches by the methods used; e.g., algorithmic, rule-based, and clustering were some of the different methods described. Since Bhuyan's work is focused more on detection, they did not describe service detection or other post-discovery enumeration. Barnett and Irwin [10] created a taxonomy of scanning techniques by scan type and attributes based in part on their collection of two years of network traffic via their network telescope. The taxonomy divided scan types into Layer 2 scans, Layer 3 scans, scans of different speeds, and source/destination of scans. As with the previous works, there is no discussion of service detection and the other phases of network attack surface mapping. Mandal and Jadhav [52] presented a discussion of open-source network security tools. Nmap was presented as a port scanning tool that one could use to scan their own network for vulnerabilities. The authors provided a discussion of the types of scans (SYN, Connect, FIN, XMAS, etc.) but did not go into detail about Nmap's other capabilities (service detection, script scanning, etc.). Roy et al. [69] provide a thorough taxonomy of reconnaissance techniques, including a description of cyber scanning; however, their work focuses on the broader topic of reconnaissance rather than the technical capabilities of the tools.

Another noteworthy distinction that sets this article apart from existing survey papers is our approach to conducting a literature survey on network attack surface mapping from a practitioner's standpoint. This perspective is informed by the first author's decades of experience in security compliance testing, penetration testing, red teaming, and threat intelligence across military, financial, and various corporate operating systems, as well as cloud/network infrastructure and applications. We delve into related works, considering their real-world utility in engineering tests and threat-intelligence-driven exercises.

Roadmap. Following this introduction, in Section 2 we discuss the tools most commonly used in network attack surface mapping [22]. We provide a full survey of papers categorized by the functions of network attack surface mapping in Sections 3–8 before briefly discussing ethics in Section 9. We conclude in Section 10.

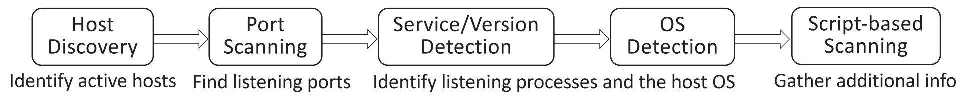

Fig. 1. Network attack surface mapping functions.

2 NETWORK ATTACK SURFACE MAPPING TOOLS

While attack surface mapping provides valuable security metrics as the first step to a test or red team operation [25], the act of attack surface mapping itself has multiple steps. We break down the steps of active attack surface mapping as shown in Figure 1. Steps can be included, omitted, and revisited as necessary. All steps are described below along with a real-world example. The host discovery step identifies that a device is accessible at a given **Internet Protocol (IP)** address; for a tester mapping a traditional or **Infrastructure-as-a-Service (IaaS)** network, this step ensures all devices are included in the test. The port scanning step identifies that a service is running and accessible at a given IP address on a given port; for a tester conducting vulnerability scans, this ensures all services will be included in the scans. Next, the service/version detection step classifies and identifies each listening service; this step provides information that, for example, could help a red teamer choose the correct exploit. The operating system detection step identifies the host operating system; this could help a compliance tester ensure all operating systems are at the correct patch level. Finally, the script scanning step provides detailed information about the service through additional active probing and/or analysis using scripts; these scripts include fuzzing, malware detection, specific version detection, and even exploitation of vulnerabilities in some cases [47].

Over the years there have been many tools that can be used for one or more of the many tasks required in network attack surface mapping. These tools vary in approach and purpose [18, 57]. For example, tools such as Shodan [54] and Censys [21] perform regular scans of the Internet according to a preset schedule, allowing users to query scans already made. Tools such as Nmap [49], Masscan [28], ZMap [23], and others allow users full control over all aspects of their scans but require those users to provide their own hardware and network connectivity to conduct the scan. Tools such as these require an adequate network infrastructure and run the risk of blowback from organizations who consider an unsolicited scan to be an attack. Table 1 summarizes the five commonly used tools discussed in this survey article. A detailed discussion of each tool follows.

2.1 Scanning Tools

2.1.1 Nmap. Arguably the most well-known network mapping tool, Nmap performs basic port scanning while extending this capability with service and version detection, operating system detection, and an extensible script engine that allows users to automate tools that make use of raw port scan data to interact further with the services for many purposes [37, 74]. Potential uses of the scripting engine include information gathering, password brute-force attempts, and some vulnerability identification [17, 35, 49, 54, 81]. Unlike Censys and Shodan, Nmap is an active scanner run by the user, so users must consider their legal and ethical responsibilities. Nmap is written in C++.

2.1.2 ZMap and ZGrab. ZMap was designed to conduct Internet-wide port scans. Through the use of such features as no rate limit ceiling, hand-crafted Ethernet frames, and stateless packet transmission, the author claims ZMap (written in the C language) can scan the entire IPv4 space approximately 1,300 times faster than Nmap [23]. Setting options properly for an asynchronous tool such as ZMap is essential to avoid overloading the source and target networks. ZGrab is the service detection counterpart to the ZMap port scanner and implements banner grabbing in support of the Censys project. Written in the Go programming language, ZGrab has a number of built-in modules to read banner information from common services like **Secure Shell (SSH)**, **File Transfer Protocol (FTP)**, **HyperText Transfer Protocol (HTTP)**, and many more. ZGrab is limited to identifying pre-programmed services but is user extensible for additional services [21].

Table 1. Network Attack Surface Mapping Tools

Tool Name	Purpose	Hosting	Target Networks	Notable Features
Nmap [49]	Port Scanning and Service Detection	User	Internal or Internet facing	Robust feature set, extensible script engine
ZMap/ZGrab [23]	Internet-wide Mapping	User	Internal or Internet facing	Fast, engine used by Censys
Masscan [28]	Internet-wide Port Scanning	User	Internal or Internet facing	Fast, limited banner grabbing
Shodan [54]	Internet-wide Mapping	Third Party	Internet facing	IoT focused, vulnerability detection
Censys [21]	Internet-wide Mapping	Third Party	Internet facing	Detailed banner info, direct database access

2.1.3 Masscan. Like ZMap, Masscan [28] was designed to scan the entire Internet, and it too uses asynchronous scanning to increase speed. Implemented in the C language, asynchronous scanners separate the threads that transmit packets from the threads that receive packets, allowing for higher speeds than Nmap. Masscan's command line options are modeled after Nmap for ease of use, and it incorporates limited banner detection capabilities [28]. Asynchronous tools can cause network congestion on networks at the source, target, or anywhere in between. As with ZMap, careful tuning is essential to ensure networks are not overwhelmed to the point where packets are lost and open ports are missed [30, 88].

2.1.4 Shodan. Known as the search engine of the **Internet of Things (IoT)**, Shodan is one of two tools we discuss that conduct scans on their own or at the request of users on their behalf, providing the results to all users. Outsourcing the actual scanning function to Shodan provides a number of advantages, particularly if the **Internet Service Provider (ISP)** of a researcher conducting scanning does not look kindly upon port scans. However, the user has less control over scans. Shodan only scans a subset of available ports. There is also less control over particular scan types and the version data that is returned. One major advantage of a tool like Shodan is that the data has already been collected and is waiting to be queried. Unlike Censys, the actual tools/implementation used by Shodan to conduct port scanning and banner grabbing are unknown [46, 82].

2.1.5 Censys. Censys, like Shodan, scans the Internet and provides the results for users to query. It uses the ZMap and ZGrab [23] scanning/banner grabbing engines discussed above. It is listed herein as a separate tool because, like Shodan, it allows researchers and testers to review scan data collected by someone else. While both Shodan and Censys can be accessed via API, Censys also facilitates access to its data via Google BigQuery. When Censys launched, it only scanned 14 ports; at the time of writing Censys has been expanded to scan over 2,500 ports [21]. However, this is only a fraction of all ports, so services running on less common ports are likely not included in the database.

2.2 Scanning Tools and MITRE ATT&CK

MITRE, a Federally Funded Research and Development Corporation, developed a framework of Adversary Tactics, Techniques, and Common Knowledge, commonly known as the ATT&CK framework. At the time of writing, the ATT&CK framework contained 196 techniques organized into 14 tactics. In general, the tactics described what is to be accomplished (e.g., Initial Access, Lateral Movement), while the techniques describe how the tactic will be accomplished (e.g., Phishing, Pass the Hash). The ATT&CK framework also describes how adversaries execute each technique, and thus is an excellent source of information for red teams seeking to emulate threats. This is relevant to this article because the framework indicates that multiple threat groups use the tools described above for Reconnaissance and Discovery. Red teams seeking to emulate these threats can consult the ATT&CK framework and referenced materials to guide them in emulating realistic threats [5].

3 HOST DISCOVERY

Host discovery is the act of determining whether or not a system exists at an IP address. By default, the port scanning tool Nmap will skip an IP address it has been asked to scan if it can't verify the host is up. It is possible to bypass host discovery, for example, to avoid skipping a host that is up but for some reason doesn't respond to discovery. It may also be skipped based on scan intention—for example, a scan of a single port over the entire Internet (a horizontal scan) would not benefit from host discovery. In the case of a per-host scan (a vertical scan), skipping host discovery will greatly slow down the scan, and testers would be better served to use Nmap options to customize host discovery to better detect the hard-to-find hosts [49].

3.1 Discovery Techniques

Hosts on the Internet can be public (intended to be accessed by anyone) or private (intended to be accessed only by a subset of users). If a host is intentionally exposed to the Internet, it can be argued that the host is public; however, it must not be assumed that the host maintainer wants all data on the host to be accessible to everyone. For example, it's obvious a bank wants its customers to be able to access their account (and only their account) from the Internet [49]. Just because the bank advertises services to the Internet, it cannot be assumed that every account on the system is meant to be accessed by the general public.

There are many potential techniques that could be used to discover a host. Nmap's default discovery technique is well documented and provides an excellent example for discussion. According to Nmap's documentation, the tool sends four packets to discover a host: **Internet Control Message Protocol (ICMP)** Echo, TCP SYN to port 443, TCP ACK to port 80, and ICMP Timestamp.

Nmap will send different packets in certain circumstances. For example, if the target is on the local network, Nmap will detect it using **Address Resolution Protocol (ARP)**. If Nmap is run in a Unix environment without root privileges, it sends standard TCP SYN packets to 443 and 80.

If an Nmap user wishes to avoid sending the traditional discovery packets either to avoid looking like an Nmap scan or to improve detection for devices that are configured to avoid it, they can configure Nmap to send different types of packets using command line options. There are options to send combinations of TCP SYN or ACK packets, UDP packets, **Stream Control Transmission Protocol (SCTP)** INIT packets, or various types of ICMP pings or IP packets.

Tools such as ZMap and Masscan are designed to scan large swaths of the Internet for open ports but can be used for host discovery [23, 28]. One can correctly assume that, if a port is listening on a host, then that host is active. The Metasploit framework, a tool used by red teams and threat actors alike to combine payloads with exploits, relies on Nmap for its host discovery capability [71].

3.2 Works Related to Host Discovery

Table 2 summarizes the works related to host discovery, which we discuss in detail below. The researchers behind these papers used host discovery to characterize networks (whether large Internet-facing networks or internal-only SDNs) and to discover the existence of hosts, whether they were traditional computers, IoT devices, or controllers for water/power. Papers had their limitations, such as limited access to networks, requiring manual intervention, and narrow scopes; but these works also had great accomplishments, such as identifying

Table 2. Works Related to Host Discovery

Related Work	Tool(s)	Purpose	Features	Limitations	Findings
Torabi et al. [82], 2020	Shodan	IoT Scanning Campaigns	Network telescope that detected scanning activity	Couldn't see behind gateways	Identified IOT botnets
Manzanares-Lopez et al. [53], 2019	Nmap	Nmap in SDNs	Implemented Nmap in an SDN controller module	No script scanning	Select Nmap functions implemented correctly
Bagyalakshmi et al. [7], 2018	Nmap	Traffic Analysis of Nmap scans of medical databases	Nmap to test and Wireshark to analyze	Only used discovery scans	Most servers accept ping packets
Kim et al. [41], 2018	Zmap Masscan Shodan	Internet-Wide Scanning	Cloud-backed custom scanning engine	No comparison of detection to Nmap	Comparable to ZMap
Bano et al. [8], 2018	ZMap	Host discovery	Defining host liveness	Does not include internal networks	Scanning at multiple OSI layers can improve host detection accuracy
Achleitner et al. [1], 2017	Nmap	SDN Deceiving Nmap	SDN-based system to deceive port scanners	Only works on SDN	Isolated malicious scanners before they could find vulnerable systems
Joshi and Singh [37], 2017	Nmap	Risk Management Framework	Demonstration of risk management framework on university network	Commands are manually run	Identified many potential vulnerabilities in the network
Zhou et al. [92], 2017	Nmap	ICS Scanning	Nmap NSE script to scan ICS	Limited to Schneider PLC	NSE script successfully scanned Schneider PLC
Myers et al. [63], 2015	Zmap Masscan	Internet-Wide Scanning	Scanning framework with primary/secondary scans and analysis.	Limited to Masscan, Unicornscan, and ZMap	Sending ICS payloads could be construed an attack

vulnerabilities, isolating malicious scanners, and categorizing IoT/ICS devices. Discovering that a host exists is the critical first step to many types of research, and these papers used a variety of tools and methods to do that.

Torabi et al. [82] used Shodan's host discovery capabilities to identify IoT devices participating in scanning campaigns. Shodan provided the type of device and the associated country, both of which helped provide additional insights for their research. Kim et al. [41] proposed an improved method for conducting Internet-wide scanning for vulnerable IoT devices, comparing their results to ZMap, Censys, and Shodan. Their scanner used IP address randomization in an attempt to avoid firewall blocks when scanning large blocks of IP addresses. Zhou et al. [92] used Shodan to discovery Schneider **Programmable Logic Controllers (PLCs)** before using their custom Nmap script to conduct a more thorough scan.

Manzanares-Lopez et al. [53] implemented ICMP, TCP, and UDP methods of host discovery in an SDN controller module as part of an effort to bring Nmap capability to an SDN. Achleitner et al. [1] developed a defensive system to deceive port scanners attempting to identify hosts on an SDN.

Bagyalakshmi et al. [7] conducted a variety of Nmap sweeps against medical databases with the intent of timing each sweep and also determining which databases supported the ping sweep. Joshi et al. [37] discussed the importance of having an accurate inventory of network assets, and they presented Nmap as a tool that can discover assets for that inventory. Myers et al. [63] presented a framework for scanning the Internet, discussing scanning methods, randomization, scan distribution, and other areas of consideration on this topic. Their method uniquely involved conducting host discovery by using ZMap, Masscan, or Unicornscan to scan for a particular port, identifying active hosts for a secondary scan.

Bano et al. [8] discuss "liveness" as a more robust definition of whether or not a host is up on the network. They use ZMap to scan at multiple levels.

3.3 Insights and Discussion

The importance of host discovery has changed over the years. Historically, as a tester, it has been beneficial to classify network targets as hosts. Once a tester identified the hosts, they could then be individually scanned using the techniques discussed later in this article. This makes sense in environments with web servers, database servers, and other servers—where each server served a particular purpose. In a cloud environment this is less relevant and may even be misleading, since a single IP address may not represent a single host. However, host discovery techniques can still be used to identify active IP addresses in a cloud environment; port scanning and other techniques can then be performed to find individual services to test. In any case, host discovery provides a way to map the attack surface at the IP level. It is up to the individual tester to determine how to use that information to achieve the objectives of a particular test.

4 PORT SCANNING

Whereas the goal of host discovery is to identify active hosts, port scanning is the act of identifying open ports, which indicate active services listening on the network. Vertically scanning tools like Nmap discover hosts first by default before executing a port scan. This is to save time—since a vertical scan involves many ports, it would be a waste of time to scan hosts that aren't known to be up. Conversely, horizontally scanning tools like ZMap do not conduct host discovery by default. This is because host discovery must send at least one ping or check at least one port on the host, so performing a separate host discovery would double the number of packets that needed to be sent per active host.

4.1 TCP Scanning

TCP is the most common protocol in use on the Internet today. An understanding of TCP's three-way handshake is essential to understanding some of the unique methods available to security researchers seeking information on TCP ports [32].

Every standard TCP connection starts with a three-way handshake. The packets used are SYN, ACK, and SYN-ACK—so named because of the flags that are set in each packet. The standard handshake starts as follows: (1) A server sends a SYN packet to a server it wishes to communicate with. (2) The destination server responds to the source server with a SYN-ACK packet. (3) The source server responds with an ACK packet, finalizing the connection so the computers can communicate freely. Scanners can often determine the status of a port without completing the entire TCP handshake. For example, if a scanner receives a SYN-ACK packet in response to a SYN packet, it knows the port is open. There is no need to send the ACK and finalize the connection. However, some of the scanners can be configured to conduct different scans [63].

Tools like ZMap and Masscan are designed to scan the entire Internet quickly. As such, these tools do not complete the TCP handshake and only listen for the initial reply from the target. Nmap can be configured for either Connect or SYN scans, as well as other scan types. Nmap can execute many different types of TCP scanning. Each type of scan has a particular purpose. The most common scan types are the SYN scan (which sends a SYN packet and evaluates the response) and the Connect scan (which mimics a typical TCP connection). Other scan types that set other packet flags are available to attempt to bypass firewalls or hide the identity of the scanning system [13, 83].

4.2 UDP Scanning

UDP is less common than TCP but still widely used, in particular for services such as the **Domain Name Service (DNS)** or **Simple Network Management Protocol (SNMP)**. Because UDP is a connectionless protocol with no inherent ability to confirm receipt of transmission, scanning for UDP services is a much greater challenge. While TCP SYN/Connect scanning can rely on the SYN-ACK packet to indicate a port is listening, UDP services have no equivalent. UDP scans are inverse in that the server responds with an ICMP error message if the port is unreachable (not running a service) and otherwise only responds if the probe is correctly structured for the listening service. Thus, a UDP scanner identifies open ports as those that return no response or a response other than ICMP-unreachable [45]. UDP services often have interesting characteristics; for example, in order for DNS to work properly, the source port used by a DNS server to issue a query becomes publicly accessible by design [51].

Nmap and ZMap allow UDP scanning. Masscan does not scan UDP ports. Nmap's service detection engine aids in UDP scanning by sending additional requests to services considered "open or filtered" to attempt to identify a service. If a reply is received to any of these probes, the port status is changed to "open." If no reply is given, the UDP protocol does not provide a definitive answer as to open or closed since the port may not be responding simply because the expected input was not given.

4.3 Most Common Ports

With 65,535 TCP ports and 65,535 UDP ports to scan, it might not always be feasible to scan the entire port range for every host. For example, Tor services close circuits by default when being scanned, greatly increasing the time needed to conduct an accurate scan of the dark web [79]. The objective of the scan will also drive the number of ports scanned, as well as the order and the timing. A red team looking to stealthily find an entry point into a network likely won't scan all these ports on all systems. However, someone conducting an exhaustive vulnerability scan with the promise to cover every possible point on the network attack surface will want to make sure they don't miss a single port [7].

For those not wanting to scan every single port, Nmap provides a prioritized port list, as a file called `nmap-services`. This file contains a number indicating how frequently each port would be expected to appear. This information was gathered from an Internet-wide port scan in 2008 as well as by asking various organizations for data on their internal network. The list in `nmap-services` accounts for both Internet-facing and internal networks combined and has been tweaked and updated but not significantly revised since its creation [48].

Nmap's default scanning configuration makes use of this data by scanning only the top 1,000 ports. This was an improvement added in 2008—previously, Nmap scanned all lower ports and named upper ports. By scanning the top 1,000 ports, Nmap ran faster and returned more open ports, so this was a significant improvement. Nmap has additional features that allow users to choose an arbitrary number of ports to scan. Finally, testers can direct Nmap to only scan ports with a frequency greater than a specified value [49].

4.4 Works Related to Port Scanning

Table 3 summarizes the works related to Port Scanning, which we discuss in detail below. The authors of these papers sometimes developed new tools or methods to improve the port scanning process Internet-wide or on unique network types like SDNs, or they used port scanning to collect data for their research or simulate attackers. Sometimes port scanning was conducted following host detection, but other times host detection was skipped in favor of a pure service detection approach. Several of these works proved the speed of large-scale or Internet-wide scanning could be increased, and the body of work as a whole demonstrated the wide impact port scanning has on security, not only because of the diversity of target types but also because of the diversity in the purpose for the research.

4.4.1 Internet of Things. Nashida et al. [30] analyzed the impact of horizontal scanning on IoT devices in a bid to minimize overwhelming the less powerful devices with a port scan. While Torabi et al. [82] used Shodan to identify IoT devices for their research and analyzed the port scans being conducted from compromised IoT devices being used as part of a botnet. Koroniotis et al. [44] used Nmap's port scanning capability to generate simulated IoT botnet traffic. They compared it to actual botnet traffic and used it to train a neural network. Sivanathan et al. [76] introduced a hierarchical scan method, whereby the most likely open ports were scanned, and subsequent ports were chosen for scanning based on the status of previously scanned ports. As discussed above, Kim et al. [41] made improvements to Internet-wide scanning capabilities, with an IoT focus. For port scanning, they compared their performance against multiple port scanners including ZMap, which was specifically designed for this type of scan. Soyer et al. [78] used ZMap as the basis for the port scanning capability in their IoT vulnerability scanning system called KISA. Markowsky and Markowsky [54] discussed the controversial Internet Census of 2012, where researchers created a botnet and used it to scan the Internet. They then provided three different scenarios scanning IoT devices using Shodan, Nmap, and Masscan. Ceron et al. [16] used

Table 3. Works Related to Port Scanning

Related Work	Tool(s)	Purpose	Features	Limitations	Findings
Izhikevich et al. [34], 2022	ZMap Censys	Internet-Wide Scanning	Developed predictive system for scanning	Not as effective against IPv6 or random host configurations	Found 92.5% of services using 1/131 bandwidth
Ceron et al. [16], 2020	Nmap	ICS Device Detection	Used Shodan to detect devices	Only finds known vulns based on version info	6% of 989 devices vulnerable
Everson and Cheng [26], 2020	Shodan	Attack Surface	Clustering attack surfaces	Only used port information	Hosts can be clustered to save time
Hashida et al. [30], 2020	N/A	Scanning Impact on IoT	Math model for network congestion	Model was not tested	Adjusting scanning speed improves IOT network scans
Steinebach et al. [79], 2020	Nmap	TOR Analysis	Hosted 20 relays on TOR network	Only saw a fraction of the TOR network	Identified top 10 requested onion services
Torabi et al. [82], 2020	Shodan	IoT Scanning Campaigns	Network telescope that detected scanning activity	Couldn't see behind gateways	Identified IOT botnets
Yuan et al. [88], 2020	Masscan	Scan detection	Port scanner in Go	No service detection or scripting	Faster than Nmap for internal networks
Koroniotis et al. [44], 2019	Nmap	Botnets	Used Nmap to simulate botnet traffic	Simulated hardware	Developed IoT Botnet traffic dataset
Manzanares-Lopez et al. [53], 2019	Nmap	Nmap in SDNs	Implemented Nmap in an SDN controller module	No script scanning	Select Nmap functions implemented correctly
Tang et al. [80], 2019	N/A	IOT Scanning behind NAT	Reverse proxy and custom scanning algorithm	Simulated hardware	New algorithm is more effective for low-bandwidth networks
Cabaj et al. [14], 2018	Nmap	SDN Mitigating Scanning	SDN detected unsuccessful connections	Focused on SYN scans only	Port scans detected by methodology
Kim et al. [41], 2018	ZMap Masscan Shodan	Internet-Wide Scanning	Cloud-backed custom scanning engine	No comparison of OS detection to Nmap	Comparable to ZMap
Sivanathan et al. [76], 2018	Nmap	IoT Classification	Used hierarchical scan to reduce ports probed to classify IoT devices	Only effective on devices for which there is existing information	Lightweight port scans can classify some IoT devices
Joshi and Singh [37], 2017	Nmap	Risk Management Framework	Demonstration of risk management framework on university network	Commands are manually run	Identified many potential vulnerabilities in the network
Soyer et al. [78], 2017	ZMap	IoT Vuln Scanning	ZMap/ZGrab-based IoT scanning system	Needs script to scan a certain service	Scanning architecture was in progress at time of writing.
Jicha et al. [36], 2016	Nmap ZMap Masscan	Combining Scanners	Scanning framework combining Masscan and Nmap	Parameters for Nmap scan not provided.	200-fold increase in speed compared to Nmap scan
Durumeric et al. [21], 2015	ZMap Censys	Censys	Censys, a scanning engine with results accessible online	Censys only scans certain ports.	Censys provides info on attack surfaces and vulnerabilities.
Markowsky and Markowsky [54], 2015	Nmap Masscan Shodan	IOT Vuln Detection	Internet Census of 2012 discussion and three scanning case studies	Ethics/legality limit scope of scans.	Internet Census of 2012 illegal; Shodan, Masscan, Nmap can identify vulnerable devices.
Tilemachos and Manifavas [81], 2015	Nmap	Automating Pen Tests	Automated penetration test system with Nmap, Metasploit, and Python	Limited to vulnerabilities that have been scripted.	Automated system compromise is possible.
Myers et al. [63], 2015	ZMap Masscan	Internet-Wide Scanning	Scanning framework with primary/secondary scans and analysis.	Limited to Masscan, Unicornscan, and ZMap	Sending ICS payloads could be construed as an attack
Kumar and Sudarsan [45], 2014	N/A	UDP Scanning	Used ARP poisoning to speed UDP scanning.	Only works on local network and would impact network traffic.	19000% speed improvement over traditional scanners.
Durumeric et al. [23], 2013	ZMap	ZMap Introduction	ZMap high-speed port scanner	Limited to IPv4.	ZMap can scan entire IPv4 space in less than 45 minutes.

Shodan to identify ICS/SCADA devices in the Netherlands. They determined that nearly 1,000 ICS devices were exposed to the Internet, and that about 6% of those devices had a known vulnerability.

4.4.2 Software Defined Networks. Manzanares-Lopez et al. [53] tackled the challenges of implementing Nmap in an SDN environment by creating a controller module that runs many Nmap functions. This module was accessible via a Northbound REST API, allowing authorized users to run many Nmap functions, including port scanning, without needing the actual Nmap application itself. Cabaj et al. [14] discussed port scanning in terms of new networks, such as 5G. They built a framework using SDN to successfully detect port scanning activity by looking for the rate of unsuccessful connections.

4.4.3 Others. Durumeric et al. [21, 23] introduced ZMap and Censys as described previously in this article. The Internet-wide scanning framework presented by Myers et al. [63] used ZMap, Masscan, and Unicornscan to conduct a scan of devices discovered to be active, scanning additional ports to gather more information. Izhikevich et al. [34] introduced GPS, an Internet-wide scanning framework designed to predict open ports to the end of reducing scanning traffic and the time required to conduct such scans. Everson and Cheng [26] presented a solution for red teams and blue teams conducting manual analysis/testing when presented with a large attack surface. By using host/port pairs obtained from Shodan port scans, they were able to cluster similar hosts based on their open ports to streamline the reconnaissance process and highlight outliers that could be indicative of vulnerabilities. Steinebach et al. [79] used Nmap and other tools to conduct port scans of services inside the Tor network. Yuan et al. [88] developed a new port scanner designed to scan the control network for the heavy ion accelerator in Lanzhou. Tang et al. [80] used a reverse proxy connection to an internal scanner that scans a large number of IoT devices behind **Network Address Translation (NAT)**. After Joshi and Singh [37] performed host detection during their risk assessment of a university network, they performed port scanning to begin the identification of potential vulnerabilities. Jicha et al. [36] provided a comparison of connection-oriented and connectionless scanners. They used a hybrid of Nmap and Masscan to improve scanning performance of a large number of ports and hosts. Kumar and Sudarsan [45] presented one of the few articles to discuss UDP scanning. They discussed the complications associated with UDP scanning and presented experimental evidence showing a dramatic increase in speed over traditional UDP scanning methods by using ARP poisoning. Tilemachos and Manifavas [81] used Nmap for the port scanning function of their automated penetration testing system. Once the target was identified using Nmap, they used Python and Metasploit to conduct testing.

4.5 Insights and Discussion

For a security test practitioner, ports are where the test begins. While low-level driver/hardware testing can start below the TCP/UDP stack, services listen on ports, and experience shows that organizations looking to secure their infrastructure rely on device/operating system manufacturers to secure at lower levels. To categorize services, a tester should choose the tool that best fits, and the papers surveyed reflect researchers doing the same: scans of Internet-facing systems were outsourced to cloud-based scanners like Shodan and Censys, whereas internal scans were performed by Nmap when customization was necessary or Masscan/ZMap when speed was needed. Knowing, for example, that cloud-based scanners don't scan every possible port or that Nmap's default configuration can get hung up on certain infrastructures is critical to a tester getting accurate results in a timely enough manner to meet test objectives. A port missed is a port left untested, so the importance of accurate, complete port scanning cannot be overstated.

5 SERVICE AND VERSION DETECTION

Service and version detection is the process of identifying the software behind the process listening on a given port [49]. By connecting to the service and, if needed, sending an expected initial transmission over the network, the response can be analyzed to determine the software that has opened the port on the target. In some cases the version can be determined as well.

There are several reasons service detection is an important step in network attack surface mapping. First, while the **Internet Assigned Numbers Authority (IANA)** provides a list of port/service pairs indicating which service is typically on each port, many services are configurable to any open port [85]. Second, just knowing the generic service type may not be enough. A quick review of the Metasploit Framework's exploits will show that exploits are generally tailored to a particular type of server. For example, an FTP exploit generally doesn't work

Table 4. Works Related to Service Detection

Related Work	Tool(s)	Purpose	Features	Limitations	Findings
Izhikevich et al. [33], 2021	ZMap Masscan Censys	Internet-Wide Scanning	Developed more efficient service detection tool	No way to truly measure accuracy	Significantly reduced time identifying services on rare ports
Steinebach et al. [79], 2020	Nmap	TOR Analysis	Hosted 20 relays on TOR network	Only saw a fraction of the TOR network	Identified top 10 requested onion services
Haque et al. [29], 2019	Nmap	Botnet Analysis	Data mined Nmap results to classify botnets	Limited mining and dependence on accurate data	Confirmed botnet, identified possible vulnerabilities
Manzanares-Lopez et al. [53], 2019	Nmap	Nmap in SDNs	Implemented Nmap an SDN controller module	No script scanning	Select Nmap functions implemented correctly
Morishita et al. [61], 2019	Nmap ZMap Censys	Honeypot detection	Port scans used to develop signatures honeypots	Many operators failed to respond	Discovered 20k trivially identifiable honeypots
Kim et al. [41], 2018	ZMap Masscan Shodan	Internet-Wide Scanning	Cloud-backed custom scanning engine	No comparison of OS detection to Nmap	Comparable to ZMap
Lee et al. [46], 2017	ZMap Shodan Censys	Scan detection	Method to detect port scans from Censys	Used simulated network	Detects horizontal scans
Soyer et al. [78], 2017	ZMap	IoT Vuln Scanning	ZMap/ZGrab-based IoT scanning system	Needs script to scan a certain service	Scanning architecture was in progress at time of writing.
Durumeric et al. [21], 2015	ZMap Censys	Censys	Censys, a scanning engine with results accessible online	Censys only scans certain ports.	Censys provides info on attack surfaces and vulnerabilities.

against all FTP servers, but only on a particular type [56]. Knowing the version is also important, since once a vulnerability is fixed, the exploit will no longer work. Knowing the version of a service will help the tester choose an exploit, and it will also help the network defender know what needs to be patched or monitored. Information about individual services can be combined to infer the purpose of the device as a whole, e.g., identifying honeypots [61].

Most network attack surface mapping tools do not have an inherent service detection capability. Nmap has a robust service detection capability based on probing and matching the responses to those probes with pre-established patterns. These patterns are in a configuration file called `nmap-service-probes`. Nmap first tries the "Null" probe by connecting to the port and waiting for a response. If no matching response is received, Nmap then sends probes appropriate for the port and protocol in question based on the extensive information in the configuration file. Through regular expression matching, Nmap attempts to determine the software, the version, and sometimes even additional details about the service from the response provided [49].

ZGrab has modules to enumerate about 20 services [2]. Since ZMap and ZGrab are the engines behind Censys, one can log into Censys and view the data collected. Together, ZMap and ZGrab collect data similar to an Nmap scan run with service detection and safe scripts. ZGrab can also be extended with additional modules to detect newer services [78]. Masscan can be configured to conduct "banner grabbing." While Masscan is primarily designed to scan the Internet for open ports, this banner grabbing feature allows some limited service detection capability for certain common protocols [63].

5.1 Works Related to Service and Version Detection

Table 4 summarizes the works related to Service and Version Detection, which we discuss in detail below. Many authors used service detection to conduct research and analysis of hosts, some in botnets, others on the Tor network, and others on SDNs. Technical contributions of these works include identification of honeypots and Tor-based onion services, with scope and depth being the predominant limitation. This is a common theme with attack surface mapping works: because the scope is so large, papers must limit the ports/services in the research or limit their research to a particular subject area in order to have an attainable goal and useful output.

Steinebach et al. [79] conducted service detection on the Tor network, overcoming the ephemeral nature of Tor services by checking them over multiple months. Results showed that software was often years out of date, though no similar data was provided for non-Tor services to compare. The services that were detected skewed toward anonymous and/or illicit marketplaces and associated support services. Haque et al. [29] proposed a framework that uses Nmap's service detection capabilities followed by some custom Java programs to analyze botnets using frequent itemsets and association rules. By using Nmap command line options that include service/version detection, operating system detection, and basic script scanning, they were able to gather enough information to support the analysis. Their framework was able to confirm the attack origin and also provide useful information about the associated command and control capability. Morishita et al. [61] used ZMap, Censys, and Nmap to develop a signature-based method for the detection of honeypots. Because honeypots are just simulations of real services, they don't always respond as expected by service detection tools and are thus easier to identify as honeypots. The authors shared their research so honeypot owners had the knowledge needed to modify their configurations to better hide the true intentions of their honeypots from would-be intruders. As part of their SDN controller module mentioned above, Manzanares-Lopez et al. [53] implemented the service and version detection capabilities of Nmap for an SDN environment. Izhikevich et al. [33] developed LZR to identify services with fewer probe attempts, thus greatly speeding the service detection aspect of network attack surface mapping.

Lee et al. [46] analyzed banner grabbing behavior to help them determine whether or not a system was being scanned by Shodan or Censys. They accomplished this by differentiating normal connection behavior from a scanner that tends to grab banners and disconnect abruptly or grab banners immediately following a SYN-only connection consistent with SYN scanners. Also, the authors were able to detect horizontal scanning by directing sensors to share information with each other about suspicious network activity. Soyer et al. [78] used ZGrab for their service detection, a good complement to their port scanning system described earlier (which uses ZMap). They extended ZGrab to test for additional protocols, with an emphasis toward IoT-relevant protocols. Their use of multiprocessing greatly hastened scans that involve large blocks of IP space. Durumeric et al. [21] integrated ZGrab into Censys to allow robust service detection for the subset of services it was capable of scanning. The authors discussed the use of **JavaScript Object Notation (JSON)** format and described the challenges associated with collecting and storing such a large amount of data for so many IP addresses. As with its open ports data, Censys presents service information via web interface, API, and Google BigQuery. In addition to host detection, port scanning, and OS fingerprinting, Kim et al.'s [41] Internet-wide scanning system incorporated service/version detection, which they compared to ZGrab. However, their system scans the entire range of TCP ports, whereas ZGrab (at the time of writing) scanned only 23 protocols.

5.2 Insights and Discussion

If host discovery is finding the building and port scanning is finding the doors, then service and version detection is analyzing those doors to get the details on their materials and the locks and other hardware that protect them. As several papers have highlighted, existing tools have shortfalls. Arguably the most important characteristic is extensibility, since even if a tool works perfectly now, a new or custom service may require a custom script. This may be why Nmap is so popular among testers, since its scripting engine is built for extensibility. (Details about script scanning can be found in Section 7.)

6 OS DETECTION

Just as service detection attempts to identify the software and version of a single service listening on a particular port, operating system detection attempts to identify the software and version of the operating system behind the host. The reasons for testers wanting to know this information for an operating system are similar as well: an exploit designed to work against a 32-bit Linux system will not necessarily work against a 64-bit Windows system without modification.

Nmap has the ability to detect the operating system of a device it scans by fingerprinting the TCP/IP stack [58]. Nmap has a large database of operating systems and their TCP/IP characteristics. If the proper option is specified

Table 5. Works Related to OS Detection

Related Work	Tool(s)	Purpose	Features	Limitations	Findings
Haque et al. [29], 2019	Nmap	Botnet Analysis	Data mined Nmap results to classify botnets	Limited mining and dependence on accurate data	Confirmed botnet, identified possible vulnerabilities
Koroniotis et al. [44], 2019	Nmap	Botnets	Used Nmap to simulate botnet traffic	Simulated hardware	Developed IoT Botnet traffic dataset
Manzanares-Lopez et al. [53], 2019	Nmap	Nmap in SDNs	Implemented Nmap in an SDN controller module	No script scanning	Select Nmap functions implemented correctly
Kim et al. [41], 2018	ZMap Masscan Shodan	Internet-Wide Scanning	Cloud-backed custom scanning engine	No comparison of OS detection to Nmap	Comparable to ZMap
Joshi and Singh [37], 2017	Nmap	Risk Management Framework	Demonstration of risk management framework on university network	Commands are manually run	Identified many potential vulnerabilities in the network
Shamsi et al. [73], 2014	NMap	Single packet OS detection	Detects OS with a single packet	Jitter and other network conditions	Hershel classification method is more accurate
Sarraute and Burroni [70], 2010	Nmap	OS Detection Neural Nets	Replaced Nmap's operating system detection with neural network-based algorithm.	Used Monte Carlo simulation for majority of hosts.	Correctly detected operating systems with high accuracy
Zhang et al. [89], 2009	Nmap	OS Detection State Vector Machine	Replaced Nmap's operating system detection with a state vector machine	High error rate for Cisco and Mac	The SVM correctly identifies many operating systems.
Medeiros et al. [59], 2007	Nmap	OS Detection ICS Neural Net	Replaced Nmap's operating system detection with a self-organizing map	Segmentation of the map left to future work	A contextual map based on Nmap's signatures grouping similar systems together.

at runtime, Nmap will send additional packets to each target, gather protocol metadata from the responses, and attempt a match with its database, reporting the percentage of confidence it has in its selection [29].

Nmap itself incorporates some limited machine learning in its operating system fingerprinting model [75]. For IPv4 tests, a simple scoring system is used where all the fingerprinted operating systems are compared with the scan results. The fingerprint with the highest score (most feature matches) wins. For IPv6, the LibLinear library is used to execute logistic regression to determine the operating system.

There have been several papers published on using machine learning/neural networks to improve operating system detection [41, 59, 70, 89]. OS detection lends itself to neural network classification because there are a limited number of operating systems Nmap detects. Nmap's OS detection database has 3,780 unique fingerprints based on responses to consistent probes, whereas the service detection probe list has 11,613 unique entries based on responses to several different possible probes [49].

6.1 Works Related to OS Detection

Table 5 summarizes the works related to OS detection, which we discuss in detail below. Machine learning was heavily featured in these papers, likely because the characteristics used to fingerprint operating systems lend themselves to clustering and neural networks. Other papers used OS information to improve botnet research or security assessments. Weaknesses in these papers centered around simulated hardware, error rates due to network condition or operating system type, and narrow focus.

6.1.1 Machine Learning. Medeiros et al. [59] used a self-organizing neural network to fill in the gaps of existing operating system detection capabilities. By using the existing Nmap signature database, the authors created a neural network that will better allow the identification of unknown devices that may not exactly match a fingerprint in the database. Their intent was to detect the operating system of unknown devices, like those used in industrial control networks, so that security administrators could apply the correct configuration standards and tests to secure the devices. The authors didn't provide any performance evaluation of their method. Zhang et al. [89] presented a solution to a problem inherent to most signature-based classification techniques: what to do when there is no signature to match. They developed a support vector machine based on an out-of-date Nmap signature file. They then scanned devices running newer operating systems to test their capability against "unknown" newer operating systems. The error rate was below 6% for Windows, Linux, and Solaris, but higher for Mac, other flavors of Linux, and Cisco devices. Sarraute and Burroni [70] used neural networks to replace Nmap's standard distance-based operating system detection. They noted that the current method could incorrectly report a more generic system as a specific one because of gaps in the data. Their process was unique because it used a three-level hierarchy of neural networks: one to determine if the operating system is relevant, a second to determine the operating system family, and then one of three neural networks as a third layer customized to each of three operating system families. This unique approach acknowledged that the responses from different operating system families differ significantly enough to complicate training and use of a single neural network to fit them all. The authors found their performance exceeded that of Nmap's method. Kim et al.'s [41] improvements to Internet-wide scanning capabilities included an OS fingerprinting module. Like Nmap, their method relied on TCP stack fingerprinting. Their capability incorporated machine learning using the k* classifier algorithm.

The research above indicates that machine learning can provide an advantage over more traditional methods, provided they are properly trained. This is especially true in the case of inexact matches of versions and other characteristics, where the fuzzier logic of machine learning has a better opportunity to identify a match when the data isn't exactly as expected.

6.1.2 Botnets. In addition to service detection, Haque et al. [29] used Nmap's OS detection capabilities to analyze botnets. By determining the operating system of an infected device, it became easier to determine if an operating system vulnerability is present that could have contributed to the device's compromise. Koroniotis et al. [44] used Nmap to generate attack traffic emulating an IoT botnet to train a neural network. Adding the OS detection caused Nmap to send additional packets, changing the behavior and thus the training of the neural network. The end result was a dataset that combined regular network traffic, IoT traffic, and the simulated attack traffic. Shamsi et al. [73] developed Hershel, a single-packet operating system classification framework. Hershel was able to adjust for jitter and other network issues inherent to scanning over the Internet in order to better classify systems with a single packet. This approach is especially useful for Internet-wide scanning, where OS detection can be particularly time-consuming.

6.1.3 Others. Manzanares-Lopez et al. [53] implemented OS Detection Capability in their SDN controller module, bringing one of Nmap's more advanced features to the SDN. Joshi and Singh [37] enabled OS detection when using Nmap to scan the network in support of their university network risk assessment. OS information provided information that will help determine identify out-of-date or otherwise vulnerable systems. Nmap was able to identify potentially vulnerable systems, which were then confirmed via penetration test with Metasploit.

6.2 Insights and Discussion

It tracks that most papers surveyed on the topic of OS detection involved improving it or using it in lab conditions. This is because the techniques used by popular scanners are confused by devices in between scanner and target, like firewalls, proxies, and even routers. It can be difficult for the scanner to know exactly what it is fingerprinting. This is especially true in a **Software-as-a-Service (SaaS)** environment, where the tester's concern is more with the services than the hosting platform.

7 SCRIPT SCANNING

At this point in the process, the tester should have been able to gather information on active hosts, open ports, basic service/version information, and host operating system. In some cases, it may be necessary to gather more information than basic banner grabbing or OS detection can provide. Script scanning is the next step in the process and can help gather that information.

Table 6. Works Related to Script Scanning

Related Work	Tool(s)	Purpose	Features	Limitations	Findings
Asaduzzaman et al. [4], 2020	Nmap	CMS Vuln Scanner	Checked Wordpress and Joomla	Only looks at plugins	Framework finds vulnerabilities
Chalvatzis et al. [17], 2019	Nmap	Vuln Scan Eval	Comparison of Nmap with two vulnerability scanners	Nmap not designed to be a vulnerability scanner	The vulnerability scanners outperformed Nmap
Redondo and Cuesta [67], 2019	Nmap	Nmap GUI	GUI and scripting language for Nmap	GUI requires server that can host web pages	GUI reduces complexity of Nmap
Rosa et al. [68], 2019	Nmap	PCOM SCADA security	Wireshark, Nmap, Metasploit, Snort for PCOM	Need more availability-focused protections	PCOM lacks security features
Seralathan et al. [72], 2018	Nmap	IoT Vuln Detection	Used Wireshark to analyze protocol, Nmap to test	Only covers a specific device	Unencrypted comms, brute-forcible RTSP
Yılmaz and Gönen [87], 2018	Nmap	ICS Protection	Script to scan PLCs	Limited to Siemens devices	Created rule to detect stop/start attacks.
Zhou et al. [92], 2017	Nmap	ICS Scanning	Nmap NSE script to scan ICS	Limited to Schneider PLC	NSE script successfully scanned Schneider PLC
Winn et al. [86], 2015	Nmap	ICS Honeypots	Low-cost ICS honeypot using a single PLC	No support for compressed or encrypted protocols.	Laptop/Raspberry Pi could simulate 75 platforms, while a PLC could simulate 5.

7.1 Nmap Scripting Engine

Nmap has a powerful scripting engine based on the Lua language [65]. Version 7.93 of Nmap has 604 scripts available for use. These scripts extend the capabilities of Nmap by allowing data collected by the baseline scanning engine to feed additional analysis and even gather additional information. The scripting engine provides Nmap users with automation capabilities so they can expand the results of their port scans. Scripts can send additional packets to a port after it has been identified as being open to determine the service configuration or other service-related information. For example, a script run against a file-sharing service might provide a list of file shares. A script run against a web service might provide more information about the website or its server. Other scripts can provide brute-force passwords, test for certain vulnerabilities, exploit service functionality to gather additional information, or do any number of other functions. Choosing the correct command line options so that Nmap runs the scripts appropriate to your situation is an important part of network attack surface mapping with Nmap [67].

Nmap scripts run at one of three different points in the Nmap scan: before the scan, during the script scanning phase, and after the scan. Scripts run before the scan can run checks to make sure the script is executed successfully. Scripts executed during the script scanning phase are designed to gather additional information after ports are open and services have been identified. Post-scan scripts allow for additional processing of data already discovered [49].

While there are many Nmap scripts to cover common protocols, sometimes a more specific use case requires a custom script to be authored, even if the protocol itself is in common use and has many supporting scripts. By using the Nmap Scripting Engine, researchers are able to build on the framework of Nmap and adapt it to many new tasks. The next subsection describes papers wherein researchers made full use of the Nmap Scripting Engine.

7.2 Works Related to Script Scanning

Table 6 summarizes the works related to Script Scanning, which we discuss in detail below. The authors of these papers developed scripts to extend the capabilities of Nmap, usually to scan or provide additional information about a particular target with specific characteristics.

7.2.1 Vulnerability Scanning. Chalvatzis et al. [17] considered the use of Nmap as a vulnerability scanner, comparing it with actual vulnerability scanners like Nessus and OpenVAS. Those authors soon discovered that Nmap's scripting engine, while highly capable and customizable, did not have built-in scripts to detect as many vulnerabililties as a bona fide vulnerability scanner. Asaduzzaman et al. [4] wrote their own Nmap script to check for vulnerabilities in **Content Management Systems (CMSs)**. Their script detected the type of CMS used and then probed for known vulnerable extensions. The authors evaluated their solution against Wordpress and Joomla. Seralathan et al. [72] used an existing Nmap script to find vulnerabilities in an IoT camera. The `rtsp-url-brute` was used to find the exact URL of the live video stream. Though the authors didn't need to customize the script, they did need to add additional URLs to the list Nmap used to conduct the brute-force attack. Once they identified the URL, they were able to view the video feed without authentication. **Industrial Control Systems (ICSs)**, a form of **Operational Technology (OT),** are less common than their **Information Technology (IT)** counterparts. While Nmap has a few OT-relevant scripts, the extensibility of Nmap's scripting engine is invaluable for more unique situations. Zhou et. al. [92] wrote a new script to detect and characterize **Programmable Logic Controllers (PLCs)**, which tend to use different network protocols than traditional devices. Yilmaz and Gönen [87] used a custom script to detect and interrogate PLCs, simulating an attack to evaluate detection capability. The `pcom-discover` script authored by Rosa et al. [68] allowed the authors to search a previously unexplored SCADA protocol for vulnerabilities.

7.2.2 *Others.*

Redondo and Cuesta [67] mentioned lack of NSE support as a limitation of the official ZenMap GUI. They provided an overview of some of the more useful NSE scripts and how they could be used to gather additional information about applications and specific devices. The authors ensured NSE capabilities were included in the NmapGUI they developed. Winn et al. [86] used custom Python scripts as well as a built-in NSE script to test a proxy honeypot they created. The NSE script used was `enip-info`, which returned information allowing the authors to fingerprint the targeted PLC. The fingerprints could then be checked for accuracy as compared to real devices, confirming the honeypot worked as accepted. The honeypot allowed simulation of a complex environment using only a single PLC.

7.3 Insights and Discussion

The surveyed papers followed the pattern of pushing Nmap's scripting engine to the limits of its capabilities and then extending it where necessary. Automation allows a test engineer to repeat tests across multiple targets, and testers will use or extend tools like Nmap but will even turn to Linux shell scripting when it is more useful to do so.

8 FIREWALL/IDS EVASION AND SPOOFING

Since port scanning and service detection could be a precursor to illicit activity, it's in the best interest of most organizations to detect it and, if necessary, block it. Unfortunately, such defensive measures can impede legitimate testers as well, making port scan results invalid and effectively hiding potential test targets from view. Nmap has a variety of options designed to evade firewalls and Intrusion Detection/Prevention Systems. While these techniques have varying success depending on the target's firewall configuration, they might help hide scans or conceal their origin [49].

There are several methods that Nmap users can use to evade firewalls and other detection/prevention mechanisms. These include tweaking the TTL value, MTU value, or checksum values; encoding the packet; tampering with packet headers; or changing the timing of the scan so as to lower suspicion. Since firewalls are also designed to let valid traffic through, it follows that it might be possible to configure a port scanning tool to send traffic close enough to legitimate traffic such that it can evade detection and/or blocking. Here, success or failure ultimately depends on the configuration of the firewall and **Intrusion Prevention System (IPS)** on the target system [47].

8.1 Detecting Port Scans

Because of the unique signatures presented by Nmap and other tools discussed earlier, it is feasible to configure an IPS to recognize the order, frequency, and types of packets and render an effective detection decision. For example, Nmap's Host Discovery scan uses a known pattern that is easily detectable. Also, while a horizontal scan may be less obvious, a vertical scan targeting a single organization will show a large number of SYN requests in a short period of time—another detectable pattern [15, 16, 93].

Neural networks are another way to detect port scans. These can be trained using existing tools, and when fed the appropriate variables representing data collected by a system during a potential scan, they can classify network activity as a scan when appropriate. There are many works that seek to improve detection with neural networks, but most of them use the same approach of training a network using existing network traffic with known intrusions that have been labeled so the network can learn to differentiate them from traditional traffic. In these cases, false-positive measurements are important to ensure that valid traffic is not mislabeled (and possibly blocked) as an attack [3, 20, 42, 66].

One method used to evade detection is distributing the port scan among several different source IP addresses. A system designed to identify scanning activity coming from one IP address might miss the activity in the interest of false-positive reduction. This can be partially countered with distributed sensors that cross-communicate to correlate scanning activity [19, 60].

8.2 Blocking Port Scans

A firewall is commonly used to block port scans. Firewalls can be configured to block packets from a certain combination of IP addresses and ports and/or a certain combination of IP addresses and ports. This provides great flexibility and allows for surgical blocks. In addition, firewalls can be configured to drop packets entirely (with no response), respond as if the port is closed, or provide a more customized response as required [27].

The inherent design of SDN provides additional opportunities for efficient port scan detection and blocking. When applied in an IPS, SDNs allow for blocking of port scans simply by discarding the flows associated with the scans [64].

8.3 Works Related to Firewall/Intrusion Detection System (IDS) Evasion

Table 7 summarizes the works related to Firewall/IDS Evasion, which we discuss in detail below. The majority of these works fell into two categories: evasion and counter-evasion. Several works used older datasets, and some provided evasion techniques that only worked in limited cases. As a whole, the works did well to illustrate the cat-and-mouse game that exists between detection and avoiding detection, a subset of the larger never-ending conflict between network attackers and defenders.

8.3.1 Machine Learning. Panchev et al. [66] discussed the use of neural networks and machine learning to detect port scan activity. Their activity focused on mobile devices, and the neural network was trained using patterns of attack traffic. Kim et al. [42] also used neural networks for attack detection. They trained their model using the KDD Cup 99 dataset—they used 10% of the dataset to train and the entire dataset to test, resulting in an accuracy rate of about 99%. Similarly, Dias et al. [20] used the KDD Cup 99 dataset and an **Artificial Neural Network (ANN)** to drive an IDS. Their key point was that signature-based IDS could fall quickly out of date, and an ANN-based system could adapt to novel attacks. Almiani et al. [3] used a Deep recurrent neural network to support an IDS. They used the NSL-KDD dataset to demonstrate their model, which consisted of a traffic processing engine and a Recurrent ANN classification engine. Their focus was on IoT DoS attacks. All these papers used datasets from 1999.

De Santis et al. [18] used data clustering and related techniques to determine which scanner was conducting the scans. They noted characteristics like the horizontal, single-port nature of ZMap and Censys scans as compared to the more intense scans from Shodan. Kelly et al. [39] explored the effectiveness of Artificial Intelligence when used to enhance both the attacker and the defender roles of a cyberattack against an SDN. They used Nmap scans for the “attacks,” and for the defender they determined which configurations were most effective at evading those.

8.3.2 Other Papers. Zitta et al. [93] analyzed the Suricata IDS tool as run on a Raspberry Pi to demonstrate its capability in a low-performance environment representative of an IoT network. By using Nmap and other tools,

Table 7. Works Related to Firewall/IDS Evasion

Related Work	Tool(s)	Purpose	Features	Limitations	Findings
Barbour et al. [9], 2021	N/A	Evading port scan detection	Evaded Zeek and Snort by generating traffic	Only works on local network hosts	Evades Zeek at rates up to 1M packets per second.
Almiani et al. [3], 2020	Nmap	RNN IoT IDS	Neural network	Older dataset	RNN works for IDS in real-time environments
Liao et al. [47], 2020	Nmap	Detecting Nmap	New rules to detect Nmap better	Only tested against Suricata	Improved detection even with IDS evasion
Kelly et al. [39], 2019	Nmap	SDN AI Defense	AI makes false network for compromised node	Only for SDNs	Deception in SDN can make recon more difficult
Manzanares-Lopez et al. [53], 2019	Nmap	Nmap in SDNs	Implemented Nmap in an SDN controller module	No script scanning	Select Nmap functions implemented correctly
Zitta et al. [93], 2019	Nmap	IOT IPS	Nmap and Metasploit pen testing Suricata IDS/IPS	Limited set of attack techniques	Rules added to Suricata detect obscure attacks
De Santis et al. [18], 2018	Shodan	Scanner classification	Hidden Markov Models	ZMap and Shodan only	HMMs classified scans accurately
Neu et al. [64], 2018	Nmap	Openflow IPS	Used Openflow to detect and block scans	Port scans on SDN only	No false positives, few false negatives
Zhang et al. [91], 2018	Nmap	ONIS Idle Scan	Updated Nmap idle scan for newer systems	Requires access to many IP addresses	Accuracy comparable to Nmap
Kim et al. [42], 2017	Nmap	DNN IDS	Detected attacks using Deep Neural Network	Older dataset	99% accuracy rating for KDD Cup 99 dataset
Lee et al. [46], 2017	ZMap Shodan Censys	Scan detection	Method to detect port scans from Censys	Used simulated network	Detects horizontal scans
Mazel et al. [57], 2017	Nmap ZMap Censys	Scanner Classification	Study of fifteen years of network traffic	Dataset limited to university traffic	Intent/behavior varied amongst different actors
Dias et al. [20], 2017	N/A	ANN IDS	Detects DoS, probe, remote-to-local, and user-to-root attacks	Older dataset	High sensitivity and specificity except for remote-to-local attacks
Cejka and Svepes [15], 2016	Nmap	Port Scan Detection	Detection method for vertical port scans	Not as effective against horizontal scans.	Source IP using over 50 destination ports is probably scanning.
Koch and Bestavros [43], 2016	N/A	Hide from scans	Combined port knocking and DNS to hide systems from scanners	Can be bypassed by knowing the domain name	Analysis showed tokens not feasible to brute force.
Marnerides and Mauthe [55], 2016	N/A	Botnet Traffic	Detailed analysis of Mariposa and Zeus botnets		Botnets have a scanning signature different from NMap
Deshpande et al. [19], 2015	Nmap	Cloud IDS w/ SNORT	Taxonomy of cloud attacks, including test results from a SNORT-based cloud IPS	Limited to port scans and flooding attacks	IDS detects attacks but can be overwhelmed by large amounts of traffic.
Panchev et al. [66], 2014	Nmap	Mobile Scan Detection	Decision trees and neural networks detected port scans on mobile devices.	Hardware limitations at time of publication.	Neural network was able to detect all types of port scans.
Mohamed et al. [60], 2013	N/A	Cloud IPS	Cloud-based IPS/IDS for IaaS with adaptive signatures	No test results presented.	Distributed IDS/IPS could detect attacks from outside and from inside.
Kang et al. [38], 2007	N/A	z-Scan technique to evade detection	Z-scan distributed scanning evaded Threshold Random Walk detection	Only effective against Positive Reward based methods	Z-scan was effective against TRW detection but less effective against hybrid detection.

the authors were able to demonstrate which attacks were detected by Suricata, describe why certain attacks were missed, and explain how to configure the software to detect and/or block the attacks. Liao et al. [47] developed IDS rules capable of detecting Nmap more effectively. They found that typical detection of Nmap scans falls off sharply when Nmap's evasion features are used, but that their new rules can still detect Nmap scans at 91.7% accuracy, even with evasion enabled.

Manzanares-Lopez et al. [53] implemented a firewall detection and evasion capability in their SDN module, which emulated many Nmap functions optimized for an SDN environment. Neu et al. [64] implemented ACK scans and pingless scanning to improve on signature- and anomaly-based IDS in their paper.

Lee et al. [46] detected abnormal patterns in TCP traffic to determine whether or not a system is being scanned by Shodan or Censys. They looked at both timing and behavior to make a determination. Mazel et al. [57] presented a study of 15 years of network traffic in an effort to understand Internet-wide scanning. This wide scope of data gave them the ability to provide useful insights into the intent and behavior of the actors behind the scanners.

Cejka and Svepes [15] developed a detection algorithm based on the characteristics unique to Nmap scans. Their technique focused on SYN scans and flags an excessively high number of ports scanned from a particular IP address. Marnerides and Mauthe [55] provided a detailed analysis of scan traffic from the Mariposa and Zeus botnets for the purposes of aiding network operators in early identification of this traffic. Mohamed et al. [60] detailed an IPS/IDS system designed for the cloud, specifically focusing on IaaS. Their solution could generate new signatures and also used multi-point detection to reduce false positives. Deshpande et al. [19] developed and tested an IPS configuration using SNORT in a private cloud environment, but their testing was limited to preventing port scanning and flooding attacks.

Zhang et al. [91] created ONIS, an alternative to the stealthy idle scan. Koch and Bestavros [43] combined port knocking with DNS functions to create a new way for systems to hide from port scanners, particularly horizontal scanners scanning the entire Internet for one port. Barbour et al. [9] developed a novel method to evade detection by IDS like Zeek and Snort. They were successful in evading detection by Zeek even up to 1 million packets per second. Kang et al. [38] created z-Scan, a method designed to alternate scanning traffic with connection attempts to known-active hosts, thus reducing the likelihood that Positive Reward detection methods like **Threshold Random Walk (TRW)** will detect the scanning activity.

8.4 Insights and Discussion

The surveyed works didn't fully cover some of the motivations for bypassing firewalls and evading detection. The criminal reasons are obvious, but for test engineers the answer is more nuanced. Though we've included firewall evasion and IDS evasion together in this article, each is used for different reasons. A test engineer conducting a penetration test wants to use firewall evasion to make sure all traffic from every test case reaches every target; this is the only way to ensure a thorough test. Alternately, a test engineer conducting a red team or purple team event may try IDS evasion techniques to evaluate the limits of the blue team's ability to detect them.

9 ETHICS AND LEGALITY

The act of discovering hosts and conducting the subsequent port scan has been the center of an ethical debate. Since port scanning is simply checking to see which parts of a host are responding, is that illegal or unethical? And at what point, if any, does it become so?[1] Kenneally and Dittrich [40] discuss IT-related ethics in their Menlo Report, where they extend the ethical considerations outlined in the 1979 Belmont Report, which discussed ethics in biomedical and behavioral research. They discussed the three principles from Belmont: Respect for Persons, Beneficence, and Justice, and they added a fourth: Respect for Law and Public Interest. While their work does not specifically focus on information security, they provide an excellent framework that can help researchers make ethical decisions before conducting network attack surface mapping for research purposes. Maaß et al. [50] reference the Menlo Report but also expand the discussion to cover current laws and ethics relevant to information security research.

In the United States, for example, there are many laws that could come into play when port scanning, and it is necessary to consider local, state, and federal laws before directing a scanner through, at, or from resources that are not owned by the person conducting the scans. The primary federal law that applies here is the **Computer**

[1]This article does not constitute legal advice, the authors are not attorneys, and anyone wishing to operate a port scanner is advised to consult a legal expert to avoid any questions of legality.

Fraud and Abuse Act (CFAA) of 1986, which has been updated numerous times in an attempt to keep up with changing technology and services [11, 31].

Smith et al. discussed the ethics and legality of port scanning, beginning with using the common analogy of checking doors to see if they will open [77]. They covered arguments for and against and note that the state of Georgia determined that port scanning was not illegal. It is worth mentioning that the target of a port scan is not the only entity a tester should be concerned with. Port scans could violate the terms of service of the tester's own ISP; in theory, legal ramifications for each and every network segment between and including scanner and target networks should be at least considered.

Ebersohn provided a review of case law on port scanning and ping flooding (sending a large number of pings to a server so quickly that it could impact its availability) [24]. They reviewed several laws, including United States laws at the federal and state level, a European law, and a law with jurisdiction in South Africa. Since many laws governing computer hacking have an "interference" clause, the author noted that port scanning was not considered to be illegal—port scans do not tend to cause any sort of interference or impact the availability of systems, except in extreme cases. Intent has a large part to play in determining whether a port scan is ethical or not. For example, a port scan done for the purposes of reconnaissance prior to an unethical act could itself be considered unethical, whereas a scan conducted for research purposes with no intent of personal gain would be more likely to be considered ethical. Competency of the scanning party should also be considered here—many tools contain settings to control the speed/flow of packets as well as what additional probes are run by scanners [28, 49]. It could be considered unethical for an individual to use a tool without the knowledge to properly control it, even with the best of intentions.

Ethics was a strong consideration in the design and ongoing usage of Censys [21]. Its designers conferred with their university network staff and local ISP and provided clues in their DNS records and via a website running on each scanner address. Most importantly, they honor exclusion requests for those who would rather not be scanned. Since they use standards-compliant network traffic and do not attempt to detect any vulnerabilities directly, they felt ethically sound in their decision to make this data available on the Censys website. Research has shown that Shodan has been actively used to facilitate IoT hacking by threat actors [6]. Perhaps the most significant advantage is that tools like Shodan and Censys turn what used to be active scanning into passive reconnaissance, allowing a would-be attacker to gain valuable information on a target without sending even a single packet directly to that target.

Legality or lack thereof aside, the research outlined above shows that port scanning has the potential to disrupt networks and/or the teams charged with defending those networks. Tools such as Censys, which provide easily verifiable information identifying the source and intent of their scans, reduce the effort and thus the potential harm. In terms of the Menlo report, this arguably tips the scales toward beneficence and helping to serve the public interest. Security researchers and testers would do well to follow their example to improve the ethical standing of their scans.

10 CONCLUSION

In this article we provided an overview of network attack surface mapping. We discussed the tools and techniques used and reviewed papers that cover the entire process, starting with literature surveys in this area before moving on to papers that covered each step in the attack surface mapping process.

Since the first TCP connection, there has been a need for tools to debug connections. With the growing popularity of cybersecurity, those tools have expanded with new capabilities to fulfill new roles. Armed with a thorough understanding of tools and their capabilities, researchers can find the tool that best fits their need. Even more important is understanding how to use the tools and all their features. An analogy to woodworking tools is apt here: an experienced professional with a hammer and chisel can produce precise, accurate results or even art; an amateur will at best produce passable results and at worst could produce inaccurate results or even an adverse impact.

For researchers or testers wanting to gain knowledge and experience in attack surface mapping, our research showed that Nmap is an ideal tool from which to learn the basics. The many available options let researchers explore different aspects of host discovery, port scanning, service detection, and more. A book specifically dedicated to Nmap fundamentals, like [74], may provide more direct instruction and examples for those learning to use the tool for the first time.

Attack surface mapping is a broad area, and there are many directions for future research. The scope of this survey was limited to the most common tools. One potential future research area would be to conduct a more comprehensive survey of the many open-source tools and scripts that have been developed by security testers and network professionals over the years.

Improving accuracy and increasing scanning speed would be of great interest to industry. Frequently, increasing accuracy reduces speed and vice versa. It may be possible to use clustering or neural networks to choose which ports to scan and subsequently which probes to send to those open ports more intelligently. It may also be possible to use neural networks to improve the interpretation of results from probes—either to increase the likelihood of a correct analysis or to more intelligently determine the next probe to send.

In addition to improving the performance and accuracy of scanning, it would be useful to extend current and proposed capabilities to additional scanning targets. While scanning of more typical computers and network devices is well researched, it would be valuable to adapt current tools and techniques to scan more atypical networks/devices such as automotive CANBUS networks and PLC devices. Fortunately, ZMap, Masscan, and Nmap are all open-source, providing ample opportunity for testers and researchers to customize them as they see fit.

Identifying a way to organize the massive amount of data that can be returned from a large port scan is another area for research. While clustering based on ports reduces some complexity, it may be more useful to cluster based on additional information, such as the banner information returned from service detection probes. Perhaps by analyzing all available data, a machine learning algorithm could determine the best way to group targets to reduce manual analysis time spent by humans.

REFERENCES

[1] Stefan Achleitner, Thomas F. La Porta, Patrick McDaniel, Shridatt Sugrim, Srikanth V. Krishnamurthy, and Ritu Chadha. 2017. Deceiving network reconnaissance using SDN-based virtual topologies. *IEEE Transactions on Network and Service Management* 14, 4 (2017), 1098–1112.

[2] David Adrian. 2020. ZGrab2 GitHub repository. GitHub. https://github.com/zmap/zgrab2

[3] Muder Almiani, Alia AbuGhazleh, Amer Al-Rahayfeh, Saleh Atiewi, and Abdul Razaque. 2020. Deep recurrent neural network for IoT intrusion detection system. *Simulation Modelling Practice and Theory* 101 (2020), 102031.

[4] Md Asaduzzaman, Proteeti Prova Rawshan, Nurun Nahar Liya, Muhmmad Nazrul Islam, and Nishith Kumar Dutta. 2020. A vulnerability detection framework for CMS using port scanning technique. In *International Conference on Cyber Security and Computer Science*. Springer, 128–139.

[5] MITRE ATT&CK. 2023. Mitre att&ck. https://attack.mitre.org

[6] Maria Bada and Ildiko Pete. 2020. An exploration of the cybercrime ecosystem around Shodan. In *2020 7th International Conference on Internet of Things: Systems, Management and Security (IOTSMS'20)*. IEEE, 1–8.

[7] G. Bagyalakshmi, G. Rajkumar, N. Arunkumar, M. Easwaran, Kumaravelu Narasimhan, V. Elamaran, Mario Solarte, Ivan Hernandez, and Gustavo Ramirez-Gonzalez. 2018. Network vulnerability analysis on brain Signal/Image databases using Nmap and Wireshark tools. *IEEE Access* 6 (2018), 57144–57151. https://doi.org/10.1109/ACCESS.2018.2872775

[8] Shehar Bano, Philipp Richter, Mobin Javed, Srikanth Sundaresan, Zakir Durumeric, Steven J. Murdoch, Richard Mortier, and Vern Paxson. 2018. Scanning the internet for liveness. *ACM SIGCOMM Computer Communication Review* 48, 2 (2018), 2–9.

[9] Graham Barbour, André McDonald, and Nenekazi Mkuzangwe. 2021. Evasion of port scan detection in Zeek and Snort and its mitigation. In *20th European Conference on Cyber Warfare and Security (ECCWS'21)*. Academic Conferences Inter Ltd., 25.

[10] Richard J. Barnett and Barry Irwin. 2008. Towards a taxonomy of network scanning techniques. In *Proceedings of the 2008 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries: Riding the Wave of Technology*. 1–7.

[11] Esha Bhandari and Rachel Goodman. 2017. Data journalism and the computer fraud and abuse act: Tips for moving forward in an uncertain landscape. In *Computation + Journalism Symposium*.

[12] Monowar H. Bhuyan, Dhruba Kr Bhattacharyya, and Jugal K. Kalita. 2011. Surveying port scans and their detection methodologies. *Computer Journal* 54, 10 (2011), 1565–1581.

[13] Elias Bou-Harb, Mourad Debbabi, and Chadi Assi. 2013. Cyber scanning: A comprehensive survey. *IEEE Communications Surveys & Tutorials* 16, 3 (2013), 1496–1519.

[14] Krzysztof Cabaj, Marcin Gregorczyk, Wojciech Mazurczyk, Piotr Nowakowski, and Piotr Żórawski. 2018. SDN-based mitigation of scanning attacks for the 5G Internet of radio light system. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*. 1–10.

[15] Tomas Cejka and Marek Svepes. 2016. Analysis of vertical scans discovered by naive detection. In *IFIP International Conference on Autonomous Infrastructure, Management and Security*. Springer, 165–169.

[16] Joao M. Ceron, Justyna J. Chromik, Jair Santanna, and Aiko Pras. 2020. Online discoverability and vulnerabilities of ICS/SCADA devices in the Netherlands. *arXiv preprint arXiv:2011.02019*.

[17] Ilias Chalvatzis, Dimitrios A. Karras, and Rallis C. Papademetriou. 2019. Evaluation of security vulnerability scanners for small and medium enterprises business networks resilience towards risk assessment. In *Proceedings of 2019 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA'19)*. 52–58. https://doi.org/10.1109/ICAICA.2019.8873438

[18] Giulia De Santis, Abdelkader Lahmadi, Jérôme François, and Olivier Festor. 2018. Internet-wide scanners classification using Gaussian mixture and hidden Markov models. In *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS'18)*. IEEE, 1–5.

[19] Prachi Deshpande, S. C. Sharma, and P. Sateesh Kumar. 2015. Security threats in cloud computing. In *International Conference on Computing, Communication & Automation*. IEEE, 632–636.

[20] L. P. Dias, Jés de Jesus Fiais Cerqueira, Karcius D. R. Assis, and Raul C. Almeida. 2017. Using artificial neural network in intrusion detection systems to computer networks. In *2017 9th Computer Science and Electronic Engineering (CEEC'17)*. IEEE, 145–150.

[21] Zakir Durumeric, David Adrian, Ariana Mirian, Michael Bailey, and J. Alex Halderman. 2015. A search engine backed by Internet-wide scanning. In *Proceedings of the ACM Conference on Computer and Communications Security*, Vol. 2015-Octob. 542–553. https://doi.org/10.1145/2810103.2813703. ISSN: 15437221.

[22] Zakir Durumeric, Michael Bailey, and J. Alex Halderman. 2014. An internet-wide view of internet-wide scanning. In *23rd USENIX Security Symposium (USENIX Security'14)*. 65–78.

[23] Zakir Durumeric, Eric Wustrow, and J. Alex Halderman. 2013. ZMap: Fast internet-wide scanning and its security applications. In *Proceedings of the 22nd USENIX Security Symposium*. 605–619.

[24] G. J. Ebersohn. 2003. Internet law: Port scanning and ping flooding: A legal perspective. *Tydskrif vir Hedendaagse Romeins-Hollandse* 66 (2003), 563.

[25] Simon Yusuf Enoch, Jin B. Hong, Mengmeng Ge, and Dong Seong Kim. 2020. Composite metrics for network security analysis. *arXiv*. https://doi.org/10.13052/jsn2445-9739.2017.007. tex.arxivid: 2007.03486.

[26] Douglas Everson and Long Cheng. 2020. Network attack surface simplification for red and blue teams. In *Proceedings of the 2020 IEEE Secure Development (SecDev'20)*. 74–80. https://doi.org/10.1109/SecDev45635.2020.00027

[27] Walter Fuertes and Patricio Zambrano. 2011. Alternative engine to detect and block port scan attacks using virtual network environments. *International Journal of Computer Science and Network Security* 11, 11 (2011), 14–23.

[28] Robert Graham. 2019. MASSCAN: Mass IP port scanner. GitHub. github.com.

[29] Afzalul Haque, Amrit Venkat Ayyar, and Sanjay Singh. 2019. A meta data mining framework for botnet analysis. *International Journal of Computers and Applications* 41, 5 (2019), 392–399. https://doi.org/10.1080/1206212X.2018.1442136

[30] Hiroaki Hashida, Yuichi Kawamoto, and Nei Kato. 2020. Impact of internet-wide scanning on IoT data communication in wireless LANs. In *2020 IEEE International Conference on Communications Workshops (ICC Workshops'20)*. IEEE, 1–6.

[31] Haifa Al Hosani, Maryam Yousef, Shaima Al Shouq, Farkhund Iqbal, and Djedjiga Mouheb. 2019. A comparative analysis of cyberbullying and cyberstalking laws in the UAE, US, UK and Canada. In *Proceedings of IEEE/ACS International Conference on Computer Systems and Applications (AICCSA'19)*, Vol. 2019-November. https://doi.org/10.1109/AICCSA47632.2019.9035368. ISSN: 21615330.

[32] Fu Hau Hsu, Yan Ling Hwang, Cheng Yu Tsai, Wei Tai Cai, Chia Hao Lee, and Kai Wei Chang. 2016. TRAP: A three-way handshake server for TCP connection establishment. *Applied Sciences (Switzerland)* 6, 11 (2016), 358. https://doi.org/10.3390/app6110358

[33] Liz Izhikevich, Renata Teixeira, and Zakir Durumeric. 2021. LZR: Identifying unexpected internet services. In *30th USENIX Security Symposium*.

[34] Liz Izhikevich, Renata Teixeira, and Zakir Durumeric. 2022. Predicting IPv4 services across all ports. In *Proceedings of the ACM SIGCOMM 2022 Conference*. 503–515.

[35] Trapti Jain and Nakul Jain. 2019. Framework for web application vulnerability discovery and mitigation by customizing rules through ModSecurity. In *2019 6th International Conference on Signal Processing and Integrated Networks (SPIN'19)*. 643–648. https://doi.org/10.1109/SPIN.2019.8711673

[36] R. Jicha, M. W. Patton, and H. Chen. 2016. Identifying devices across the IPv4 address space. In *2016 IEEE Conference on Intelligence and Security Informatics (ISI'16)*. 199–201. https://doi.org/10.1109/ISI.2016.7745469

[37] Chanchala Joshi and Umesh Kumar Singh. 2017. Information security risks management framework – A step towards mitigating security risks in university network. *Journal of Information Security and Applications* 35 (2017), 128–137. https://doi.org/10.1016/j.jisa.2017.06.006

[38] Min Gyung Kang, Juan Caballero, and Dawn Song. 2007. Distributed evasive scan techniques and countermeasures. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 157–174.

[39] Jonathan Kelly, Michael Delaus, Erik Hemberg, and Una May Orreilly. 2019. Adversarially adapting deceptive views and reconnaissance scans on a software defined network. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM'19)*. 49–54.

[40] Erin Kenneally and David Dittrich. 2012. The Menlo Report: Ethical principles guiding information and communication technology research. Available at SSRN 2445102.

[41] Hwankuk Kim, Taeun Kim, and Daeil Jang. 2018. An intelligent improvement of internet-wide scan engine for fast discovery of vulnerable IoT devices. *Symmetry* 10, 5 (2018), 151.

[42] Jin Kim, Nara Shin, Seung Yeon Jo, and Sang Hyun Kim. 2017. Method of intrusion detection using deep neural network. In *2017 IEEE International Conference on Big Data and Smart Computing (BigComp'17)*. IEEE, 313–316.

[43] William Koch and Azer Bestavros. 2016. PROVIDE: Hiding from automated network scans with proofs of identity. In *Proceedings of the 4th IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb'16)*. 66–71. https://doi.org/10.1109/HotWeb.2016.20

[44] Nickolaos Koroniotis, Nour Moustafa, Elena Sitnikova, and Benjamin Turnbull. 2019. Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-iot dataset. *Future Generation Computer Systems* 100 (2019), 779–796.

[45] Sumit Kumar and Sithu D. Sudarsan. 2014. An innovative UDP port scanning technique. *International Journal of Future Computer and Communication* 3, 6 (2014), 381. https://doi.org/10.7763/ijfcc.2014.v3.332

[46] Seungwoon Lee, Seung Hun Shin, and Byeong Hee Roh. 2017. Abnormal behavior-based detection of Shodan and Censys-like scanning. In *International Conference on Ubiquitous and Future Networks (ICUFN'17)*. 1048–1052. https://doi.org/10.1109/ICUFN.2017.7993960. ISSN: 21658536.

[47] Si Liao, Chenming Zhou, Yonghui Zhao, Zhiyu Zhang, Chengwei Zhang, Yayu Gao, and Guohui Zhong. 2020. A comprehensive detection approach of Nmap: Principles, rules and experiments. In *2020 International Conference on Cyber-enabled Distributed Computing and Knowledge Discovery (CyberC'20)*. IEEE, 64–71.

[48] Gordon Lyon. 2008. Nmap: Scanning the internet. *Defcon 16* (2008).

[49] Gordon Lyon. 2020. Nmap: The network mapper–Free security scanner. Nmap.org.

[50] Max Maaß, Henning Pridöhl, Dominik Herrmann, and Matthias Hollick. 2021. Best practices for notification studies for security and privacy issues on the Internet. In *The 16th International Conference on Availability, Reliability and Security*. 1–10.

[51] Keyu Man, Zhiyun Qian, Zhongjie Wang, Xiaofeng Zheng, Youjun Huang, and Haixin Duan. 2020. DNS cache poisoning attack reloaded: Revolutions with side channels. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 1337–1350.

[52] Nabanita Mandal and Sonali Jadhav. 2016. A survey on network security tools for open source. In *2016 IEEE International Conference on Current Trends in Advanced Computing (ICCTAC'16)*. IEEE, 1–6.

[53] Pilar Manzanares-Lopez, Juan Pedro Muñoz Gea, Josemaria Malgosa-Sanahuja, and Adrian Flores-de la Cruz. 2019. A virtualized infrastructure to offer network mapping functionality in SDN networks. *International Journal of Communication Systems* 32, 10 (2019), e3961.

[54] Linda Markowsky and George Markowsky. 2015. Scanning for vulnerable devices in the Internet of Things. In *Proceedings of the 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS'15)*, Vol. 1. 463–467. https://doi.org/10.1109/IDAACS.2015.7340779

[55] Angelos K. Marnerides and Andreas U. Mauthe. 2016. Analysis and characterisation of botnet scan traffic. In *2016 International Conference on Computing, Networking and Communications (ICNC'16)*. IEEE, 1–7.

[56] David Maynor. 2007. Metasploit toolkit for penetration testing, exploit development, and vulnerability research. In *Metasploit Toolkit for Penetration Testing, Exploit Development, and Vulnerability Research*. Elsevier. https://doi.org/10.1016/b978-1-59749-074-0.x5000-4

[57] Johan Mazel, Romain Fontugne, and Kensuke Fukuda. 2017. Profiling internet scanners: Spatiotemporal structures and measurement ethics. In *2017 Network Traffic Measurement and Analysis Conference (TMA'17)*. IEEE, 1–9.

[58] João Paulo S. Medeiros, Agostinho M. Brito, and Paulo S. Motta Pires. 2009. A data mining based analysis of Nmap operating system fingerprint database. In *Advances in Intelligent and Soft Computing*, Vol. 63 AISC. Springer, 1–8. https://doi.org/10.1007/978-3-642-04091-7_1. ISSN: 18675662.

[59] João Paulo S. Medeiros, Allison C. Da Cunha, Agostinho M. Brito, and Paulo S. Motta Pires. 2007. Automating security tests for industrial automation devices using neural networks. In *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'07)*. 772–775. https://doi.org/10.1109/EFTA.2007.4416854

[60] Hassani Mohamed, Lebbat Adil, Tallal Saida, and Medromi Hicham. 2013. A collaborative intrusion detection and prevention system in cloud computing. In *2013 Africon*. IEEE, 1–5.

[61] Shun Morishita, Takuya Hoizumi, Wataru Ueno, Rui Tanabe, Carlos Ganan, Michel J. G. Van Eeten, Katsunari Yoshioka, and Tsutomu Matsumoto. 2019. Detect me if you... Oh wait. An internet-wide view of self-revealing honeypots. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM'19)*. 134–143.

[62] Thomas Morris, Shengyi Pan, Jeremy Lewis, Jonathan Moorhead, Nicholas Younan, Roger King, Mark Freund, and Vahid Madani. 2011. Cybersecurity risk testing of substation phasor measurement units and phasor data concentrators. In *Proceedings of the 7th Annual Workshop on Cyber Security and Information Intelligence Research*. 1–1.

[63] David Myers, Ernest Foo, and Kenneth Radke. 2015. Internet-wide scanning taxonomy and framework. In *Conferences in Research and Practice in Information Technology Series*, Vol. 161. 61–65. ISSN: 14451336.

[64] Charles V. Neu, Cássio G. Tatsch, Roben C. Lunardi, Regio A. Michelin, Alex M. S. Orozco, and Avelino F. Zorzo. 2018. Lightweight IPS for port scan in OpenFlow SDN networks. In *IEEE/IFIP Network Operations and Management Symposium: Cognitive Management in a Cyber World (NOMS'18)*. 1–6. https://doi.org/10.1109/NOMS.2018.8406313

[65] Paulino Calderon Pale. 2015. Mastering the Nmap scripting engine. In *Mastering the Nmap Scripting Engine*. Packt Publishing Ltd., 95.

[66] Christo Panchev, Petar Dobrev, and James Nicholson. 2014. Detecting port scans against mobile devices with neural networks and decision trees. In *International Conference on Engineering Applications of Neural Networks*. Springer, 175–182.

[67] Jose Manuel Redondo and Daniel Cuesta. 2019. Towards improving productivity in nmap security audits. *Journal of Web Engineering* 18, 7 (2019), 539–578.

[68] Luis Rosa, Miguel Freitas, Sergey Mazo, Edmundo Monteiro, Tiago Cruz, and Paulo Simões. 2019. A comprehensive security analysis of a scada protocol: From OSINT to Mitigation. *IEEE Access* 7 (2019), 42156–42168.

[69] Shanto Roy, Nazia Sharmin, Jaime C. Acosta, Christopher Kiekintveld, and Aron Laszka. 2022. Survey and taxonomy of adversarial reconnaissance techniques. *ACM Computing Surveys (CSUR)* 55, 6 (2022), 1–38.

[70] Carlos Sarraute and Javier Burroni. 2010. Using neural networks to improve classical operating system fingerprinting techniques. *arXiv preprint arXiv:1006.1918*. http://arxiv.org/abs/1006.1918. tex.arxivid: 1006.1918.

[71] Offensive Security. 2021. Payloads - Metasploit unleashed. Offensive-security.com. https://www.offensive-security.com/metasploit-unleashed/payloads/

[72] Yogeesh Seralathan, Tae Tom Oh, Suyash Jadhav, Jonathan Myers, Jaehoon Paul Jeong, Young Ho Kim, and Jeong Neyo Kim. 2018. IoT security vulnerability: A case study of a Web camera. In *2018 20th International Conference on Advanced Communication Technology (ICACT'18)*. IEEE, 172–177.

[73] Zain Shamsi, Ankur Nandwani, Derek Leonard, and Dmitri Loguinov. 2014. Hershel: Single-packet os fingerprinting. *ACM SIGMETRICS Performance Evaluation Review* 42, 1 (2014), 195–206.

[74] David Shaw. 2015. *Nmap Essentials*. Packt Publishing Ltd.

[75] Alban Siffer. 2019. Machine learning in Nmap. https://blog.amossys.fr/nmap-ml.html

[76] Arunan Sivanathan, Hassan Habibi Gharakheili, and Vijay Sivaraman. 2018. Can we classify an iot device using TCP port scan? In *2018 IEEE International Conference on Information and Automation for Sustainability (ICIAfS'18)*. IEEE, 1–4.

[77] Bryan Smith, William Yurcik, and David Doss. 2002. Ethical hacking: The security justification redux. In *International Symposium on Technology and Society*. 374–379. https://doi.org/10.1109/istas.2002.1013840

[78] Onur Soyer, Kwan Young Park, Nomota Hiongun Kim, and Tae Soo Kim. 2017. An approach to fast protocol information retrieval from IoT systems. In *Lecture Notes in Electrical Engineering*, Vol. 448. Springer, 218–225. https://doi.org/10.1007/978-981-10-5041-1_38. ISSN: 18761119.

[79] Martin Steinebach, Marcel Schäfer, Alexander Karakuz, and Katharina Brandl. 2020. Detection and analysis of Tor onion services. *Journal of Cyber Security and Mobility* 9, 1 (2020), 141–174. https://doi.org/10.13052/JCSM2245-1439.915

[80] Fengxiao Tang, Yuichi Kawamoto, Nei Kato, Kazuto Yano, and Yoshinori Suzuki. 2019. Probe delay based adaptive port scanning for IoT devices with private IP address behind NAT. *IEEE Network* 34, 2 (2019), 195–201.

[81] Valkaniotis Tilemachos and Charalampos Manifavas. 2015. An automated network intrusion process and countermeasures. In *ACM International Conference Proceeding Series*, Vol. 01-03-Octo. 156–160. https://doi.org/10.1145/2801948.2802001

[82] Sadegh Torabi, Elias Bou-Harb, Chadi Assi, ElMouatez Billah Karbab, Amine Boukhtouta, and Mourad Debbabi. 2020. Inferring and investigating IoT-generated scanning campaigns targeting a large network telescope. *IEEE Transactions on Dependable and Secure Computing* (2020).

[83] Roman Trapickin, Oliver Gasser, and Johannes Naab. 2015. Who is scanning the internet? In *Proceedings of the Seminars Future Internet and Innovative Internet Technologies and Mobile Communications*, Vol. 81. 81–88. https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2015-09-1/NET-2015-09-1_11.pdf

[84] Eugene Tumoyan and Daria Kavchuk. 2012. The method of optimizing the automatic vulnerability validation. In *Proceedings of the 5th International Conference on Security of Information and Networks*. 75–78.

[85] Luc Vinet and Alexei Zhedanov. 2013. Internet assigned numbers authority (IANA) procedures for the management of the service name and transport protocol port number registry. *Journal of Chemical Information and Modeling* 53, 9 (2013), 1689–1699. http://arxiv.org/abs/1011.1669%0Ahttp://dx.doi.org/10.1088/1751-8113/44/8/085201. ISBN: 9788578110796 tex.arxivid: 1011.1669.

[86] Michael Winn, Mason Rice, Stephen Dunlap, Juan Lopez, and Barry Mullins. 2015. Constructing cost-effective and targetable industrial control system honeypots for production networks. *International Journal of Critical Infrastructure Protection* 10 (2015), 47–58.

[87] Ercan Nurcan Yılmaz and Serkan Gönen. 2018. Attack detection/prevention system against cyber attack in industrial control systems. *Computers & Security* 77 (2018), 94–105.

[88] Chao Yuan, Jinze Du, Min Yue, and Tao Ma. 2020. The design of large scale IP address and port scanning tool. *Sensors* 20, 16 (2020), 1–12. https://doi.org/10.3390/s20164423

[89] Bofeng Zhang, Tiezheng Zou, Yongjun Wang, and Baokang Zhang. 2009. Remote operation system detection base on machine learning. In *4th International Conference on Frontier of Computer Science and Technology (FCST'09)*. 539–542. https://doi.org/10.1109/FCST.2009.21

[90] Mengyuan Zhang, Lingyu Wang, Sushil Jajodia, and Anoop Singhal. 2018. Network attack surface: Lifting the concept of attack surface to the network level for evaluating networks' resilience against zero-day attacks. *IEEE Transactions on Dependable and Secure Computing* 18, 1 (2018), 310–324.

[91] Xu Zhang, Jeffrey Knockel, and Jedidiah R. Crandall. 2018. Onis: Inferring TCP/IP-based trust relationships completely off-path. In *IEEE Conference on Computer Communications (IEEE INFOCOM'18)*. IEEE, 2069–2077.

[92] Guangkai Zhou, Jun Bai, Bailing Wang, and Jia Song. 2017. A method of scanning industrial control system equipment. In *2nd International Conference on Mechatronics Engineering and Information Technology (ICMEIT'17)*. https://doi.org/10.2991/icmeit-17.2017.28

[93] Tomas Zitta, Marek Neruda, Lukas Vojtech, Martina Matejkova, Matej Jehlicka, Lukas Hach, and Jan Moravec. 2019. Penetration testing of intrusion detection and prevention system in low-performance embedded IoT device. In *Proceedings of the 2018 18th International Conference on Mechatronics - Mechatronika (ME'18)*. 1–5.

Received 17 November 2022; revised 28 October 2023; accepted 24 December 2023