Exploring Firewall Misconfigurations in the Wild

Insights From and Beyond "Beyond the Horizon: Uncovering Hosts and Services Behind Misconfigured Firewalls"

Marija Jovanovikj
majo00014@stud.uni-saarland.de

ABSTRACT

A firewall plays a critical role in protecting networks by acting as a barrier between the trusted and untrusted network, restricting which trusted hosts and services can be accessed from the untrusted network. However, its effectiveness heavily depends on correct configuration, such as precise rule definitions and proper rule ordering. In this report, we discuss common firewall rule misconfigurations, with a special focus on a firewall misconfiguration that exposes protected services to the public Internet. A simple careless exchange of the source and destination port numbers caused due to a human-error, allows an attacker to reach protected services that are meant to be unreachable from the public Internet. As a result, attackers can discover up to 12.60% more services, just by scanning from two popular source ports, namely ports 80 and 53. We end this report by discussing possible countermeasures and giving an overview of possible future directions.

1 INTRODUCTION

Since the emergence of the Internet, the number and diversity of interconnected networks have grown rapidly, ranging from large and independent Autonomous Systems (AS) up to small local area networks (LAN). Communication between hosts in networks relies on exchanging messages via protocols that define the format, how to send and receive data, the order of messages exchanged and the appropriate actions needed to be taken at each step. Initially, when the Internet was first designed, it provided no security guarantees. Instead, the Internet was seen as a way for honest and legitimate users to share benign information with each other. As attacks on early protocols emerged, the need to strengthen network communication became evident.

Despite building secure protocols, there was an additional need to restrict who can access which host and service in a network, or in other words restricting who can perform the protocols with the network hosts. Large network organizations often have several services that are meant for public access, as well as many private services meant for explicit internal usage. One of the ways to achieve this restriction goal is to use **firewalls**. Firewalls act as a barrier between the trusted internal network and the untrusted external network, often the Internet. The core idea with firewalls is to intercept each packet that is meant to either leave or enter the trusted network and analyze the packet's header fields. Based on the collected information from the packet, the firewall checks its defined list of policy rules and matches the packet with one of its policies and applies the required action.

A firewall searches for a matching policy rule starting from the top of the rule list. Once a matching rule is found, the remaining rules are ignored. Therefore, it is crucial to order the rules in a logical manner. Unfortunately, most of the firewall misconfigurations happen due to **bad ordering** of the policy rules. If an important rule that is meant to deny specific malicious traffic is not appropriately placed somewhere in the beginning of the policy list, there is a risk it will never be applied. Instead, the firewall will most likely match the malicious traffic with some general rules with wider accepting scopes that are placed in the beginning. To prevent unintentionally allowing malicious traffic, the best practice is to start with the most specific rules that allow or block specific traffic based on specific header fields and end with the most general ones, such as deny all traffic rule at the bottom.

Apart from correctly ordering rules, it is crucial to define the rules correctly in the first place, so they behave the intended way. Unfortunately, configuring rules is **manual work** prone to human-errors. Errors such as simply exchanging the values between the source and destination ports can become very costly, essentially making the rule behave exactly opposite from the intention. For example, if the intention is to allow any inbound connections to a publicly-facing web server to TCP port 80, exchanging the source port with the destination port will allow any inbound traffic to reach any service, not just the web service on the server, as long as the traffic comes from port 80. Such exchange of ports can be detected by attackers who constantly scan the IP address space for service discoveries. Carefully and creatively crafted scanning probes can easily discover protected services which are not meant for public access. Interestingly, just by scanning from source ports 80 and 53, attackers can discover up to 12.60% more services. This percentage can significantly increase if we test from more source ports. To combat human-errors which are inevitable, researchers look at ways to **automate** firewall configurations and perform checks.

The rest of the paper is structured as follows: In Section 2 we present some related work close to our topic of discussion. In Section 3 we present background information on firewalls and understand how they operate. In Section 4 we analyze different and common types of firewall rule misconfigurations. In Section 5 we focus on a firewall misconfiguration that exposes protected services to the Internet. In Section 6 we lead a brief discussion about the future of firewall management. Lastly, Section 7 covers the conclusion. The aim of this report is to explore the common errors in the wild and help to correct them and prevent them from future occurrences.

2 RELATED WORK

In this section, we discuss related papers that address the several different topics highlighted in this report. We first start off with IP scanning, then we move on to the root cause for firewall misconfigurations. We end this section by overviewing the different ways to detect and resolve the firewall misconfigurations.

2.1 IP Scanning

Durumeric et al. [8] examine the evolution of Internet-wide scanning enabled by high-speed tools like ZMap and Masscan, finding that most scanning traffic consists of horizontal scans targeting a single port across many IP addresses. They show that these scans often originate from single hosts or small address ranges in bulletproof hosting providers rather than botnets, and conclude that many network operators still fail to detect or block such easily identifiable activity. Raftopoulos et al. [12] analyze the impact of large-scale Internet scanning by studying a February 2011 IPv4-wide scan conducted by the Sality botnet and observing effects on hosts after the event. They find that only about 2% of scanned IP addresses responded to the probes, and roughly 8% of those responding hosts were later compromised. Wan et al. [18] study how scan origin affects host discovery by running IPv4 scans for HTTP, HTTPS, and SSH from seven geographically and topologically diverse networks. They find that a single vantage point can miss a nontrivial fraction of responsive hosts, up to 8.4% for HTTP, 4.6% for HTTPS, and 18.2% for SSH, due to factors such as filtering, routing policies, packet loss, and localized outages.

To solve the 32-bit IPv4 address exhaustion, IPv6 was introduced with 128-bit addresses. Exhaustive IPv4 scanning is feasible and already mastered with tools like Masscan, but the enormous IPv6 address space makes brute-force scanning impractical, requiring more sophisticated methods. Williams et al. [19] developed 6SENSE that uses reinforcement learning on known IPv6 addresses to iteratively generate and scan high-probability candidates. Song et al. [15] developed AddrMiner that applies pattern-based and learning techniques tailored to regions with no, few, or many seed addresses (known active IPv6 addresses used for generating new active target addresses). Hou et al. [9] developed 6Scan that dynamically adapts its asynchronous probing based on observed regional responses to improve hit rate and efficiency.

2.2 Root Causes for Firewall Misconfigurations

A firewall is a powerful tool for protecting a network's security, though when they are unintentionally misconfigured, their main purpose will be defeated. Firewalls may contain vulnerabilities that in the first place were shipped by the manufacturers themselves, often due to design flaws. Still, the occurrence of such vulnerabilities is relatively low. The main reasons for the vulnerabilities found in firewalls are due to human errors. Therefore, it is important to acknowledge them and appropriately fix them. Alfayyadh et al. [2] emphasize that **usability** plays a critical role in the proper functioning of a firewall, especially for personal firewalls that are managed by regular Internet users. Firewalls that provide vague information, as well as poor visibility of alerts and changes result in users configuring their personal firewalls poorly. The researchers conclude their work by stressing the importance of proper design during the development process. Voronkov et al. [17] further stressed the importance of keeping usability in mind when designing solutions. Even if the solution fixes the problems, its adoption depends on the usability of the actual product. Dietrich et al. [7] inspected what are the most common security misconfigurations caused by system operators'. Their findings show that in regards to firewalls, the most common misconfigurations are **disabled firewalls** and **faulty filter settings**.

2.3 Automated Solutions for Firewall Misconfigurations Detection

Manually detecting misconfigurations in complex environments is challenging, therefore the need for automated and formal method solutions is of great importance. Saâdaoui et al. [13] proposed a formal method for detecting and correcting misconfigurations in **multi-firewall environments**, using **Firewall Decision Diagrams (FDDs)** to model rule interactions. The method uses inference systems and a SAT solver to analyze configurations and correct existing rules, rather than adding new ones. Abbes et al. [1] presented a static analysis method that detects rule anomalies and evaluates the impact of adding or removing rules. They used a **Firewall Anomaly Tree (FAT)** structure where anomalies appear when multiple rules share the same path and supports dynamic updates allowing administrators to test changes during rule creation and placement. Bringhenti et al. [4] presented a methodology for **automatically** allocating packet filters and generating firewall rules that satisfy the security requirements. Specifically, they used a **formal correctness-by-construction MaxSMT** model that ensures the resulting configuration is both valid and optimal, minimizing the number of rules.

3 BACKGROUND ON FIREWALLS

A firewall is a network security system used for segregating the internal network from the external network, usually the Internet, by monitoring packets that pass through in each direction and examining their header fields, such as source and destination ports, IP addresses, application types. Some firewalls even allow to select predefined services to pass through the firewall. For example, the Palo Alto next generation firewalls [10] provide the **application-default** option in order to restrict applications by allowing them to pass only if they are using their appropriate default port. Applications running on unusual ports will be denied. Each examined packet is then matched against a policy rule. If a firewall does not find a matching rule, the default rule placed at the very bottom will be applied, which usually instructs to drop the packet. In general, when creating a firewall rule, three things must be declared such as the name of the rule, source and destination parameters. The rest of the features are optional and are used to further enhance and support the rule.

Source Address	Source Port	Destination Address	Destination Port	Action
192.168.1.2	80	10.10.10.20	22	Allow
10.10.0.0/24	Any	192.168.0.0/24	443	Deny
Any	Any	Any	Any	Deny

Table 1: Example of Firewall Rules Forming the Set of Firewall Policies

Table 1 shows how simple firewall rules in general look like. Reading the first firewall rule we learn that the host with IP address 192.168.1.2 is allowed to send packets from port 80 towards the host with IP address 10.10.10.20 on port 22. The second rule states that any host belonging to the subnet 10.10.0.0/24 is not allowed to send any packets to hosts belonging to the subnet 192.168.0.0/24 on port 443. The last rule is the default rule which is applied only if the packet does not match any of the previous two rules and by default it instructs the firewall to drop the packet.

Firewalls can be configured to operate as **stateless** or **stateful**. A stateless firewall treats every packet as independent, analyzing its header fields and matching them against its set of rules. That being said, in order to allow **bidirectional** communication between the internal and external network, **two separate rules** need to be defined. One rule will be for the internal towards external network communication and the other rule will be for the external towards internal network communication. On the other hand, stateful firewalls keep track of connections, also known as flows. Such stateful firewalls intercept a packet and check if it can be associated with an **established connection**. Thus, **only one rule** needs to be defined, as the responses coming back will be matched with the initially established connection.

Rule	Src IP/Net	Src Port	Dst IP/Net	Dst Port	Proto	Action
#1	$INT	*	$EXT	25	TCP	ACCEPT
#2	$EXT	25	$INT	*	TCP	ACCEPT
#3	$EXT	*	$WEB	80	TCP	ACCEPT
#4	$WEB	80	$EXT	*	TCP	ACCEPT
#5	*	*	*	*	*	DROP

Table 2: Stateless Firewall Rules Adapted From [11]

Table 2 shows a stateless firewall. To establish bidirectional communication between an internal host and some external host on port 25, rule 1 and rule 2 need to be defined separately. Rule 1 defines the direction from internal to external network, rule 2 defines from external to internal network. However, rule 2 allows any machine from the outside to access any machine on any port on the inside if it uses source port 25. Rather, what we would like to allow is only packets coming back from established connections to pass through, not any arbitrary ones.

Rule	State	Src IP/Net	Src Port	Dst IP/Net	Dst Port	Proto	Action
#1	NEW	$INT	*	$EXT	25	TCP	ACCEPT
#2	NEW	$EXT	*	$FTP	21	TCP	ACCEPT
#3	NEW	$EXT	*	$WEB	80	TCP	ACCEPT
#4	ESTABLISHED, RELATED	*	*	*	*	TCP	ACCEPT
#5	*	*	*	*	*	*	DROP

Table 3: Stateful Firewall Rules Adapted From [11]

Table 3 shows a stateful firewall. To establish a bidirectional communication initiated from an internal host to some external host on port 25, only rule 1 is required. Notice how rule 1 only specifies the internal to external direction. When a packet from outside arrives, it will be matched to an already established connection, so rule 4 will be applied. Thus, stateful firewalls overcome the weakness of stateless firewalls.

To sum up the background section, it is recommended to use stateful rules, as stateless rules are vulnerable by nature. If stateless rules are the only option, they can be strengthened by activating TCP flag filtering to block inbound connections and whitelist only necessary UDP traffic. Last but not least, regularly reviewing and updating the firewall rules is crucial for quicker misconfiguration detection and reducing the possible attack impact.

4 FIREWALL RULE CREATION MISCONFIGURATIONS

A firewall's effectiveness heavily depends on correct configuration such as precise rule definitions and proper rule ordering. All these tasks are done mostly manually which can lead to firewall misconfigurations due to human-errors. A properly functioning firewall should not have conflicting rules, each rule should serve a different purpose and the priorities of the rules should be modelled correctly. In this section, we discuss the most common rule errors that occur when operating a firewall.

4.1 Shadow Rule

A firewall rule is said to be shadowed when a previous rule matches all the packets that match this rule, such that the shadowed rule will never be activated [3]. This indicates the importance of correctly ordering the rules, starting from most specific to most general ones.

As an example taken from [16], let us have two rules defined as **rule 1: {tcp, *.*.*.*, any, 161.120.33.40, 80, accept}** and **rule 2: {tcp, 140.192.37.*, any, 161.120.33.40, 80, deny}**. The first rule allows any host on any port to connect to host with IP address 161.120.33.40 on port 80. The second rule denies any packets coming from hosts with IP addresses in the range 140.192.37.* directed towards host 161.120.33.40 on port 80. Now let the firewall receive a packet with the fields **{tcp, 140.192.37.61, 1234, 161.120.33.40, 80}**. The packet will be compared to the first rule, see that it matches the rule and the firewall will apply it, thus accepting the packet. The second rule is more specific and more suitable for the received packet, however due to the order, the rule will be ignored.

Such misconfigurations are not limited to situations with one single firewall. They may occur in situations where we have **multiple firewalls** conflicting with each other. For example, let us have such situation as depicted in Figure 1. Let us have a packet coming from the Internet towards domain D1.1. The packet has the following fields **{tcp, 189.124.32.60, 4000, 140.192.22.10, 25}**. The packet is first intercepted by firewall FW0 and based on its rule 3 it will allow the packet to move forward. The packet then reaches the firewall FW1, and the only matching rule is the default rule at the very bottom, resulting in the packet being denied. This turns out to be an anomaly where either firewall FW0 is allowing a stream that it should deny, or firewall FW1 is denying a stream that it should accept. When a rule in one firewall shadows a rule in another firewall, this is called an **inter-firewall shadowing anomaly**. The two firewalls can individually be configured correctly, however the issue lies when the two sets of rules are put together causing shadowing. This will lead to parts of the network suffering from the shadowing misconfiguration. Thus, network access control policy needs to work on a macro level, and the sum of all policies from several different devices need to work well together [3].

In case of a single firewall, it is important to review the order of the rules and make sure they are ordered starting from most specific to most general ones. On the other hand, when working with multiple firewalls things can quickly become complex. Therefore, researchers have been working on formal solution for the anomaly problem, developing safe algorithms to dynamically reconfigure firewalls based on emerging attacks [16].

4.2 Correlation Anomaly

Correlation anomaly happens when two rules that have different filtering actions become correlated. In other words, the first rule will match some packets that should be handled by the second rule and the second rule will match some packets that should be handled by the first rule. This causes the rules to partially overlap traffic handling, where the higher priority rule catches the overlapped traffic. Unlike shadow rules that are overridden all the time, a correlated rule is overridden only in specific cases, so only some of the time. This may reflect in a way that the policy will cause problems only to certain users or machines, while appearing to work for others. Unless the correlation anomaly is detected carefully, for example by using visualization tools, it may be assumed that there is a problem with the computer's configuration or some users' actions.

Correlation can also cause **total shadowing**, where multiple instances of correlation, several different rules, end up covering the entire scope of another rule that is positioned as lower priority. Furthermore, correlation also occurs in larger network architecture with multiple firewalls containing upstream (closer to the Internet/core network) and downstream (closer to a specific subnet) firewalls. Different subnets usually have different downstream policies which may contradict with the policies of upstream firewalls. For example, an upstream firewall may block certain traffic which a specific downstream firewall allows. Due to the policy of the upstream firewall, the subnet will not receive the packet. To avoid such correlation problems in this type of architecture, upstream firewalls should include additional rules that are specific to certain subnets, making both policies become aligned [3].

4.3 Redundant Rules

Redundant rules are two different rules that cover the same types of packets and take the same action. For example, a rule that blocks traffic from ports in the range 20 to 30 and another rule that blocks traffic from port 22. Since port 22 is within the range of the prior rule, the second rule is not needed and causes redundancy. This does not cause any harm, however it does slow down the overall performance and policy enforcement.

The interesting part is when we discuss redundancy in architectures with multiple firewalls. For example, a downstream firewall blocking traffic that an upstream firewall already blocks. At first glance, this may seem redundant and unnecessary, however it all depends on the **threat model**. If an organization also considers users in the inside zone as possible threat, such as an attacker that is already in the internal network, then this "redundancy" in the downstream firewall will prevent malicious traffic from the attacker.

4.4 Irrelevant Rules

A rule is considered irrelevant if a firewall cannot encounter a packet that makes the rule fire. For example, blocking packets from routes that have no physical connection to the network protected by the firewall. Another example is blocking traffic from outside hosts that anyway cannot reach internal hosts due to the internal hosts not being accessible from the Internet, such as not having port forwarding enabled. The key point to differentiate an irrelevant rule from a redundant is that a redundant rule duplicates the function of an existing rule ordered more higher, whereas an irrelevant rule tries to match traffic that is physically and logically impossible to occur. As with redundant rules, irrelevant rules also do not cause any room for exploitation, however it does impact the performance by slowing down policy enforcement, as the firewall needs to go through more rules before it finds the appropriate one.

5 FIREWALL MISCONFIGURATION EXPOSING PROTECTED SERVICES TO THE PUBLIC INTERNET

Flawed firewall rules can unintentionally allow inbound connections from special source ports to bypass the firewall. For example, there is a service on port P that is not supposed to be public and accept inbound traffic. An administrator may configure the firewall poorly, such that the protected service can be reached from outside exclusively using source port X. If the attacker sends scanning probes from random ports to port P, the attacker will not get any responses. However, if the attacker guesses X, the probe with source port X will successfully establish a connection to the service.

5.1 Methodology and Workflow

Deng et al. [6] studied this type of misconfiguration by performing three phases, as depicted in Figure 2.

Phase 1

The idea with phase 1 is to form a list of hosts that exclusively match the threat model. To achieve that, the researchers sent scanning probes from a specific designated port towards a target port and recorded the responsive hosts. Then, they scanned the previous responsive hosts from a different random high source port repetitively, and those that responded again were marked as irrelevant (goes against the threat model) and were removed. The repetitive scanning ensures low false positive rate and circumvents packet loss. The remaining hosts formed the candidate list.

Phase 2

The idea with phase 2 is to find out the candidates which truly serve the target service and have not gone offline during measurement. To achieve that, researchers sent application layer probes to the candidate list and collected the responses. Once a response was recorded, the responding host was not probed again. The above two steps were repeated until the response rate was below 1%. The collected responses formed the response list.

Phase 3

The idea with phase 3 is to confirm the desired false positive rate from the previous phases. To achieve that, researchers scanned the services from the response list from random high ports and removed the responsive ones as they were irrelevant to the threat model. Additionally, for every irrelevant host they removed their

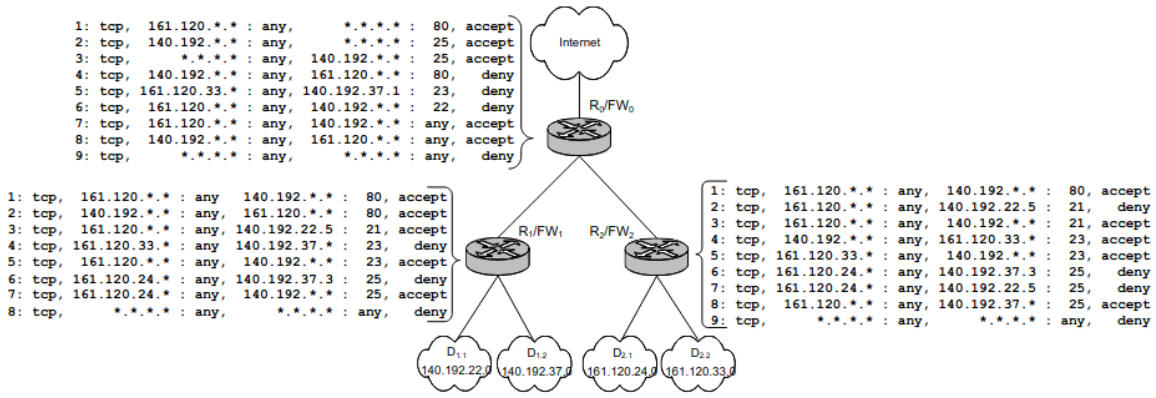

Figure 1: Network Topology with Multiple Firewalls Adapted From [16, Figure 2, p. 2].

previous recorded responses from phase 2. The above steps were repeated to ensure response rate was below 1%. The remaining validated responses were used to extract service information such as software versions and supported cryptographic algorithms.

Vantage points and control groups. The researchers used five vantage points (placed in the United States, Germany, Singapore, India, and Brazil, respectively) to increase the coverage and mitigate packet loss and region-based filtering. Each vantage point executed every step of the workflow, and the results (responsive hosts) were merged into a union set after each step. Furthermore, the researchers used control groups for verifying the hypothesis that affected services have higher security risks than public services due to false sense of security provided by firewalls. The control groups consisted of public services (unprotected by firewalls), whose hosts were randomly picked from the irrelevant hosts in Phase 1. These public services were probed by following the steps in Phase 2 of the workflow, except that connections were initiated from random high ports. Finally, they parsed the responses and compared the security risks of the affected services and public services.

5.2 Experiment Results

Deng et al. [6] selected the source port to be **port 80** for TCP-based services and **port 53** for UDP-based services. As destination ports the researchers chose the ports corresponding to 15 most common services: **SSH, Telnet, RDP, IPMI, SNMPv1, SNMPv2c ,SNMPv3, FTP, SMB, MySQL, MongoDB, HTTP, HTTPS, DNS and NTP**. Their results show that by testing only from those two source ports, they were able to reach up to **12.60%** more services. The ratios of affected services to public services were higher for UDP services than for TCP services, especially for the more popular protocols like DNS and NTP. The affected hosts were located in 221 countries and regions and belonged to 15,837 different ASes. The results suggest that ISPs are most prone to such firewall misconfigurations. Apart from the goal of discovering such private services due to the misconfigurations, the researchers also aimed to compare the **protection level** of the services meant to be private versus the protection level of the public services. Their hypothesis is that administrators rely more than they should on the firewall guarantees and they do not protect the private services as much as the public facing services. Table 4 shows the main results from their conducted experiment.

Protocol	Count	Key Findings
SSH	234,984	94.85% outdated OpenSSH versions.
Telnet	50,820	98.06% require login; 31 allow direct shell.
RDP	7,931	29.31% on EoL Windows.
IPMI	4,242	29.8% no auth; 1.5% anonymous; 8.7% null usernames.
SNMPv1	42,894	MAC address leakage.
SNMPv2c	36,753	Same issue as SNMPv1.
SNMPv3	465,033	Same issue as SNMPv1.
FTP	32,172	70.83% outdated/EoL software.
SMB	19,419	68.08% anonymous sessions; 63.44% EoL Windows.
MySQL	19,456	54.08% EoL MySQL.
MongoDB	338	76.33% public compromised; 2% private.
HTTP	222,539	Running internal sites.
HTTPS	193,630	Insecure certificates.
DNS	334,358	87.40% recursive; 63.67% respond to ANY.
NTP	824,389	2.70% respond to monlist.

Table 4: Summary of Uncovered Internal Services

Regarding Secure Shell (SSH), a cryptographically secured protocol for remote access to services, they uncovered **234,984 SSH services** behind misconfigured firewalls, where **OpenSSH** is the predominant (94.85%) vendor used. The uncovered services tend to run older OpenSSH versions indicating the possible usage of older operating systems and other outdated software. Moving on to Telnet, a protocol for remote access to services that does not use cryptographic protection, they uncovered **50,820 Telnet services**, and 98.06% of them ask for login when establishing connection. Only 31 of them directly give access to shells without any authentication. Regarding Remote Desktop Protocol (RDP), a protocol for remote secure manipulation of computer devices developed by Microsoft, **7,931 RDP services** were uncovered and 29.31% of the affected hosts run end-of-life Windows versions, whereas for the public services the percentage is only 9.70%. For the Intelligent Platform Management Interface (IPMI) protocol, a protocol

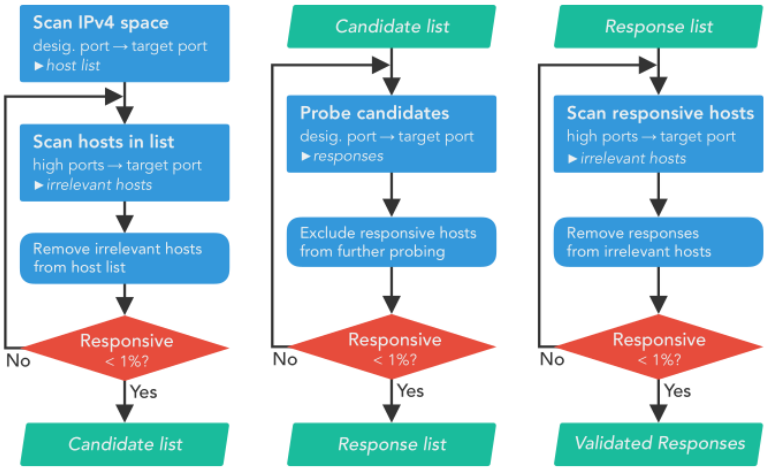

Figure 2: Three-Phase Workflow (from left to right), Adapted From [6, Figure 2, p. 4].

for remote hardware management, researchers uncovered **4,242 IPMI services** and probed them for their authentication configurations. Out of those, 29.8% do not use any form of authentication, 1.5% allow anonymous login via a low-privilege account without a password and finally 8.7% allow null usernames for login. Regarding Simple Network Management Protocol (SNMP), a protocol for monitoring and managing devices on IP networks, they uncovered **42,894 SNMPv1 services**, **36,753 SNMPv2c services** and **465,033 SNMPv3 services**. Interestingly, 163,827 devices give away their MAC addresses without any authentication, allowing for further fingerprinting and attacks. For File Transfer Protocol (FTP), a protocol for moving files between devices over TCP/IP networks, they uncovered **32,172 FTP services**. In total 70.83% of these servers run end-of-life or outdated software. For Server Message Block (SMB), a protocol for sharing files, printers, and other resources like serial ports across a network, researchers uncovered **19,419 SMB services**, out of which 68.08% allow anonymous sessions compared to public SMB services with only 24.42%. The percentage of private services running Windows is only 7.66%, whereas for public services it is 61.30%. In total 63.44% of private hosts run end-of-life Windows versions, while the percentage for the public services is only 44.40%. For MySQL, an open source relational database management system (RDBMS) used to store and manage data, they uncovered **19,456 MySQL services**, out of which 54.08% run end-of-life MySQL versions, compared to only 35.60% for the public services. For MongoDB, a NoSQL database used for storing structured, semistructured, and unstructured data, they uncovered **338 MongoDB services**. From the unsecured public MongoDB services, 76.33% have been compromised by ransomware, whereas only 2% of the private were affected. This indicates that the firewall bypass technique has not been actively exploited in the wild. For HyperText Transfer Protocol (HTTP) and HTTP Secure (HTTPS), protocols for web transfer of data, 222,539 and 193,630 services were uncovered, respectively. Most of them serve internal websites, use insecure certificates and run end-of-life versions. Significant percentage, 50.21% run small footprint servers which are likely embedded devices. For Domain Name System (DNS), a protocol for translating domain names into IP addresses, **334,358 DNS services** were uncovered and out of them 87.40% support recursive queries, compared to 27.99% for public services. The percentage of uncovered services responding to ANY queries is 63.67%, whereas for public services it is 15.37%. Finally, regarding Network Time Protocol (NTP), a protocol for synchronizing computer clocks across networks, **824,389 NTP services** were uncovered, out of which 2.70% respond to monlist queries, compared to 1.63% for public services.

6 DISCUSSION

Firewalls were introduced in the 1990s as the basic stateless packet filtering type, inspecting only the header fields of a packet, without looking into the content the packet is carrying. Over the years, firewalls evolved into stateful types enhanced with application-level security such as filtering traffic based on application data content, antivirus capabilities, intrusion detection and SSL decryption. Nowadays, we have machine learning (ML) powered firewalls that apply machine learning for analyzing traffic patterns and identifying anomalies. This allows firewalls to dynamically update to evolving threats. Despite the firewall evolution, still two concepts were preserved and those are the importance of correct rule creation and correct rule ordering. These two tasks remain to be manually done and are prone to human errors which eventually cause firewall misconfigurations.

To reduce the possible misconfigurations, firewalls should come with intuitive interface design to improve the usability of the tool and its correct application. Furthermore, as we are living in the Artificial Intelligence (AI) era, AI should be incorporated into the firewall management process, to help reduce the human errors. For example, when creating a rule, AI should make further suggestions to the administrators on how to improve and strengthen the policy rule and detect possible misconfigurations even before the rule is published. Additionally, once a rule is published, AI should help the administrator analyze the overall global state of the firewall policy

rules and detect possible ordering problems and suggest appropriate fixes. At last, AI should constantly collect network telemetry data and recommend policy updates for quick and dynamic adoption, with minimal negative impact. Additionally, AI can be used by researchers to craft interesting scanning probes for better discovery of vulnerabilities. It is important to note that inaccuracies may be spotted in AI models, therefore it is crucial to use AI as a helping tool and not completely rely on it. In crucial operations that deal with live traffic such as publishing rules, it is important for a human to perform cross checking and confirm the logic and validity of the suggestions. This is the point where human expertise and AI helping tools should work hand in hand. Furthermore, to make the AI solution more powerful and accurate, training needs to be performed on a narrow scope, such as the exact threat model. Sommer et al. [14] state that when choosing a specific machine-learning algorithm, it is important to be able to reason why this particular choice performs well for the given domain, instead of just focusing on the mathematical results obtained. If a solid argument cannot be given for the relation between the features and the attacks of interest, the findings and solutions will likely not be very beneficial. To sum up, when using AI to solve real world problems, the focus should not be solely on the numerical results, but on the domain specific results.

In regards to current research work, many published papers address the different types of firewall misconfigurations in the wild, as well as their root causes. However, most of the findings do not accurately depict the prevalence of such vulnerabilities. A valid reason for that could be the uncontrollable nature of networks, their volatility which can cause packet losses. To reduce this effect and ensure a wide scope of valid results, continuous iterative measurements over time can be useful. On the other hand, there are certain parameters that can be controlled during an experiment conduction and can increase the accuracy of the findings. Such controllable parameters are strategically placing sufficient number of distinct vantage points to achieve a higher coverage and reduce the impact of region-based filtering, scanning services not only for their well-known default ports, but also for other unusual ports leading to the increase of discovered services. To accurately depict the prevalence of misconfigurations, researchers can manipulate the source ports of the scanning probes to include as many different port numbers as possible, rather than focusing on few popular ones. Furthermore, to find out even more possible misconfigurations, researchers can focus on crafting scanning probes with unique combinations of not just source and destination ports, but also experimenting with different values for the packet flags. This way, the possibility of finding out a misconfiguration, possibly a very specific one which captures complex rule interactions, significantly increases. Last but not least, it is very important that researchers make use of their findings by disclosing the discovered misconfigurations to the vulnerable victims such as network operators, so that they can be fixed, the possible attack impact is reduced and the root causes are confirmed.

To wrap up the discussion, it is important to consider the direction for future work. As the Internet rapidly adopts IPv6, clearly the focus should be on protecting IPv6 services. As the IPv6 scanning methods become more sophisticated, the need for proper protection of IPv6 services increases. Unfortunately, findings show that the enablement of IPv6 traffic is not directly proportional to the deployment of the access control mechanisms for IPv6 devices, as we would expect. Czyz et al. [5] from their conducted research found out that devices such as routers are twice as reachable over IPv6 than for IPv4, particularly for protocols such as SSH, Telnet, SNMP and BGP. Furthermore, they found thousands of hosts that are only open over IPv6. They found out that IPv4 policies differ from IPv6 policies and IPv6 devices are more likely to respond to closed ports, compared to IPv4 devices that are more likely to silently drop packets. From their analysis, they show that about 26% of the hosts had at least one open application more on IPv6, whereas for IPv4 the percentage is 18%. On average, routers exposed 84% more services via IPv6, SSH was 166% more reachable, while servers exposed 12% more, including Telnet and SSH. Among servers without HTTP support, IPv6 exposure was even higher, with large increases for SMB (49%), Telnet (112%), and SNMP (343%). Therefore, future work should prioritize IPv6 deployment with equally robust access control and security policies to prevent widespread unintended service exposure.

7 CONCLUSION

A firewall is an effective and powerful security arsenal that requires careful and precise configuration. If a firewall is not properly configured, such as due to human errors, it may let malicious traffic pass through, as well as uncover private services. To strengthen the procedure of firewall configuration and management, researchers focus on coming up with automated and formal method solutions. Such solutions help detect the presence of anomalies, which can be very helpful when working with complex network topologies that include multiple firewalls. Furthermore, with the increase of IPv6 traffic, the need for stronger on par access control mechanisms for dual-stack devices is also increasing. Firewall vendors are increasingly incorporating AI solutions in their firewall products, with the goal to help administrators configure things as smooth as possible. As we have already seen, firewalls are targets to various types of attacks, therefore it is crucial for firewalls to dynamically adapt to the growing diversity of attacks and become resilient.

REFERENCES

[1] Tarek Abbes, Adel Bouhoula, and Michaël Rusinowitch. 2016. Detection of firewall configuration errors with updatable tree. *International Journal of Information Security* 15, 3 (2016), 301–317. https://doi.org/10.1007/s10207-015-0290-0

[2] Bander Alfayyadh, James Ponting, Mohammed Alzomai, and Audun Jøsang. 2010. Vulnerabilities in personal firewalls caused by poor security usability. In *2010 IEEE International Conference on Information Theory and Information Security*. 682–688. https://doi.org/10.1109/ICITIS.2010.5689490

[3] Michael Alicea and Izzat Alsmadi. 2021. Misconfiguration in Firewalls and Network Access Controls: Literature Review. *Future Internet* 13, 11 (2021). https://doi.org/10.3390/fi13110283

[4] Daniele Bringhenti, Guido Marchetto, Riccardo Sisto, Fulvio Valenza, and Jalolliddin Yusupov. 2023. Automated Firewall Configuration in Virtual Networks. *IEEE Transactions on Dependable and Secure Computing* 20, 2 (2023), 1559–1576. https://doi.org/10.1109/TDSC.2022.3160293

[5] J Czyz, M Luckie, M Allman, and M Bailey. 2016. Don't Forget to Lock the Back Door! A Characterization of IPv6 Network Security Policy. In *undefined*. https://doi.org/10.14722/ndss.2016.23047

[6] Qing Deng, Juefei Pu, Zhaowei Tan, Zhiyun Qian, and Srikanth V. Krishnamurthy. 2025. Beyond the Horizon: Uncovering Hosts and Services Behind Misconfigured Firewalls . In *2025 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 1770–1788. https://doi.org/10.1109/SP61157.2025.00164

[7] Constanze Dietrich, Katharina Krombholz, Kevin Borgolte, and Tobias Fiebig. 2018. Investigating System Operators' Perspective on Security Misconfigurations.

In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18)*. Association for Computing Machinery, New York, NY, USA, 1272–1289. https://doi.org/10.1145/3243734.3243794

[8] 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)*. USENIX Association, San Diego, CA, 65–78. https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/durumeric

[9] Bingnan Hou, Zhiping Cai, Kui Wu, Tao Yang, and Tongqing Zhou. 2023. 6Scan: A High-Efficiency Dynamic Internet-Wide IPv6 Scanner With Regional Encoding. *IEEE/ACM Trans. Netw.* 31, 4 (Jan. 2023), 1870–1885. https://doi.org/10.1109/TNET.2023.3233953

[10] Kiwi. 2023. Demystifying Application Default. (2023). https://live.paloaltonetworks.com/t5/community-blogs/demystifying-application-default/ba-p/546526 Accessed on November 13, 2025.

[11] Dr. Katharina Krombholz. 2024. Network Security. (2024). Slides from the Security Core Lecture, Saarland University.

[12] Elias Raftopoulos, Eduard Glatz, Xenofontas Dimitropoulos, and Alberto Dainotti. 2015. How Dangerous Is Internet Scanning?. In *Traffic Monitoring and Analysis*, Moritz Steiner, Pere Barlet-Ros, and Olivier Bonaventure (Eds.). Springer International Publishing, Cham, 158–172.

[13] Amina Saâdaoui, Nihel Ben Youssef Ben Souayeh, and Adel Bouhoula. 2017. A New FDD-Based Method for Distributed Firewall Misconfigurations Resolution. In *Information Systems*, Marinos Themistocleous and Vincenzo Morabito (Eds.). Springer International Publishing, Cham, 369–383.

[14] Robin Sommer and Vern Paxson. 2010. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. In *2010 IEEE Symposium on Security and Privacy*. 305–316. https://doi.org/10.1109/SP.2010.25

[15] Guanglei Song, Jiahai Yang, Lin He, Zhiliang Wang, Guo Li, Chenxin Duan, Yaozhong Liu, and Zhongxiang Sun. 2022. AddrMiner: A Comprehensive Global Active IPv6 Address Discovery System. In *2022 USENIX Annual Technical Conference (USENIX ATC 22)*. USENIX Association, Carlsbad, CA, 309–326. https://www.usenix.org/conference/atc22/presentation/song

[16] E.; Hamed H. Taibah, M.; Al-Shaer. 2004. Dynamic Response in Distributed Firewall Systems. *DePaul CTI Technical Report* (2004).

[17] Artem Voronkov, Leonardo Horn Iwaya, Leonardo A. Martucci, and Stefan Lindskog. 2017. Systematic Literature Review on Usability of Firewall Configuration. *ACM Comput. Surv.* 50, 6, Article 87 (Dec. 2017), 35 pages. https://doi.org/10.1145/3130876

[18] Gerry Wan, Liz Izhikevich, David Adrian, Katsunari Yoshioka, Ralph Holz, Christian Rossow, and Zakir Durumeric. 2020. On the origin of scanning: the impact of location on Internet-wide scans. In *IMC '20*. ACM SIGCOMM, 662–679. https://doi.org/10.1145/3419394.3424214 ACM Internet Measurement Conference, IMC 2020, IMC ; Conference date: 27-10-2020 Through 29-10-2020.

[19] Grant Williams, Mert Erdemir, Amanda Hsu, Shraddha Bhat, Abhishek Bhaskar, Frank Li, and Paul Pearce. 2024. 6Sense: Internet-Wide IPv6 Scanning and its Security Applications. In *33rd USENIX Security Symposium (USENIX Security 24)*. USENIX Association, Philadelphia, PA, 2281–2298. https://www.usenix.org/conference/usenixsecurity24/presentation/williams