

Wazuh SIEM

HOME LAB SECURITY MONITORING

2026

Prepared by:
Daria Kurnosova

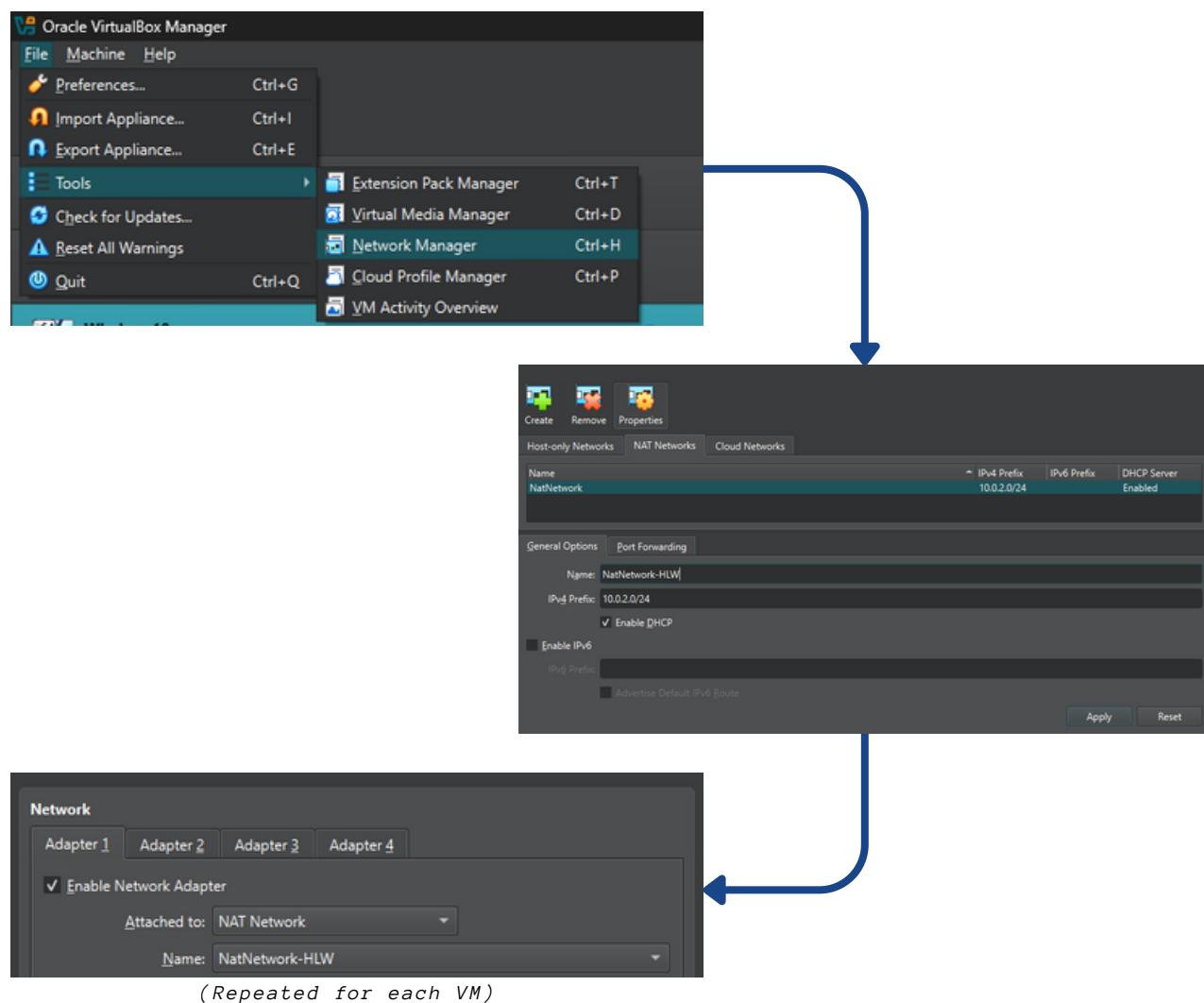
dariakurnosova.contact@gmail.com

Cracow, Poland

The purpose of this report is to guide you through my **Home Lab**. The idea lies in showing my practical knowledge of working with **Wazuh**, **Nmap**, **Hydra**, and different operating systems. I worked specifically with **Kali Linux**, **Ubuntu Linux**, and **Windows 10** here. So, I'm actually going to 'Play Hacker.' Windows 10 is the **victim**, Kali is the **attacker**, and Ubuntu is the **server**.

*Actions speak louder than words, though.
LET'S GET TO WORK! :)*

I. First of all, I set up a **NAT Network**, so my Virtual Machines could download updates and talk to each other.



II. The next step was to configure the Virtual Machines. So, I started with **Windows 10**, which is our “Victim” for this lab. I checked the IP Address and disabled the firewall.

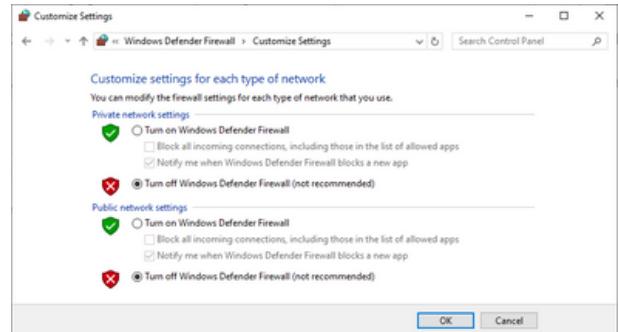
```
Microsoft Windows [Version 10.0.19045.3803]
(c) Microsoft Corporation. All rights reserved.

C:\Users\dkmic>ipconfig

Windows IP Configuration

Ethernet adapter Ethernet:

Connection-specific DNS Suffix . : lan
Link-local IPv6 Address . . . . . : fe80::ff48:31bd:d609:bc1c%4
IPv4 Address . . . . . : 10.0.2.15
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 10.0.2.1
```



Then I checked Ubuntu ip, updated the system and installed essential tools.

```
dokshy@dokshy-VM: ~$ sudo apt update
dokshy@dokshy-VM: ~$ sudo apt install net-tools
dokshy@dokshy-VM: ~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.8/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
        valid_lft 405sec preferred_lft 405sec
    inet6 fe80::a00:27ff:fe1:123/64 scope link
        valid_lft forever preferred_lft forever
```

Checked Kali ip, updated the system and tried to ping Ubuntu and Windows. **SUCCESS!**

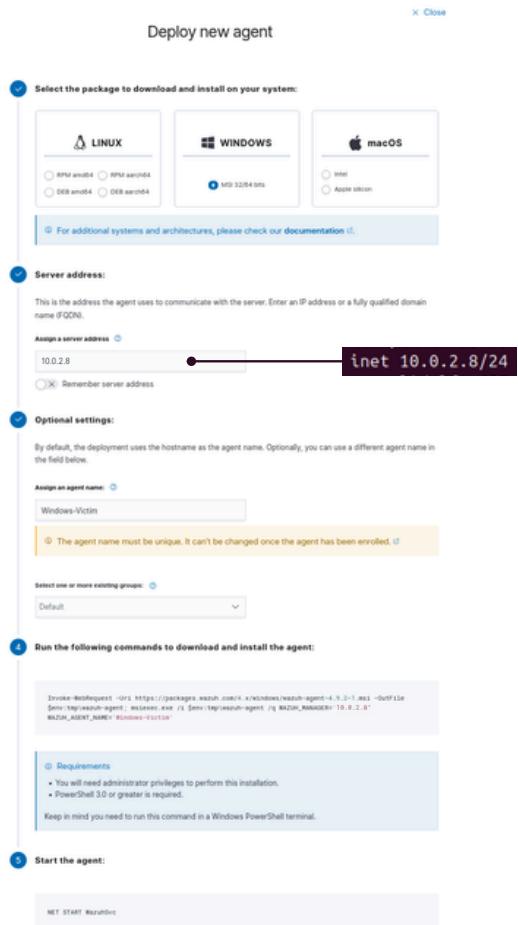
```
kali㉿kali: ~$ ip a
File Actions Edit View Help
(kali㉿kali)-[~]
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.8/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0
        valid_lft 572sec preferred_lft 572sec
    inet6 fe80::ca9:729:1b89:6994/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

```
—(kali㉿kali)-[~]
$ ping 10.0.2.4
PING 10.0.2.4 (10.0.2.4) 56(84) bytes of data.
64 bytes from 10.0.2.4: icmp_seq=1 ttl=128 time=1.06 ms
64 bytes from 10.0.2.4: icmp_seq=2 ttl=128 time=1.16 ms
64 bytes from 10.0.2.4: icmp_seq=3 ttl=128 time=1.16 ms
64 bytes from 10.0.2.4: icmp_seq=4 ttl=128 time=0.905 ms
64 bytes from 10.0.2.4: icmp_seq=5 ttl=128 time=0.914 ms
^C
— 10.0.2.4 ping statistics —
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 0.905/1.040/1.162/0.112 ms

—(kali㉿kali)-[~]
$ ping 10.0.2.8
PING 10.0.2.8 (10.0.2.8) 56(84) bytes of data.
64 bytes from 10.0.2.8: icmp_seq=1 ttl=64 time=2.37 ms
64 bytes from 10.0.2.8: icmp_seq=2 ttl=64 time=0.479 ms
64 bytes from 10.0.2.8: icmp_seq=3 ttl=64 time=0.633 ms
64 bytes from 10.0.2.8: icmp_seq=4 ttl=64 time=1.61 ms
64 bytes from 10.0.2.8: icmp_seq=5 ttl=64 time=0.470 ms
^C
— 10.0.2.8 ping statistics —
5 packets transmitted, 5 received, 0% packet loss, time 4037ms
rtt min/avg/max/mdev = 0.470/1.111/2.369/0.757 ms

—(kali㉿kali)-[~]
$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=113 time=23.1 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=113 time=21.7 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=113 time=31.8 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=113 time=29.4 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=113 time=29.4 ms
^C
— 8.8.8.8 ping statistics —
5 packets transmitted, 5 received, 0% packet loss, time 4008ms
rtt min/avg/max/mdev = 21.697/27.080/31.804/3.949 ms
```

IV. Then I deployed new agent (Windows) through Wazuh Dashboard...



...and deployed the Wazuh Agent on the target Windows endpoint via PowerShell and established a connection to the Manager.

The PowerShell window shows the command to download and install the Wazuh Agent on a Windows machine:

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Writing web request
Writing request stream... (Number of bytes written: 1790698)

env:\tmp\wazuh-agent; msieexec.exe /i $env:\tmp\wazuh-agent /q WAZUH_MANAGER='10.0.2.8' WAZUH_AGENT_NAME='Windows-Victim'
```

An arrow points down from this window to another PowerShell window showing the service start command:

```
PS C:\Windows\system32> NET START WazuhSvc
The Wazuh service is starting.
The Wazuh service was started successfully.
```

Let's check... Here! Successful integration of the Windows Endpoint. The SIEM is now receiving telemetry logs in real-time!



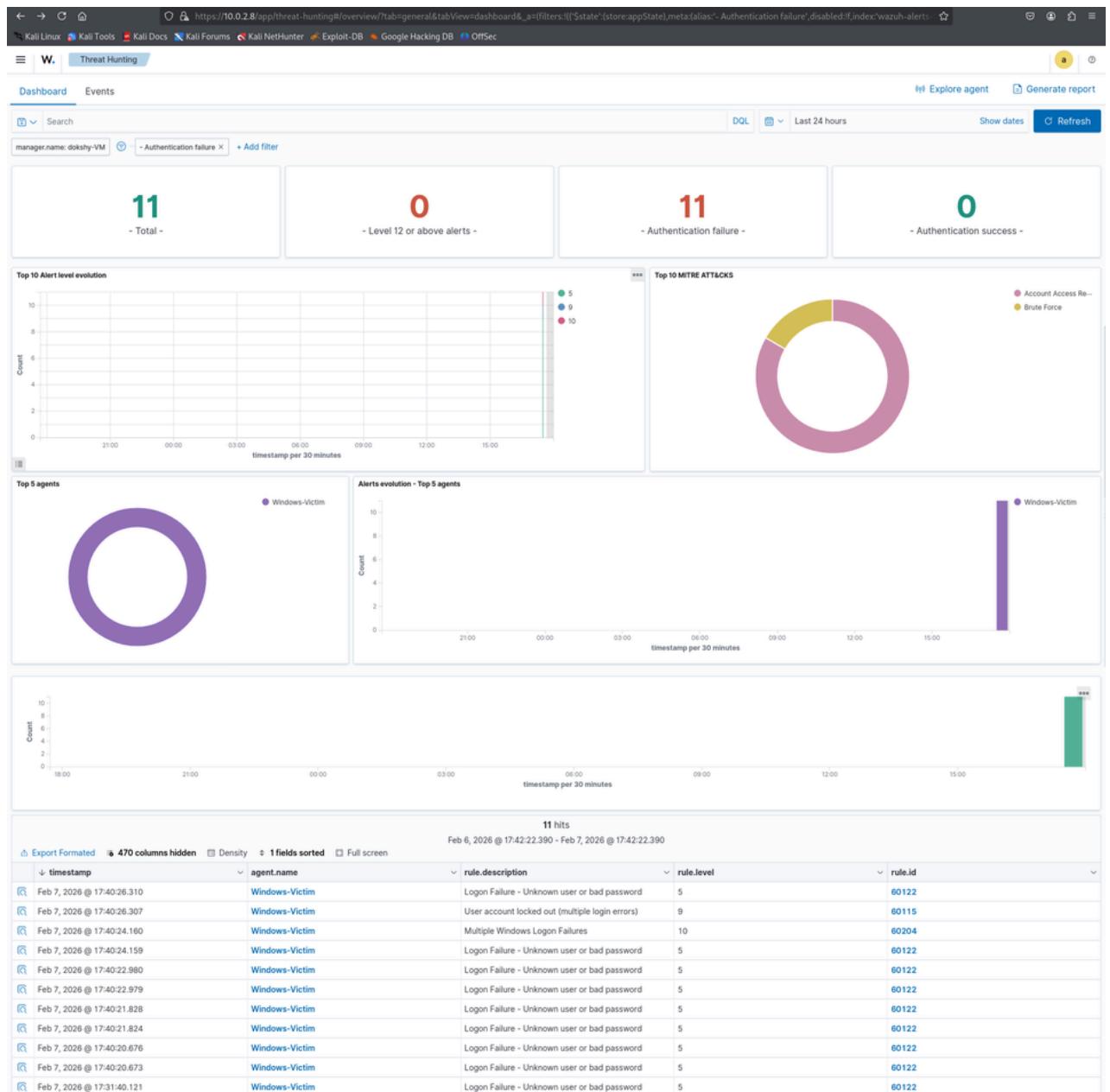
And now comes the fun part: **The Red Team Attack!** I am going to use Kali Linux to attack Windows, and verify if Ubuntu (Wazuh) catches it.

V. I verified that Kali could see the Windows open ports using **Nmap** and tried to **brute-force** the Administrator password via SMB.

★ As I know, SMBv1 is an old, insecure version of the protocol, you can use crackmapexec instead.



The Moment of truth... Wazuh actually shows brute-force!



I focused on **External Threat Detection** to stop an attacker at the door. But a strong defense strategy must also include insider threats – threats that have already bypassed the perimeter or came from inside the network.

To address this, the next phase of the lab implements **File Integrity Monitoring (FIM)**. By establishing a cryptographic baseline of critical system files, we shift from detecting 'who is logging in' to detecting 'what is being changed,' ensuring that even if an attacker gains access, their actions—such as installing malware or modifying sensitive data—trigger an immediate alert.



I also made FIM project on Python, you can check it out [here](#).

**Thank you, and see you
next time!**