Security Onion

Owen McDaniel & Scott Anderson (Version 1)
Zuzanna Wieczorek & Trevor Devries (Version 2)

September 29, 2025

Summary

Security Onion is an open-source platform designed for threat detection, network monitoring, and log management. It integrates multiple security tools—such as Zeek for protocol analysis, Suricata for intrusion detection, Wazuh for host-based monitoring, and Kibana for visualization—into a unified system. The strength of Security Onion lies not only in its deployment but also in how effectively analysts can manage and interpret the collected data. In this tutorial, we will demonstrate the setup and configuration of Security Onion, explore its key components, and practice investigating suspicious activity through guided labs and challenges.

Contents

1 Objectives of this Tutorial >

1. Understand the purpose and core components of Security Onion (Network and Host detection);
2. Learn how to deploy and configure Security Onion in a virtual environment;
3. Practice analyzing network and host activity through Security Onion's interfaces and tools;
4. Apply that knowledge by investigating simulated attacks through step-by-step activities and challenges.

This tutorial is not a complete user's guide to Security Onion. It is a structured introduction to its most important features. Through walkthroughs, activities, and challenges, we will cover the following:

1. Security Onion integrates multiple open-source tools for intrusion detection and log analysis. To understand its capabilities, it is important to learn both its architecture and the roles of its core components.
2. This tutorial aims to provide a basic understanding of the process of setting up Security Onion in a visualized lab environment, including configuration and service verification.
3. This tutorial also aims to introduce students to analyzing data from Zeek and Suricata, using both command-line queries and the Security Onion Console web GUI for visualization and correlation.
4. Finally, the skills presented in this tutorial are put to the test through a series of guided challenges, which simulate realistic network threats and allow participants to practice detection and investigation workflows.

2 Required Background <>

We assume some knowledge in the following areas:

1. Familiarity with Proxmox and installing operating systems in virtual environments;
2. Understanding of fundamental networking concepts such as the TCP/IP model, IP addressing, ports, and common protocols (DNS, HTTP, SSH, ICMP);
3. Introductory knowledge of cybersecurity and SOC operations, including concepts such as intrusion detection systems (IDS), intrusion prevention systems (IPS), and log analysis.

It is not the goal of this tutorial to be completely self-contained and self-explanatory. As such, the tutorial assumes certain background skills and knowledge. The following are some areas where we expect the users of this tutorial to have previous skills/knowledge:

1. Practical experience working with computers and virtualization software. The tutorial does not cover how to install virtualization software itself.
2. Fundamental knowledge of networking concepts. This includes an understanding of how data travels across networks, familiarity with packet-based communication, and basic concepts such as IP addresses, ports, and transport protocols (TCP and UDP).
3. A general awareness of cybersecurity principles and terminology. Concepts such as intrusion detection, network security monitoring (NSM), packet capture (PCAP), and endpoint monitoring are assumed.

3 Hardware and Software Requirements <>

To successfully complete this tutorial, the following hardware and software resources are recommended:

1. Security Onion ISO VM;
2. Windows 11 VM (host for SOC GUI);
3. Windows 10 VM (attack vector);
4. Linux Ubuntu VM (attack vector);
5. Windows metasploitable VM (attack point);
6. Virtualization software such as VirtualBox, VMware Workstation, or VMware Fusion;

The tutorial is designed for hands-on practice in a virtualized environment. A reasonably modern laptop or desktop computer with sufficient memory and processing power is required to run both the Security Onion server and one or more client machines simultaneously.

4 Network Architecture for Security Onion <>

Here is the network topology for the lab environment:

1. Bridge 990 (explain why)
2. Security Onion ISO (192.168.90.6);
3. Windows 11 VM (192.168.90.4);
4. Windows 10 VM -NoElastic (192.168.90.7);
5. Windows 10 VM -Elastic (192.168.90.5);
6. Linux Ubuntu VM (192.168.90.3);
7. Windows metasploitable VM (192.168.90.8);

5 A Problem: Modern Network Threats <>

Modern organizations face a variety of network-based threats:

1. Malware and ransomware that spread rapidly across networks;
2. Phishing campaigns that lead to credential theft and unauthorized access;
3. Command-and-control (C2) channels used by attackers to exfiltrate data;
4. Insider threats and compromised endpoints that bypass perimeter defenses;
5. Advanced Persistent Threats (APTs) that remain undetected for long periods.

Network infrastructures are more exposed than ever before. Traditional perimeter defenses such as firewalls are often insufficient, as attackers use encryption, obfuscation, and legitimate services to mask their activities. Organizations require tools that can provide deep visibility into network traffic, correlate host and network events, and support analysts in detecting and investigating suspicious behavior. This is the context in which **Security Onion** provides value, acting as a comprehensive platform for network security monitoring (NSM), intrusion detection, and log management.

6 The Solution: Security Onion <>

Security Onion is an open-source platform that integrates multiple security tools into one environment:

1. Provides network security monitoring for protocol analysis;
2. Delivers signature-based intrusion detection using Suricata;
3. Supports host-based monitoring and log analysis;
4. Offers powerful search and visualization;
5. Centralizes alerts and workflows in the SOC web interface.
6. Offers powerful host based detection.

Security Onion provides a unified defensive platform that combines host-based and network-based detection capabilities. Instead of relying on a single tool, it integrates multiple open-source components into a cohesive solution that allows analysts to detect, investigate, and respond to threats effectively.

By deploying Security Onion, organizations gain:

- Visibility into network traffic and endpoint activity;
- Correlation of alerts across different tools;
- A centralized platform for threat hunting, log management, and incident response;
- The ability to practice and train SOC workflows in a realistic environment.

In this tutorial, we will demonstrate how to set up Security Onion, use its core tools, and investigate simulated attacks through the SOC interface by analyzing the logs.

7 Core Components of Security Onion <>

Security Onion integrates several powerful open-source tools, each serving a unique role:

1. **Zeek (formerly Bro)** – Protocol analysis and metadata extraction from network traffic;
2. **Suricata** – Signature-based intrusion detection and packet inspection;
3. **SOC Web Interface** – Central dashboard for alerts, hunting, and incident correlation.
4. **Elastic Agent Host Based Detection** A power host based agent that can detect things at the host level.

Each component contributes to Security Onion's layered defense approach:

- **Zeek** provides rich context about connections, DNS queries, HTTP traffic, and other protocols, helping analysts detect unusual behaviors.
- **Suricata** uses community and custom rulesets to detect known malicious traffic patterns, producing real-time alerts.
- **SOC Interface** unifies these tools, enabling security analysts to pivot between alerts, logs, and timelines during investigations.

Together, these components form a comprehensive security monitoring platform that is greater than the sum of its parts.

8 Deploying Security Onion in Virtual Environment <>

1. Start the Security Onion VM and Win11 VM;
2. Log into the server and go to http://192.168.90.6 on Win11 web browser.
3. Log into the SOC web GUI.

```
admin@securityonion.local
password
```

Use Win11 VM as a host for Security Onion.

9 Configuring Services and Verifying Installation <>

1. In the SOC CLI run `sudo so-status` to check the core services.;
2. Access the SOC web interface to confirm dashboards and alerts are operational.

1. The `so-status` provides a concise overview of all running services. A fully functional installation will show green status for Zeek, Suricata, Wazuh, Elasticsearch, and Kibana. Any services marked as failed or inactive require troubleshooting before proceeding.
2. Finally, the SOC web interface is accessed via a browser to visually verify that dashboards, alerts, and log feeds are operational.

10 Activity: Sync Security Onion to pfSense (Time Synchronization) <>

Ensure accurate time synchronization between your Security Onion sensor and pfSense firewall using Chrony:

1. Edit the Chrony configuration file:
   - Run: `sudo vi /etc/chrony/chrony.conf`
   - Add the following lines:

     ```
     server <pf sense IP> iburst minpoll 2
     maxpoll 2 xleave
     driftfile /var/lib/chrony.drift
     makestep 0.1 3
     rtcsync
     ```

2. Manually add the pfSense NTP server in Chrony:
   - Run: `sudo chronyc add server <pf sense IP> iburst`
   - Then force synchronization with: `sudo chronyc makestep`
3. Edit the Chronyd service override configuration:
   - Run: `sudo systemctl edit chronyd`
   - Add the following content:

     ```
     [Service]
     ExecStart=
     ExecStart=/usr/sbin/chronyd -f
     /etc/chrony/chrony.conf
     ```

11 Activity: Sync Security Onion to pfSense (cont.) <>

4. Save and reload systemd, then restart the service:

Deliverables:

1. Verification that Security Onion is synchronized with pfSense by running: `chronyc tracking`

HINTS:

1. Confirm pfSense is running the NTP service and accessible at `<PF sense IP>`.
2. If synchronization fails, verify that your Security Onion VM can reach pfSense using `<PF Sense IP>`.
3. Use `systemctl status chronyd` to ensure the Chrony service starts automatically on boot.

12 Activity: Installing and Configuring Elastic Agents (Linux + Windows) <>

Deploy endpoint agents on both Ubuntu and Windows targets to collect host-level telemetry and correlate it with network alerts.

1. **Prepare target systems:**
   - **Ubuntu target:** Disable firewall temporarily: *sudo ufw disable* and confirm *sudo ufw status*.
   - **Windows target:** Open an elevated PowerShell and run: *netsh advfirewall set allprofiles state off*; verify with *netsh advfirewall show allprofiles*.
2. **Download installers and enrollment tokens:**
   - From the Security Onion SOC, go to **Downloads** → **Elastic Agent Installers** (or Wazuh agent packages) and copy the Fleet enrollment command / token.
3. **Install Elastic Agent on Ubuntu (Security Onion wrapper executable):**
   - Assume the file is in *~/Downloads* and named *so-elastic-agent_linux_amd64*.
   - Make the SO wrapper executable: *sudo chmod +x so-elastic-agent_linux_amd64*.
   - Run the Security Onion elastic agent installer wrapper: *sudo ./so-elastic-agent_linux_amd64*.

13 Activity: Installing and Configuring Elastic Agents (Continued) <>

4. **Install Elastic Agent on Windows:**
   - Navigate to your Security Onion IP (e.g., *https://192.168.90.6*), go to Downloads, click the Windows Elastic Agent link, keep the file, then open an elevated PowerShell and run the installer as Administrator.
   - When prompted, paste the Fleet enrollment command from the SOC (use *--insecure* if the SOC uses self-signed certs).
5. **Verify registration and telemetry:**
   - Check Fleet → Agents (Elastic) or the Wazuh manager UI for agent health.
   - Confirm example telemetry: a process start event (Windows) and a heartbeat from Ubuntu.

Deliverables:

1. Make sure Fleet is showing both Ubuntu and Windows agents as "Healthy."

14 Exploring Security Onion's Web GUI <>

1. Log into the Security Onion Web GUI through Win11 using `http://192.168.90.6`.
2. Log in with your administrator credentials created during setup.
3. Explore the main dashboards: Overview, Alerts and Hunt.

1. Zeek alerts can be found in Hunt interface.
2. Suricata alerts can be found in Alerts interface.
3. You can filter logs by the source/destination IP, log type, protocol, etc.

15 Activity: Using ECS Fields <>

On the Security Onion GUI, perform the following tasks:

1. Go to Hunt. Ensure your time range is set. Enter the base query: `event.dataset:zeek.*`
2. Expand a log entry by clicking the arrow (→) on the far left of any row in the Events Table. The expanded panel shows all the fields and their values for that specific event.
3. Find the ECS field you want to add (e.g., `source.ip`, `dns.question.name`). Click the tiny icon next to the field name in the expanded panel. This icon usually looks like a column/table icon or a plus sign (+). Clicking it toggles that field as a column in the main Events Table.
4. Repeat this for all the fields you need.

Choosing your own ECS fields can make the log analysis process more efficient.

16 Activity: Log Generation <>

Generate basic network activity (ARP/ICMP/Internal HTTP) to confirm the Security Onion bridge is logging traffic between internal hosts.

1. Open Command Prompt on Ubuntu01 (192.168.90.3) and run a simple ping to the Metasploitable target: ping 192.168.90.8
2. Open a web browser and navigate to a web service running on a lab machine (e.g., if the Metasploitable box has a web server): http://192.168.90.8
3. Attempt to connect to a service on the Metasploitable target that's likely not running or is blocked (or try a common malicious port): `telnet 192.168.90.8 4444`

1. Creates entries in Zeek's `conn.log` showing ICMP traffic between the two hosts.
2. Creates entries in Zeek's `http.log` and `conn.log` showing internal web activity.
3. May generate a generic Suricata alert or a connection-related notice in `eve.json` on Security Onion, confirming it's inspecting traffic.

17 Role of Zeek in Threat Detection <>

1. Zeek passively monitors network traffic to generate detailed logs;
2. Logs include connection metadata, DNS queries, HTTP requests, and security notices;
3. Enables threat hunting by identifying anomalies and suspicious patterns;
4. Works in conjunction with other SOC tools to provide context-rich alerts.

1. Zeek (formerly Bro) is a powerful network analysis framework that does not block traffic but instead passively monitors it. By capturing and parsing network traffic, Zeek produces structured logs that give insight into all aspects of network communications.
2. Zeek's log files include `conn.log` for connection metadata, `dns.log` for domain queries, `http.log` for web requests, and `notice.log` for alerts and suspicious activities. These logs provide the foundation for detecting anomalies in the network.
3. Analysts can use Zeek logs to perform threat hunting, such as identifying command-and-control (C2) traffic, unusual DNS queries, or repeated failed connection attempts. This enables proactive detection of attacks that may not trigger traditional signatures.
4. Zeek integrates with Security Onion's other components (Suricata, Wazuh, Kibana) to enrich alerts and provide context. While Suricata detects threats using signatures, Zeek offers behavioral visibility, allowing SOC analysts to correlate data and respond effectively.

18 Activity: Exploring Zeek Log Files <>

1. Go to the SOC left menu and click Dashboards.
2. Observe the widgets on the dashboard (e.g., Top Talkers, Top Domains, Protocol Breakdown). These visually summarize the logs.
3. Go to Hunt. Run the query: `event.dataset:zeek.notice`. Click on one of the resulting log entries to expand it.

1. Zeek stores network traffic information in structured log files, making it easier for SOC analysts to examine detailed metadata about connections, DNS queries, HTTP requests, and notices about suspicious activity.
2. The main log files include:
   - `conn.log`: records connection metadata such as source and destination IPs, ports, and protocols;
   - `dns.log`: captures DNS query and response activity;
   - `http.log`: logs HTTP requests, including methods and URLs;
   - `notice.log`: alerts for events like suspicious activity or anomalies.
3. By reviewing and correlating these logs, analysts can identify patterns indicative of malicious activity, such as repeated failed connections, anomalous DNS queries, or suspicious HTTP requests, forming the foundation for threat detection.

19 Challenge 1: Log Ingestion Verification <>

1. Find Zeek Logs in Hunt.
2. Find Suricata Alerts.
3. Pivot Suricata Alert to Related Logs.

HINTS:

1. The main Events table shows log entries for `zeek.conn`, `zeek.http`, `zeek.dns`,
2. The Alerts list shows entries with different severity.

20 Activity: C2 Beacon <>

Simulate repeated, suspicious internal traffic patterns without relying on external domains. We simulate C2 over an internal IP/Port and DNS requests to a fake TLD (Top-Level Domain).

1. On Ubuntu01 use the `nslookup` command to repeatedly query a fake, randomly-named domain ending in a fake TLD (e.g., `.lab`). Crucially, the DNS server is the Metasploitable machine itself (or a designated local DNS server, if configured). Command:
   ```
   for i in {1..20}; do nslookup
   7tXy2zLpKrA.lab 192.168.90.8; sleep 1;
   done
   ```
2. Use nc (netcat) to repeatedly attempt connections to a high port (like 8080 or 8443) on the Metasploitable target. Command:
   ```
   for i in {1..20}; do nc -z -w 1
   192.168.90.8 8080; sleep 1; done
   ```

Expected log results:

1. Generates repeated entries in Zeek's `dns.log` to a specific internal server (if configured) with unusual, high-entropy domain names, standing out as anomalous beaconing behavior.
2. Creates many frequent, often short or failed, connection entries in Zeek's `conn.log` from 192.168.90.3 to 192.168.90.8:8080, indicating C2 beaconing.

21 Challenge 2: Detecting a C2 Beacon <>

On the Security Onion GUI, perform the following tasks:

1. Find the suspicious domain connection attempt.
2. Read and analyze the logs.
3. Pivot to Related Events/PCAP.

HINTS:

1. Go to Hunt and filter the logs by the source IP.
2. Click the expand arrow on one of the alerts in the list. In the details panel, find the event characteristics (src ip, protocol, port etc.) Click the action after clicking over the chosen field to "Hunt" or "View Related Events".

22 Activity: Rulesets and Custom Rule Writing <>

Use Security Onion VM CLI:

1. Edit the Custom Rule File. The custom rules are typically included from the standard Suricata configuration directory. Input command:
   `sudo vi /nsm/suricata/custom.rules`
2. Add the rule in `vi`. Press the letter `i` to enter INSERT MODE. Type the rule below:
   ```
   alert http any any -> any any
   (msg:"Custom Rule:  Malicious Keyword
   badstuff Detected"; content:"badstuff";
   http_uri; sid:1000002; rev:1;)
   ```
3. Press the Esc key to exit INSERT MODE. Type `wq` and press Enter.
4. Restart Suricata to load the new rule: `sudo so-suricata-restart`

Custom rules enable the SOC analyst to look for recognizable characteristics such as the name.

23 Challenge 3: Triggering and Investigating Malicious HTTP Traffic <>

Send the specific HTTP request that contains the keyword, triggering the custom rule.

1. Set your own rule for Suricata.
2. Log in to Win10 (192.168.90.7) and open a Web Browser.
3. Enter the Malicious URL.
4. Go to the Secuerity Onion GUI and find the Suricata Alert in the SOC.
5. Correlate with the Zeek logs.

HINTS:

(a) Change the keyword in the previously written rule.

(b) Type the Metasploitable IP followed by the path containing the keyword badstuff: `http://192.168.90.8/malicious/download_badstuff.exe`

(c) Look for the most recent alert in Suricata containing the keyword.

24 Transition: Network vs Host-Based Detection in Security Onion <>

Before moving into host-based detection, it's important to understand how it complements the network-based detection your partner has been working on.

Network Detection (SOC Sensors):

- Uses tools like **Zeek**, **Suricata**, and **Stenographer** to analyze traffic mirrored from the network.
- Focuses on packet-level and flow-based telemetry — such as DNS lookups, HTTP requests, SSL/TLS handshakes, and protocol anomalies.
- Detects suspicious activity that crosses the wire: command-and-control traffic, lateral movement, beaconing, and data exfiltration attempts.
- Provides visibility into network communications but cannot see local user or process activity inside a host.

25 Transition: Host-Based Detection in Security Onion <>

Host-Based Detection (Elastic Agents):

- Expands visibility from network flows into the **endpoint itself**.
- Elastic Agents collect host telemetry — including process creation, file modifications, registry edits, logins, and system configuration changes.
- These logs are sent directly to the Security Onion SOC, where they are correlated with network alerts for end-to-end context.
- This layered approach enables analysts to determine not just *that* a threat occurred on the network, but also *what happened on the affected system*.

In summary: Network detection answers *"what crossed the wire?"*, while host-based detection answers *"what happened on the endpoint?"*. Together, they provide a comprehensive defense-in-depth view within Security Onion.

Next: Host-Based Detection with Elastic Agents

26 Activity: Validate Sysmon Telemetry via Elastic Agent (Air-Gapped) <>

Confirm that host-level telemetry from Sysmon is being collected and forwarded to Security Onion via the Elastic Agent.

(a) **On the Windows 10 host: Verify Sysmon is active.**

- Open an elevated PowerShell and run: `Get-Service Sysmon64`
- Ensure the status shows **Running**.

(b) **Check for Sysmon event generation.**

- Generate activity: `notepad`, `cmd`.
- Then confirm Sysmon is logging events: `Get-WinEvent -LogName "Microsoft-Windows-Sysmon/Operational" -MaxEvents 5 | Format-List`
- Look for entries such as **Event ID 1 – Process Creation**.

(c) **Verify Elastic Agent is forwarding events.**

- From the analyst (Windows 11) monitoring VM, open the SOC → **Elastic** → **Discover**.
- Confirm your Windows 10 endpoint shows a recent *last activity* timestamp.

27 Activity: Validate Sysmon Telemetry via Elastic Agent (Continued) <>

(a) **Search for Sysmon events in the SOC.**

- In Elastic → **Discover**, run this KQL query:

```
event.provider:"Microsoft-Windows-Sysmon"
```

- Verify process creation, network connection, and registry events appear from the Windows 10 host.

28 Challenge: Investigate Application Execution Using Sysmon Data <>

Analyze process-creation telemetry from Sysmon to identify application activity on the Windows 10 host.

(a) In the SOC → **Elastic** → **Discover**, apply the following KQL filters:

```
process.name:"cmd.exe"
```

(b) Expand an event and interpret its context:

- Does this look like normal administrative activity?
- Could it represent potential misuse of CMD?

(c) (Optional) Broaden your query:

```
process.command_line:*cmd* or *notpad*
```

29 Activity 5: Observe Command-Line Execution (Windows) <>

Confirm that commands typed in CMD appear in Security Onion telemetry.

(a) On the Windows 10 host, open **Command Prompt (Run as Administrator)**.

(b) Type the following and press Enter:

```
whoami
```

(c) Wait about 30 seconds for Elastic Agent to forward logs.

(d) In the SOC → **Elastic** → **Discover**, paste this KQL (replace ¡win10-host¿):

```
host.name:"<win10-host>"
and process.parent.name:"cmd.exe"
```

(e) Locate the process that matches your current CMD PID and expand it.

(f) Observe:

- `process.pid`
- `process.command_line`
- `user.name`

Verify the `whoami` command appears in the log.

30 Challenge 5: Investigate Application Execution Using Sysmon Data <>

Independently identify the log entry for a new command executed from CMD.

(a) On the Windows 10 host, open **Command Prompt (Run as Administrator)**.

(b) Execute:

```
notepad
```

(c) Wait 30 seconds for the event to reach the SOC.

(d) In the SOC → **Elastic** → **Discover**, reuse the KQL:

```
type in the correct query
to find the opening of cmd
and the line to open notepad
```

(e) Locate and expand the event showing your `notepad` execution.

(f) Note the `process.command_line` and `user.name` fields, and describe in one sentence what this log reveals.

31 Conclusion and Lessons Learned <>

Summarize the key takeaways from working with Security Onion throughout this tutorial:

(a) Security Onion provides a unified platform that integrates network (Zeek, Suricata) and host-based (Wazuh) monitoring tools;

(b) Analysts can pivot between alerts and logs in the SOC interface to build a complete picture of an incident;

(c) Hands-on activities demonstrated how to detect, investigate, and respond to real-world style threats;

(d) Defense-in-depth requires multiple layers of detection — no single tool is sufficient on its own;

(e) Continuous learning, tuning of rules, and practice are essential for effective SOC operations.

HINTS:

1. Think back across the modules — what part of Security Onion did you find most useful or interesting?
2. Consider how the integration of Zeek, Suricata, and Wazuh mirrors real SOC workflows.
3. Reflect on challenges faced during the lab (e.g., troubleshooting, interpreting logs) and how they enhanced your understanding.
4. Lessons learned can be technical (rule writing, PCAP analysis) or strategic (importance of layered security).

32 Appendix: Solutions, Command Outputs, and Screenshots <

(a) Answers to the Quiz;

(b) Solutions to the Challenges:

i. Challenge 1;

ii. Challenge 2;

iii. Challenge 3;

iv. Challenge 4;

v. Challenge 5;

vi. Challenge 6;

vii. Bonus Challenge;

(c) Change Log.

Answers to the Quiz:

(a) Over-protective configurations can cause major hindrances to some applications, effectively crippling them.

(b) Packet filter firewalls are able to control traffic related to ports, IP addresses and network requests. Application layer firewalls can do all that and in addition, can configure rules with respect to specific applications and services.

(c) "Defense in depth" approach is not about just firewalls. There are many layers of protection. Firewalls are often the first line of protection for a network.

(d) They are more properly called "firmware" firewalls because most firewalls carry out their duties at the software level as applications in Linux, rather than as actual circuits in hardware.

Solutions to the Challenges:

(a) **Challenge 1:**

i. In "Inbound Rules" section, click on "New Rule" at the right pane.
ii. In resultant wizard, Custom Rule type. All Programs.
iii. Protocol type: ICMPv4. Scope: Any IP address.
iv. Action: Block the connection. Profile: All three.
v. Give a RULENAME and description. Then, finish.
vi. Repeat the same by changing Protocol type to ICMPv6.
vii. Repeat the same thing twice (once for ICMPv4 and another for ICMPv6) in "Outbound Rules".

To disable rules using CLI, use the command: `netsh advfirewall firewall set rule name="RULENAME" new enable=no`, where RULENAME is the name of rule which one has assigned at the end of creating every rule.

(b) **Challenge 2:**

i. In "Inbound Rules" section, click on "New Rule" at the right pane.
ii. In resultant wizard, Custom Rule type.
iii. Program: The program path is: `C-Windows-System32-mstsc.exe`.
iv. Services: *Customize*. Apply to all services beginning with the name "Remote Desktop Services".
v. Any protocol type – Any IP addresses – Block the connection-All Profiles
vi. Give a name and description. Then, finish.
vii. Repeat the same for "Outbound Rules".

To create an exception for 192.168.1.100/24 subnet,

i. In "Inbound Rules" section, right click on the rule created in PART-1 of the challenge.
ii. Go to `General` tab. Change *Action* from `Block the connection` to *Allow the connection*.
iii. Go to `Scope` tab. Do the following step for both `Local IP address` and `Remote IP address`
iv. Change radio selection to `These IP addresses:`. Click on button `Add` and enter the subnet value `192.168.1.100/24`
v. Apply the modifications and `OK` to change the rule.
vi. Repeat the same for "Outbound Rules".

(c) **Challenge 3:**

i. To block the offending website(s), you will need an IP address. In command prompt execute the following command:
`nslookup websitename`

ii. In the Windows Firewall Advanced Settings:
Create an inbound and outbound rule as follows:

A. Select `"New Rule"`
B. Select `"All Programs"`
C. Protocol type `Any` and `All Ports`
D. Under `"Which remote IP addresses does this rule apply to?"`
E. Add the IP address of the offending website and select `"Next"`.
F. Select `"Block the connection"`
G. Select for all profiles.
H. Name the rule.

iii. Now test to make sure your rule works

(d) **Challenge 4:**

i. A. In `gufw`, switch the state of the current profile to *OFF*.
B. Open a terminal.
C. Execute the following 3 commands to make sure each of the default chains in `iptables` are set to `ACCEPT`:
`sudo iptables --policy INPUT ACCEPT`
`sudo iptables --policy OUTPUT ACCEPT`
`sudo iptables --policy FORWARD ACCEPT`
D. You might execute the following commands to confirm each chain states "Chain {chain-name} (policy ACCEPT)":
`sudo iptables -L`
OR
`sudo iptables -L INPUT`
`sudo iptables -L OUTPUT`
`sudo iptables -L FORWARD`

ii. On the Kali VM:
A. Open a terminal.
B. Initially, execute the following command to see if you can determine the IP address of the target machine ("Ubuntu-gufw"):
`nmap 192.168.1.0/24`
C. Once you have the IP address, execute the following command:
`nmap -p "*" {IP Address} -sV -sS -O -T4`
D. What you see is that Nmap reports all ports as closed, and the OS could not be identified.

iii. Return to the "Ubuntu-GUFW":
A. In `gufw`, switch status to *ON*.
B. Set "Incoming" to *DENY*, if it is not already set to that.

iv. Repeat step 2. Nmap reports that all ports are now filtered. The OS is still not recognizable.

v. Return to the "Ubuntu-GUFW":

A. In `gufw`, switch the state of the current profile to *OFF*.

B. In the terminal, execute the following commands:
```
sudo iptables --policy INPUT DROP
sudo iptables --policy OUTPUT DROP
sudo iptables --policy FORWARD DROP
```

vi. Repeat step 2. Nmap still reports that all ports are filtered, the OS indeterminate.

vii. Before continuing, set `iptables` back to an open state:
```
sudo iptables --policy INPUT ACCEPT
sudo iptables --policy OUTPUT ACCEPT
sudo iptables --policy FORWARD ACCEPT
```

(e) **Challenge 5:**
Open `gufw`.

i. In the **Rules** section, use the + button to create a new rule. On the **Preconfigured** tab:
Block and log all traffic related to Steam and send a message to the person attempting communication, conveying the block.

A. Policy: "Allow".

B. Direction: Both. Category: All. Subcategory: Telephony.

C. Application: Skype-Normal. Copy values and Jump to "Advanced" tab.

D. Log all. To ports - 23390:23399. Add.

ii. A. To block the offending website(s), you will need an IP address. In the terminal, execute the following command:
`nslookup {websitename.tld}`

B. In the **Rules** section, use the + button to create a new rule. On the **Advanced** tab:
Policy: *Deny*;
Direction: *Both*;
Interface: *All Interfaces*;
Log: *Log All*;
Protocol: *Both*.

C. To, IP: {IP Address};

D. Add the rule.

E. Test that you are no longer able to get to the offending website(s) in the browser. Try **`gogle.com`** and **`facbook.com`**

F. Test that you can still reach other unrelated websites. Try **`wtcamerastore.com`**, **`wtelectronicsstore.com`**, and **`wtshoestore.com`**. This set of websites should not be blocked.

G. You might experiment with the difference between setting Direction in the rule to *In* or *Out*. You should find that *In* does not block web requests (because they originate from your own computer).

H. Viewing the **Rules** section, select the rule in the list of rules, and click – to delete it.

(f) **Challenge 6:**
On the "Ubuntu-GUFW" VM, open a terminal:

i. A. The following should work:
sudo iptables -A INPUT -p icmp -j DROP
More specifically:
sudo iptables -A INPUT -p icmp --icmp-type 8 -j DROP
Or perhaps more appropriately [**?**]:
sudo iptables -A OUTPUT -p icmp -j ACCEPT
sudo iptables -A INPUT -p icmp --icmp-type echo-reply -s 0/0 -j ACCEPT
sudo iptables -A INPUT -p icmp --icmp-type destination-unreachable -s 0/0 -j ACCEPT
sudo iptables -A INPUT -p icmp --icmp-type time-exceeded -s 0/0 -j ACCEPT
sudo iptables -A INPUT -p icmp -j DROP

B. To delete the rules, repeat any of the commands you executed above and change `-A` to `-D`.

ii. A. To block the offending website(s), `iptables` enables you to use either IP address or domain name. Blocking by IP address might be a little easier to implement in our current case, but it is not necessarily appropriate in a real-world environment; whereas blocking by domain name is better targeted, but can be more tedious at the moment if entering all the offending websites:
In a terminal, execute the following sequence of commands:
sudo iptables -I INPUT 1 -s {IP Address} -j drop
sudo iptables -I OUTPUT 1 -d {IP Address} -j drop
Or, as the better option:
sudo iptables -I INPUT 1 -s gogle.com -j drop
sudo iptables -I OUTPUT 1 -d gogle.com -j drop
... etc.

B. Test that you are no longer able to get to the offending website(s) in the browser. Try **gogle.com** and **facbook.com**

C. Test that you can still reach other unrelated websites. Try **wtcamerastore.com**, **wtelectronicsstore.com**, and **wtshoestore.com**. This set of websites should not be blocked.

D. To delete the rules:
sudo iptables -D INPUT 1

sudo iptables -D OUTPUT 1
etc.

(g) **Bonus Challenge:**

`Computer Configuration` –> `Administrative Templates` –> `Network` –> `Network Connections` –> `Windows Firewall` –> `Domain Profile`.

Details:

i. Disabling `Local program exceptions` will make it work such that the client machines cannot specify any local program exceptions to Windows Firewall policies set by Domain Controller.

ii. Windows Firewall provides notifications only when in a critically unsecure state. As such, `Prohibit Notifications` should be disabled.

iii. Dropped packets sometimes hold essential value in conducting forensic analysis. As such, it is always recommended to perform `logging of dropped packets`.

iv. Disabling `Local ports exceptions` will make it work such that the client machines cannot specify any local exceptions for ports in contrast to Windows Firewall policies set by Domain Controller.

Change Log:

Ver.	Date	Authors	Changes
v1.0	March 2025	Owen McDaniel & Scott Anderson	First draft of tutorial.
v2.0	September 2025	Zuzanna Wieczorek & Trevor Devries	Major content and structure changes.

References

- ChatGPT. "Generate a short summary about Security Onion." OpenAI, 29 September 2025, chat.openai.com.

- ChatGPT. "Generate a concise tutorial description based on the provided format." OpenAI, 29 September 2025, chat.openai.com.

- ChatGPT. "Generate a concise description of the required background based on the provided format." OpenAI, 1 October 2025, chat.openai.com.

4. Security Onion Solutions. *Security Onion Documentation.* Available: `https://docs.securityonion.net` [Accessed: Oct. 2025].

5. Zeek Project. *Zeek Documentation.* Available: `https://docs.zeek.org` [Accessed: Oct. 2025].

6. Open Information Security Foundation. *Suricata IDS/IPS Documentation.* Available: `https://suricata.io/documentation/` [Accessed: Oct. 2025].

7. Wazuh Inc. *Wazuh Documentation.* Available: `https://documentation.wazuh.com` [Accessed: Oct. 2025].

8. MITRE ATT&CK Framework. Available: `https://attack.mitre.org` [Accessed: Oct. 2025].

9. Cichonski, P., Millar, T., Grance, T., and Scarfone, K. *NIST Special Publication 800-61 Revision 2: Computer Security Incident Handling Guide.* National Institute of Standards and Technology, 2012.

10. Sanders, C. *Practical Packet Analysis, 3rd Edition.* San Francisco: No Starch Press, 2017.

11. Northcutt, S., Zeltser, L., Winters, S., Frederick, K., Ritchey, R. *Inside Network Perimeter Security.* Sams Publishing, 2005.

12. OpenAI. ChatGPT. "Give me a template to write a challenge." Prompted by Trevor De Vries, OpenAI ChatGPT, 30 September 2025.

13. OpenAI. ChatGPT. "Give me a template to write an activity." Prompted by Trevor De Vries, OpenAI ChatGPT, 30 September 2025.

33 Appendix: Command Outputs and Screenshots <

Below are example outputs and screenshots for key activities and challenges. Use these to validate your results and troubleshoot issues.

Example: `so-status` Security Onion: so-status zeek: running suricata: running wazuh-manager: running elasticsearch: running logstash: running kibana: running

Example Suricata Alert (eve.json)

timestamp: 2025-10-01T18:42:53.123456Z $event_type$: $alert src_ip$: 192.168.1.50$dest_ip$: 203.0.113.15$signature_id$: 1000002$signature$: $ETMALWAREMaliciousHTTPUser - Agent severity$: 2

Example Zeek DNS Log Entry 1729012451.123456 C2dNfE3pAqD 192.168.1.50 203.0.113.53 udp 53 example-malware.com A - NOERROR F 123.45.67.89

Screenshot Placeholders: - SOC → Suricata Alerts panel - SOC → Zeek HTTP/DNS logs - SOC → Wazuh Authentication alerts

Use these as reference points — your logs may differ in timestamps, IPs, or hostnames.