Summary

In this presentation, we will outline the security architecture solutions provided by Secureini, a Moroccan company dedicated to protecting clients from threats and attacks. We will detail the technologies used and the step-by-step process followed to ensure a secure network. This includes measures to detect and prevent potential threats, as well as implementing secure remote access via VPN solutions. Our approach is designed to enhance the overall security posture of client systems through robust defensive strategies and proactive threat monitoring.

Secureini

Challenges

Company XYZ was initially using a traditional network setup, with services like FTP to store internal documents and Telnet for remote access.Recognizing the need for better security, XYZ approached our services to enhance their network protection.

To address their security concerns, we implemented a range of advanced technologies. I, A.Karim, as a cybersecurity consultant at securini, i was responsible for assessing the network's vulnerabilities and proposing the best solutions. We started by setting up a firewall using pfSense to control incoming and outgoing network traffic. For intrusion detection and prevention, we deployed Snort, which helps in identifying and blocking potential threats. Finally, for secure remote access, we implemented Twingate, ensuring encrypted and safe connections for employees working remotely.

This comprehensive security upgrade helped to significantly improve the protection of XYZ’s network.

Firewall Implementation with pfSense

To secure the **network**, we'll add a **firewall**. Since the company has a small security budget, instead of buying expensive hardware, we'll use **pfSense**, a trusted open-source firewall. **pfSense** is reliable, widely used, and backed by a large community that offers support and tutorials. It's a flexible solution with features like **VPN** and intrusion detection, making it a great choice for the company's needs. Regular updates and solid performance ensure that **pfSense** provides effective network protection.

Note: I previously configured OPNsense, but due to its smaller community engagement and the many problems I encountered without finding solutions, I decided to switch to pfSense for better support and stability.

Firewall Implementation with pfSense

add rules

After installing pfSense and configuring it with two interfaces,one for the local network (**LAN**) and one for the public network (**WAN**)—we will proceed to secure the network. Upon logging into the **pfSense** dashboard, we'll set up specific rules to protect XYZ's network from insecure connections.

As the first step, we'll replace the previously used insecure services with more secure alternatives. For example, we'll switch from using **FTP** to **SFTP** for secure file transfers and from Telnet to **SSH** for secure remote access.

In **pfSense**, we'll configure firewall rules to block non-trusted traffic and external threats, ensuring that the network is safeguarded from potential attacks.

Firewall Implementation with pfSense

add rules

As XYZ is an e-commerce platform with a self-hosted website on an internal server, we know that the majority of incoming traffic consists of **HTTP** (port 80), **HTTPS** (port 443), and **SSH** (port 22) for our remote developers and system administrators. Since there is no need for traffic on other ports, we will block all incoming traffic except for these essential ports.

Why?

This approach minimizes the attack surface by restricting access to only the necessary services. By blocking unnecessary ports, we reduce the potential entry points for malicious traffic and make the network more secure.

Firewall Implementation with pfSense

add rules

In **pfSense**, under **Rules** → **WAN**, we configured a rule to block all incoming and outgoing traffic:

- Action: **Block**
- Protocol: **Any**
- Source: **Any**
- Destination: **Any**
- Log: **Enable**

This configuration ensures that no unauthorized traffic enters or leaves the network, providing an extra layer of security and preventing potential threats.

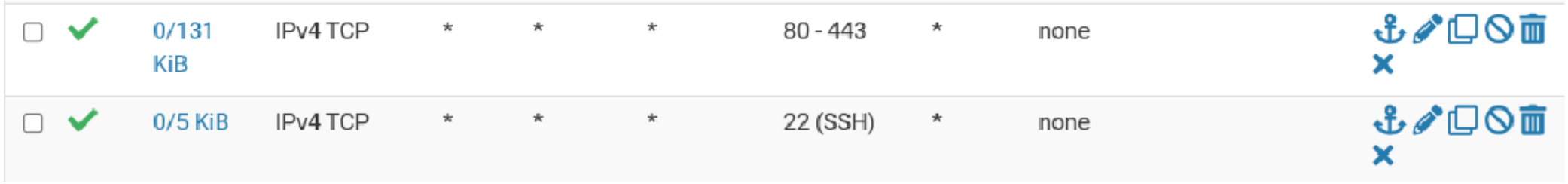
Firewall Implementation with pfSense

add rules

After blocking all traffic, we allowed only essential web services:

- **HTTP** & **HTTPS** (Web Traffic):
  - Action: **Pass**
  - Protocol: **TCP**
  - Destination Port: HTTP → **HTTPS**

For **SSH**, we added a similar rule but with a custom destination port to enhance security. This approach ensures only necessary traffic is permitted while reducing exposure to unauthorized access.



Firewall Implementation with pfSense

add rules

In the previous rules, we blocked all traffic and allowed only **HTTP**, **HTTPS**, and **SSH** by specifying destination ports.

Why?

When a user browses a website, their browser sends a request as follows:

- **Source:** client_ip:any_port (chosen by the browser)
- **Destination:** our_server_ip:80_or_443

This is why we apply our rule based on the destination port, ensuring that only traffic on the relevant ports (80, 443 for **HTTP** and **HTTPS**, and 22 for **SSH**) is allowed through.

Firewall Implementation with pfSense

add rules

To meet XYZ's request, we need to allow outgoing traffic for ports 80, 443, and 53 by creating the following rules:

☐	✓	0/0 B	IPv4 TCP/UDP	*	53 (DNS)	*	*	*	none
☐	✓	0/0 B	IPv4 TCP	*	80 - 443	*	*	*	none

In the previous rules, we specified destination ports, but now we will add the ports to the source ports.

Why?

Because these services are part of our network, so when someone requests them, the outgoing traffic originates from our network, making it the source of the traffic.

Firewall Implementation with pfSense

add rules

Since we've replaced Telnet, **FTP**, and **SMB** with more secure alternatives, we need to block these insecure services.
The process is the same as before: add block rules. The only difference is that we will adjust either the destination or source ports,
but the result will be the same, blocking those insecure protocols effectively.

☐	✗	0/0 B	IPv4 TCP	*	*	*	445 (MS DS)	*	none
☐	✗	0/0 B	IPv4 TCP	*	*	*	21 (FTP)	*	none
☐	✗	0/0 B	IPv4 TCP	*	*	*	23 (Telnet)	*	none
☐	✗	0/0 B	IPv4 TCP	*	*	*	135 - 139	*	none

Firewall Implementation with pfSense

add rules

Firewall / Rules / WAN

Floating | WAN | LAN

Rules (Drag to Change Order)

	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
✗	0/0 B	IPv4 TCP	*	*	*	23 (Telnet)	*	none			
✗	0/0 B	IPv4 TCP	*	*	*	135 - 139	*	none			
✗	0/0 B	IPv4 TCP	*	*	*	6881 - 6889	*	none			
✗	0/0 B	IPv4 TCP	*	*	*	445 (MS DS)	*	none			
✗	0/0 B	IPv4 TCP	*	*	*	21 (FTP)	*	none			
✓	0/0 B	IPv4 TCP/UDP	*	53 (DNS)	*	*	*	none			
✓	0/0 B	IPv4 TCP	*	80 - 443	*	*	*	none			
✓	0/131 KiB	IPv4 TCP	*	*	*	80 - 443	*	none			
✓	0/5 KiB	IPv4 TCP	*	*	*	22 (SSH)	*	none			
✗	0/42 KiB	IPv4 *	*	*	*	*	*	none			

Add | Add | Delete | Toggle | Copy | Save | Separator

Firewall Implementation with pfSense

IDS/IPS Implementation with Snort

To enhance security, we implemented an Intrusion Detection System (**IDS**) and Intrusion Prevention System (**IPS**) using Snort in **pfSense**. These tools help detect and prevent attacks in real-time.

Why SNORT?

Snort is open-source, easy to use, and backed by a large community that provides extensive support and solutions. Its wide range of tutorials and active development make it a reliable choice for intrusion detection and prevention.

Note: I previously configured Suricata in OPNsense, due to a technical problem that i didn't find solution for it, I decided to switch to pfSense for better support and stability.

IDS/IPS Implementation with Snort

Install & Configure SNORT

To install and configure Snort in pfSense, start by navigating to **System → Package Manager → Available Packages** search for "**Snort**," and click Install. Once installed, access the **Snort** panel via **Services → Snort**. From there, configure an interface to monitor traffic typically **WAN** for external threats. Enable the interface, save the settings, and proceed to configure rules and categories. **Snort** provides predefined rulesets, such as **emerging-scan.rules** for detecting port scans, which can be enabled for real-time threat detection. Additional custom rules can also be added to tailor the intrusion detection and prevention system to specific security needs.

IDS/IPS Implementation with Snort

Install & Configure SNORT

After installing Snort and the emerging-scan.rules ruleset, activate key rules to detect threats like port scanning, a common first step in attacks. Go to **Services → Snort**, select **WAN**, and enable the emerging-scan.rules under **WAN Categories**. Next, in WAN Rules, enable specific rules to detect port scanning, especially for tools like Nmap. Simply enable the rules and save changes.

Since SSH is in use, it becomes a potential target for attackers. To enhance security, we create a rule to detect brute-force attacks or multiple failed login attempts and add it to **custom.rules** under **WAN Rules**.

IDS/IPS Implementation with Snort

Implementing Twingate VPN

To improve network security, we implemented **Twingate** as our **VPN** solution. **Twingate** ensures secure remote access to our network while keeping it safe from external threats.

Why Twingate?

Twingate is a modern, easy to set-up solution that provides strong security for remote connections. It allows users to access the network safely, even from mobile devices, without compromising privacy. Its simple setup and reliable performance make it a great choice for secure connections, and it has been recognized and acknowledged by the tech community for its effectiveness.

Note: I had previously configured OpenVPN in pfSense, but due to a technical issue I couldn't resolve, I decided to switch to Twingate for its easier setup and enhanced security.

Implementing Twingate VPN

Using **Twingate** keeps our network accessible from anywhere, at any time, and from any device, whether it’s a laptop or mobile phone. **Twingate** provides a mobile app that ensures we stay connected to our network securely, even while on the go. This flexibility allows employees to access company resources remotely with ease and confidence, whether they’re working from home, traveling, or in a different office location.

Note: I had previously configured OpenVPN in pfSense, but due to a technical issue I couldn't resolve, I decided to switch to Twingate for its easier setup and enhanced security.

Implementing Twingate VPN


THANK YOU
FOR YOUR
ATTENTION