Proposal for Converting Legacy Servers into New-Generation Firewalls: A Comprehensive Guide

Abstract

This proposal outlines the steps necessary to repurpose legacy servers into modern, efficient, and secure firewalls using open-source technologies, primarily based on Linux. By leveraging Linux distributions such as Debian and Ubuntu, along with advanced security tools like `iptables`, `fail2ban`, and intrusion detection systems (IDS) like **Zeek** and **Snort**, this solution enhances network security while optimizing the use of outdated hardware. The approach ensures effective network monitoring, threat detection, malware prevention, and active defense against attacks.

Briefing Plan

1. Introduction

The project focuses on converting outdated servers into modern, efficient Linux-based firewalls, which can help optimize organizational costs and improve the environmental footprint. By utilizing existing hardware, this initiative aims to reduce technological waste and enhance the security infrastructure of an organization.

2. Project Overview

- **Objective**: Repurpose old servers into cost-effective Linux-based firewalls.
- **Target Audience**: IT departments, network administrators, and sustainability teams within the organization.

3. Steps in the Project

- **Assessment of Existing Infrastructure**: Evaluate current hardware and assess its suitability for conversion into a firewall.
- **Selection of Linux Distribution**: Choose an appropriate lightweight Linux distribution (e.g., Ubuntu Server, CentOS, or specialized firewall OS like pfSense, OPNsense).
- **Firewall Configuration**: Install and configure firewall software to meet the organization's security requirements, such as intrusion detection, traffic filtering, VPN setup, and monitoring tools.
- **Testing and Deployment**: Test the converted firewalls in a lab environment to ensure performance and security compliance before full deployment across the organization.
- **Maintenance Plan**: Establish procedures for regular updates, security patches, and performance monitoring.

4. Benefits of Converting Old Servers into Linux-based Firewalls

A. Cost Optimization

- **Reduction in Hardware Costs**: Repurposing old servers significantly lowers the need to purchase new, expensive hardware.
- **Low Operational Costs**: Linux-based firewalls are often open-source, reducing software licensing costs compared to proprietary firewall solutions.
- **Energy Efficiency**: Modern Linux distributions are lightweight and less resource-intensive, leading to lower power consumption, thus reducing overall energy costs.
- **Reduced Maintenance Costs**: Linux-based systems tend to require fewer resources for ongoing maintenance, saving on IT labor costs.

B. Security Enhancements

- **Advanced Firewall Features**: Linux-based firewalls provide robust security, such as customizable packet filtering, intrusion prevention systems (IPS), and VPN support, improving the overall network defense.
- **Open-Source Flexibility**: Linux-based firewalls can be easily adapted and integrated with existing infrastructure, allowing for a more tailored and responsive security setup.

- **Regular Security Updates**: Linux systems are actively updated with the latest security patches, ensuring the firewall remains resilient to emerging threats.

C. Environmental Impact

- **Reduction of E-Waste**: Repurposing old servers reduces the amount of electronic waste, contributing to environmental sustainability.
- **Longer Hardware Lifecycle**: Extending the life of IT equipment delays the need for new devices, reducing the environmental footprint associated with manufacturing, transportation, and disposal of electronics.

5. The Role of the European Union's "Right to Repair" Measures

The European Union has been taking significant steps toward minimizing electronic waste through its "Right to Repair" measures. These measures ensure that consumers and businesses have the ability to repair and extend the lifespan of their electronic devices, including servers, through the following actions:

- **Encouraging Repairability**: Manufacturers are now required to make their products, including servers, easier to repair by providing necessary parts and repair manuals.
- **Supporting Recycling**: The EU encourages the recycling of electronic waste and the repurposing of components, such as old servers, to reduce the need for new devices.
- **Reducing Waste**: By making repair more accessible and affordable, the EU helps prevent premature disposal of electronic goods, which can have harmful environmental effects.
- **Sustainability Goals**: These measures align with the EU’s broader sustainability and circular economy initiatives, ensuring that electronic products have a longer, more productive life cycle.

This "Right to Repair" initiative supports projects like converting old servers into firewalls by promoting the reuse and repurposing of hardware, aligning with environmental and cost-saving objectives.

To compare your `freeutm` (a free and open-source UTM system) against other popular firewall solutions like FortiGate, pfSense, Endian, and Cisco firewalls, I will present the information in a table format that incorporates the specific attributes of your system:

Feature	FreeUTM (Your Solution)	FortiGate	pfSense	Endian UTM	Cisco Firewalls
License	Free, Open-source, No paid version, Free License, No restrictions on old servers.	Paid, Commercial	Free, Open-source, Paid support available	Paid, Commercial (with limited free version)	Paid, Commercial (with free trial available)
Deployment	Can be installed on old servers without issue.	Hardware appliance or VM (requires compatible hardware)	Virtual machine or hardware	Hardware appliance or virtual machine	Hardware appliance or virtual machine
Traffic Filtering	Supports connection tracking, custom rules (iptables), and dynamic IP sets.	Advanced traffic filtering (supports deep packet inspection, SSL inspection)	Supports packet filtering, VPN, etc.	Advanced filtering, URL filtering, deep packet inspection	Advanced filtering, SSL inspection, application-level filtering

VPN Support	Supports VPN setup using iptables and additional configurations.	Comprehensive VPN support (SSL, IPsec, L2TP)	Comprehensive VPN support (IPsec, OpenVPN)	IPsec, SSL VPN support	Comprehensive VPN support (IPsec, SSL, AnyConnect)
Intrusion Detection & Prevention (IDS/IPS)	Configures Zeek, Snort, and Maltrail for IDS/IPS monitoring.	Built-in FortiGuard IPS and Antivirus	IDS via Suricata or Snort	Built-in IPS, malware protection	Built-in IPS (FirePOWER, Cisco Umbrella)
Web Filtering	No built-in web filtering.	Advanced web filtering with FortiGuard	Web filtering via Squid and proxy rules	Web filtering and malware protection	Web filtering with Cisco Umbrella
Antivirus	Supports ClamAV, Maldet, and RKhunter for malware scanning.	FortiSandbox for antivirus and sandboxing	ClamAV support	Built-in malware protection	Cisco AMP for endpoints, integrated antivirus
Updates & Maintenance	No paid support, relies on community and manual updates.	Regular firmware and signature updates with FortiCare support	Community updates, optional paid support for updates	Regular updates with commercial support options	Regular firmware updates with Cisco support

Performance	Suitable for older servers; performance depends on system resources.	High performance with dedicated hardware	Good performance, especially on modern hardware	Suitable for small to medium-sized networks	High performance with dedicated hardware
Ease of Use	Command-line configuration (Linux-based), may require technical expertise.	Web-based GUI for easy management	Web-based GUI, very user-friendly	Web-based management console	Web-based management, advanced interface
High Availability (HA)	Not natively supported in the free version (depends on configuration).	Supports Active-Active or Active-Passive HA	Supports HA with CARP	Supports HA (cluster configurations)	Supports HA with failover clustering
Firewall Features	Basic stateful firewall (iptables), advanced rule creation, and logging.	Stateful firewall with advanced capabilities	Stateful firewall with GUI rule management	Stateful firewall with advanced filtering rules	Stateful firewall with advanced rule management
Reporting & Monitoring	Basic logging and monitoring, integrates with tools like Zeek	Advanced logging and reports with FortiAnalyzer	Detailed reporting via RRD graphs and logs	Advanced reporting with centralized	Advanced reporting with Cisco Security Suite

	and Snort for advanced monitoring			management	
SAN (Storage Area Network) Support	Not applicable; storage configurations not part of the firewall system.	Limited, through storage appliances or integration	Not directly applicable	Not directly applicable	Integration with SAN through Cisco solutions
Cost	Completely free, no costs for licensing.	Requires a paid subscription/license	Free, but support and updates are paid	Paid subscription required	Paid subscription required
Sanction Restrictions	No sanctions or geographical restrictions.	Some geo-blocking capabilities	Geo-blocking with additional configuration	Geo-blocking capabilities	Advanced geo-blocking and compliance features

Key Takeaways:

1. **FreeUTM is a completely free, open-source solution with the flexibility to install on older hardware without restrictions.**
2. **Other systems like FortiGate, pfSense, Endian, and Cisco offer advanced features like VPN, IDS/IPS, and web filtering but typically require paid licenses or commercial support.**

3. **FortiGate and Cisco are geared more towards enterprises, offering high availability, dedicated hardware, and advanced security features.**
4. **pfSense and Endian are more flexible with open-source options, but still offer commercial versions for enterprise-scale support.**
5. **FreeUTM doesn’t offer certain enterprise features like SAN support, built-in HA, or geo-blocking out-of-the-box, but it is highly configurable with a wide range of free, community-driven tools.**

This table should give you a clear comparison to showcase the strengths and limitations of FreeUTM versus other popular UTM systems.

---

2. Choosing the Right Linux Distribution

2.1. Minimal Linux Distribution: The first step in creating a firewall is selecting the right operating system (OS). For this purpose, a minimal installation of a stable Linux distribution is recommended.

- **Debian** and **Ubuntu Server** are highly recommended due to their stability, extensive documentation, and large support community. A minimal installation ensures that only the necessary packages are installed, minimizing the system's attack surface.

- **Alternative Options**: If system resources are extremely limited, lightweight distributions like **Lubuntu** or **BunsenLabs** can also be considered, as they are optimized for low-resource environments while still providing robust Linux functionality.

Certainly! Below is some relevant data on e-waste in the European Union, along with key statistics about the scale of the issue. These can serve as a baseline for calculating the impact of converting old servers into firewalls to help mitigate e-waste.

European Union E-Waste Statistics

Statistic	Value	Source/Year

Total E-Waste Generation in the EU	12.3 million metric tons (2019)	European Commission, 2021
Average E-Waste per Person in the EU	16.6 kg per person per year	European Commission, 2021
E-Waste Recycling Rate in the EU	42.5% (2019)	European Commission, 2021
Percentage of E-Waste Sent to Landfills	40%	European Commission, 2021
Projected E-Waste Growth (per year)	4-5% annual increase	Global E-Waste Monitor, 2020
Value of E-Waste Recycling	€55 billion annually (global market)	United Nations University, 2020
Total E-Waste Recycled in the EU	5.2 million metric tons (2019)	European Commission, 2021

How Converting Old Servers into Firewalls Can Help

By repurposing old servers instead of discarding them as e-waste, this project can help address the growing challenge of e-waste. Here's how the project might contribute to reducing e-waste and its environmental impact:

1. **Reduction in E-Waste**: Each server repurposed into a Linux-based firewall prevents it from being discarded as waste. This will help reduce the total volume of e-waste generated by the organization and potentially by others in the broader ecosystem.

2. **Reuse of Components**: Repurposing old servers involves reusing valuable parts, including processors, memory, and storage, which are often made from scarce and precious resources like gold, copper, and rare earth metals. This helps reduce the demand for new devices, curbing the environmental impact of manufacturing and mining.

3. **Extension of Hardware Lifespan**: By extending the useful life of old servers, this project aligns with the EU's "Right to Repair" initiatives, which encourage a longer lifecycle for electronic devices. This results in fewer servers needing to be disposed of or recycled, thereby contributing to a reduction in the total e-waste generated.

Potential Environmental Impact Calculation

To calculate the potential reduction in e-waste, you can use the following approach:

1. **Estimate the number of old servers to be converted**: Suppose you have 100 servers that would otherwise be replaced by new devices. If each server weighs approximately 20 kg and is typically discarded at the end of its lifecycle, this results in:

   $$ 100 \text{ servers} \times 20 \text{ kg/server} = 2000 \text{ kg of e-waste} $$

2. **Estimate the e-waste reduction**: By converting these servers into Linux-based firewalls, you can keep them in use, thus preventing the 2000 kg of e-waste from being generated.

3. **Calculate potential contribution to EU e-waste reduction**: Since the total e-waste generated by the EU in 2019 was 12.3 million metric tons (12.3 billion kg), your project's impact in terms of e-waste reduction can be calculated as a percentage of the total e-waste produced:

   $$ \frac{2000 \text{ kg of e-waste prevented}}{12,300,000,000 \text{ kg total e-waste}} \times 100 = 0.00001626\% $$

While this impact may seem small on a global scale, this is just one organization’s contribution. If scaled across multiple organizations, the collective impact on reducing e-waste can become significant.

Conclusion

By converting old servers into Linux-based firewalls, the project helps reduce the environmental impact of e-waste. Even though individual contributions might seem minimal, when aggregated, they can make a notable difference in the fight against growing e-waste in the European Union. Additionally, aligning with the EU's "Right to Repair" measures strengthens the case for maximizing hardware lifecycles, reducing the need for new devices, and minimizing electronic waste.

---

3. Configuring the Firewall with `iptables`

3.1. Overview of `iptables`: `iptables` is a powerful and flexible utility for managing network traffic rules on a Linux-based firewall. It operates by defining

rules for incoming and outgoing packets, allowing or blocking traffic based on criteria such as IP address, protocol, and port number.

3.2. Basic Firewall Configuration: To secure the server, the following basic `iptables` commands are applied to control network traffic:

```
sudo iptables -A INPUT -m conntrack --ctstate
ESTABLISHED,RELATED -j ACCEPT
sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT  # Allow SSH
sudo iptables -A INPUT -j DROP
```

- The first rule allows established and related connections, which are necessary for maintaining active sessions.
- The second rule specifically allows SSH traffic on port 22, enabling remote management of the firewall.
- The third rule denies all other incoming traffic, ensuring that only explicitly allowed connections are accepted.

3.3. Optimization of Rules: To minimize overhead, it is crucial to avoid unnecessary rules. Efficient rule ordering is key, where more frequently used rules are placed at the top to reduce processing time. The following example uses `conntrack` for connection tracking, making it efficient by grouping related packets:

```
iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j
ACCEPT
```

3.4. Persistence of Rules: To ensure that the rules persist across reboots, use the `iptables-persistent` package:

```
sudo apt install iptables-persistent
```

```
sudo netfilter-persistent save
```

```
sudo netfilter-persistent reload
```

This configuration saves the current `iptables` rules and ensures they are reloaded upon system restart.

To configure `iptables` as a router on a Linux machine, you essentially need to set up IP forwarding and configure appropriate NAT (Network Address Translation) rules for routing traffic between different network interfaces. This setup allows the Linux machine to forward packets between networks, effectively routing traffic and acting as a gateway.

Step-by-Step Configuration:

1. Enable IP Forwarding

IP forwarding is the key to enabling the machine to route traffic between different networks.

To enable IP forwarding temporarily (until the next reboot), run the following command:

```
 sysctl -w net.ipv4.ip_forward=1
```

To make the change permanent, edit the `sysctl.conf` file:

```
vi /etc/sysctl.conf
```

Find the line `#net.ipv4.ip_forward = 1` and uncomment it (remove the #), or add it if it's not there:

```
net.ipv4.ip_forward = 1
```

Then apply the changes:

```
sysctl -p
```

2. Configure Network Interfaces

Assume the Linux machine has two network interfaces:

- `eth0`: The external interface (connected to the Internet or upstream network).
- `eth1`: The internal interface (connected to the local network).

You need to assign IP addresses to these interfaces. This can either be done manually via the `ifconfig` or `ip` command, or automatically via DHCP.

For example, to assign static IP addresses:

```
ifconfig eth0 192.168.1.1 netmask 255.255.255.0
ifconfig eth1 192.168.2.1 netmask 255.255.255.0
```

3. Set Up NAT (Network Address Translation)

To allow machines on your internal network (connected to `eth1`) to access the external network (connected to `eth0`), you'll need to set up NAT.

Use the `iptables` command to enable NAT on the external interface (`eth0`). This will masquerade internal IP addresses (from `eth1`) as the external IP of `eth0`.

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

This rule translates all outgoing packets from the internal network to appear as though they are coming from the external network's IP address.

4. Allow Forwarding of Traffic

You need to allow packets to be forwarded between the interfaces (`eth0` and `eth1`). Add the following `iptables` rules to allow forwarding between the interfaces.

To allow traffic from the internal network (`eth1`) to reach the external network (`eth0`), add this rule:

```
iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT
```

To allow traffic from the external network (`eth0`) to reach the internal network (`eth1`), add this rule:

```
 iptables -A FORWARD -i eth0 -o eth1 -m state --state
RELATED,ESTABLISHED -j ACCEPT
```

5. Save iptables Configuration

By default, iptables rules will not persist across reboots. To make sure your configuration is saved:

On most Linux distributions, you can save the iptables configuration using:

```
 iptables-save > /etc/iptables/rules.v4
```

Or, if you're using a distribution like Red Hat or CentOS, use the service command to save:

```
 service iptables save
```

6. Testing the Routing Setup

To test the routing:

- From a machine in the internal network (connected to `eth1`), try pinging an external host (e.g., `ping 8.8.8.8`).
- The Linux machine should route the traffic from the internal machine to the external network and return the response.

Example `iptables` Script for Routing

Here's an example shell script that configures iptables as a basic router:

```
#!/bin/bash

# Enable IP forwarding
sysctl -w net.ipv4.ip_forward=1

# Configure NAT (Masquerading)
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

# Allow forwarding from eth1 (internal) to eth0 (external)
iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT

# Allow established connections to be forwarded back from eth0
to eth1
iptables -A FORWARD -i eth0 -o eth1 -m state --state
RELATED,ESTABLISHED -j ACCEPT

# Save iptables rules
iptables-save > /etc/iptables/rules.v4
```

Make the script executable and run it:

```
chmod +x /path/to/iptables-router.sh
```

```
./path/to/iptables-router.sh
```

Additional Notes:

- **Firewall Considerations:** You may need to add additional firewall rules depending on your security requirements. For example, to block certain traffic types, you can specify `DROP` or `REJECT` rules in the `iptables` configuration.
- **DHCP Server:** If you want the router to assign IP addresses to devices on the internal network, you can set up a DHCP server like `isc-dhcp-server`.

This setup provides basic routing capabilities using `iptables`. More complex setups may require additional configurations, such as advanced routing, VPNs, or more granular firewall rules.

create Zone and Manage it with Iptables and ipset

Creating zones with `iptables` and `ipset` involves using `ipset` to define groups of IP addresses (or entire networks) and then applying `iptables` rules that reference these groups (i.e., "zones"). By combining these two tools, you can create more granular and efficient firewall configurations that mimic zones.

Steps to Create and Manage Zones with `iptables` and `ipset`

Here's a step-by-step guide on how to define and manage zones using `ipset` and `iptables`.

1. Install and Set Up `ipset`

`ipset` is a utility that allows you to create sets of IP addresses or networks, which can then be used in `iptables` rules.

Install `ipset` (if not already installed)

```
sudo apt update
sudo apt install ipset     # On Debian/Ubuntu
```

2. Create IP Sets for Zones

You can create sets for different zones such as trusted, untrusted, or blocked. For example, let's create a set for trusted IPs and untrusted IPs.

Create `ipset` for Trusted Zone (Internal network)

```
sudo ipset create trusted_zone hash:ip
sudo ipset add trusted_zone 192.168.1.0/24  # Add trusted
internal network
```

Create `ipset` for Untrusted Zone (External network)

```
sudo ipset create untrusted_zone hash:ip
sudo ipset add untrusted_zone 203.0.113.0/24  # Add untrusted
```

```
external network
```

You can also add individual IPs to the sets:

```
sudo ipset add trusted_zone 192.168.1.10  # Add a specific trusted IP
```

3. Create `iptables` Rules Using `ipset`

Once you have created `ipset` sets (which will act as your zones), you can use these sets in `iptables` rules.

Example: Allow Traffic from Trusted Zone

Allow traffic from the `trusted_zone` to your server:

```
sudo iptables -A INPUT -m set --match-set trusted_zone src -j ACCEPT
```

Example: Block Traffic from Untrusted Zone

Drop traffic from the `untrusted_zone`:

```
sudo iptables -A INPUT -m set --match-set untrusted_zone src -j DROP
```

Example: Allow Specific Ports for Trusted Zone

Allow SSH (port 22) and HTTP (port 80) only from the `trusted_zone`:

```
sudo iptables -A INPUT -m set --match-set trusted_zone src -p
tcp --dport 22 -j ACCEPT
sudo iptables -A INPUT -m set --match-set trusted_zone src -p
tcp --dport 80 -j ACCEPT
```

Example: Block All Incoming Traffic from Untrusted Zone

You can block all incoming traffic from the `untrusted_zone`:

```
sudo iptables -A INPUT -m set --match-set untrusted_zone src -j
DROP
```

4. Saving `ipset` and `iptables` Rules

When you create `ipset` sets and `iptables` rules, they will not persist after a reboot unless saved. Here's how to ensure they persist:

Save `ipset` Sets

To make sure your `ipset` sets persist across reboots, you need to save them:

Save the `ipset` configuration to a file:

```
sudo ipset save > /etc/iptables/rules.v4
```

You can add a command to restore the `ipset` configuration at boot time. Add this to a startup script (e.g., `/etc/rc.local` on systems that support it):

```
ipset restore < /etc/iptables/rules.v4
```

Save `iptables` Rules

On most Linux systems, `iptables` rules are saved with `iptables-save` and restored with `iptables-restore`:

Save the `iptables` configuration:

```
sudo iptables-save > /etc/iptables/rules.v4
```

Restore `iptables` rules at boot by configuring your system's startup process. On systems using `systemd`, you can create a systemd service to restore the rules, or use `iptables-persistent` if available.

To install `iptables-persistent` (on Debian/Ubuntu):

```
sudo apt-get install iptables-persistent
```

This will prompt you to save your current `iptables` rules and automatically restore them on boot.

5. Managing Zones Dynamically

Once you have your zones set up, you can manage them dynamically by adding or removing IP addresses from the `ipset` sets as needed. This allows for on-the-fly modifications without needing to change your `iptables` rules.

Adding IPs to an Existing Zone (e.g., trusted zone):

```
sudo ipset add trusted_zone 192.168.2.0/24  # Add a new trusted network
```

Removing IPs from a Zone:

```
sudo ipset del trusted_zone 192.168.1.10  # Remove a specific IP from trusted zone
```

Check the Contents of a Zone:

```
sudo ipset list trusted_zone  # List all IPs in the trusted zone
```

6. Example: Combining Multiple Zones

You can have multiple zones such as `internal`, `dmz`, `external`, etc., and apply different rules to each zone.

Example: Set up three zones

Create sets for multiple zones:

```
sudo ipset create internal_zone hash:ip
sudo ipset create dmz_zone hash:ip
sudo ipset create external_zone hash:ip
```

Add IPs to these zones:

```
sudo ipset add internal_zone 192.168.1.0/24
sudo ipset add dmz_zone 10.1.1.0/24
sudo ipset add external_zone 198.51.100.0/24
```

Apply firewall rules based on these zones:

Allow access from the `internal_zone` to `dmz_zone` (e.g., allow HTTP traffic from internal to DMZ):

```
 sudo iptables -A FORWARD -m set --match-set internal_zone src
-m set --match-set dmz_zone dst -p tcp --dport 80 -j ACCEPT
```

Block all traffic from the `external_zone`:

```
sudo iptables -A INPUT -m set --match-set external_zone src -j
DROP
```

Summary

To create and manage zones with `iptables` and `ipset`:

1. **Use `ipset` to define sets of IP addresses or networks (acting as zones)**.
2. **Use `iptables` to reference those sets and apply rules to traffic**.
3. **Manage IP addresses dynamically by adding/removing them from the sets**.
4. **Ensure persistence across reboots by saving `ipset` and `iptables` configurations**.

This approach gives you powerful flexibility in managing complex firewall configurations based on zones.

Example for Koosha Zone :

To create a "Koosha" zone using `ipset` and `iptables`, you can follow the steps below. This will involve creating an `ipset` named `koosha_zone` and then applying firewall rules using `iptables` to manage traffic for that zone. Here's how you can do it:

Step 1: Install `ipset` and `iptables` (if not already installed)

Make sure `ipset` and `iptables` are installed on your system:

For Debian/Ubuntu:

```
sudo apt update
sudo apt install ipset iptables
```

Step 2: Create the "Koosha" Zone Using `ipset`

1. **Create the `ipset` for the "Koosha" zone**. We'll create a `hash:ip` type set called `koosha_zone`, which can hold IP addresses (or networks) you want to include in the "Koosha" zone.

```
sudo ipset create koosha_zone hash:ip
```

2. **Add IP addresses or networks to the "Koosha" zone**. You can add individual IPs or networks that belong to the "Koosha" zone.

For example, if you want to add a specific IP or subnet:

```
sudo ipset add koosha_zone 192.168.1.100    # Add a specific IP
sudo ipset add koosha_zone 10.0.0.0/24      # Add an entire
```

```
subnet
```

3. **Verify the contents of the `koosha_zone`:**

```
sudo ipset list koosha_zone
```

Step 3: Create `iptables` Rules for the Koosha Zone

Now that you've created the `koosha_zone` set in `ipset`, you can use `iptables` to apply firewall rules to the traffic from or to that zone.

Example 1: Allow Traffic from the "Koosha" Zone

If you want to allow traffic from the `koosha_zone` (e.g., from trusted IPs), you can create an `iptables` rule like this:

```
sudo iptables -A INPUT -m set --match-set koosha_zone src -j
ACCEPT
```

This rule allows incoming traffic from any IP in the `koosha_zone`.

Example 2: Block Traffic from the "Koosha" Zone

If you want to block incoming traffic from the `koosha_zone`:

```
sudo iptables -A INPUT -m set --match-set koosha_zone src -j
DROP
```

This rule will block any traffic from the IPs listed in the `koosha_zone`.

Example 3: Allow Specific Ports from the "Koosha" Zone

You might want to allow traffic from the `koosha_zone` only for specific services or ports (e.g., SSH on port 22 or HTTP on port 80). Here's how you can do it:

```
sudo iptables -A INPUT -m set --match-set koosha_zone src -p tcp
--dport 22 -j ACCEPT   # Allow SSH
sudo iptables -A INPUT -m set --match-set koosha_zone src -p tcp
--dport 80 -j ACCEPT   # Allow HTTP
```

Example 4: Deny All Traffic from the "Koosha" Zone

If you want to deny all incoming traffic from the `koosha_zone` but allow other zones or services:

```
sudo iptables -A INPUT -m set --match-set koosha_zone src -j
REJECT
```

This will reject all traffic from the `koosha_zone` while still allowing traffic from other sources.

Step 4: Save `ipset` and `iptables` Rules (Optional)

To ensure the `ipset` and `iptables` rules persist across reboots, you'll need to save them.

Save `ipset` rules:

```
sudo ipset save > /etc/iptables/rules.v4   # Save ipset to a
file
```

You can add a command to restore the `ipset` rules at boot time, for example, by adding the restore command in `/etc/rc.local` (on systems that support it):

```
ipset restore < /etc/iptables/rules.v4
```

Save `iptables` rules:

You can save the `iptables` rules using `iptables-save`:

```
sudo iptables-save > /etc/iptables/rules.v4
```

For systems using `iptables-persistent` (on Debian/Ubuntu), you can install the package to ensure rules persist:

```
sudo apt-get install iptables-persistent
```

This will automatically restore your rules on boot.

Step 5: Verify and Test the Rules

After setting up the "Koosha" zone and applying the rules, it’s important to verify that everything is working correctly.

Check the `ipset` list:

```
sudo ipset list koosha_zone
```

This will display all IPs in the `koosha_zone` set.

Check the `iptables` rules:

```
sudo iptables -L -v -n
```

This will show the current `iptables` rules along with the packet counts.

Example Summary of Steps

1. **Create `ipset` for the Koosha Zone:**

```
sudo ipset create koosha_zone hash:ip
```

2. **Add IPs or subnets to the Koosha Zone:**

```
sudo ipset add koosha_zone 192.168.1.100
sudo ipset add koosha_zone 10.0.0.0/24
```

3. **Create `iptables` rules for the Koosha Zone:**

Allow traffic from the zone:

```
 sudo iptables -A INPUT -m set --match-set koosha_zone src -j
ACCEPT
```

Block traffic from the zone:

```
 sudo iptables -A INPUT -m set --match-set koosha_zone src -j
DROP
```

4. **Save the `ipset` and `iptables` configurations:**

```
sudo ipset save > /etc/iptables/rules.v4
sudo iptables-save > /etc/iptables/rules.v4
```

5. **Verify the rules**:

```
sudo ipset list koosha_zone
sudo iptables -L -v -n
```

By following these steps, you can successfully create and manage a "Koosha" zone with `ipset` and `iptables`, which will allow you to control traffic from specific IPs or networks effectively.

---

4. Intrusion Detection and Prevention Systems (IDS/IPS)

4.1. PSAD: Port Scan Attack Detector

PSAD is a useful tool for detecting and reacting to port scans, a common tactic used by attackers to probe for vulnerabilities. After installing and configuring PSAD, the system can automatically block IP addresses involved in scanning attempts.

```
sudo apt install psad -y
sudo psad --sig-update
sudo systemctl restart psad
```

The configuration file (`/etc/psad/psad.conf`) should be updated to enable automatic IDS on specific ports:

```
ENABLE_AUTO_IDS Y;
AUTO_IDS_TCP_PORTS any;
AUTO_IDS_UDP_PORTS any;
```

4.2. Zeek (Formerly Bro): Zeek is a comprehensive network monitoring tool used to analyze network traffic, identify anomalies, and detect malicious activities. Installing Zeek on a Linux server provides detailed network analysis and can act as a robust IDS.

```
sudo apt install curl gnupg2 wget -y
```

```
curl -fsSL
https://download.opensuse.org/repositories/security:zeek/xUbuntu
_22.04/Release.key | gpg --dearmor | tee
/etc/apt/trusted.gpg.d/security_zeek.gpg
```

```
echo 'deb
http://download.opensuse.org/repositories/security:/zeek/xUbuntu
_22.04/ /' | tee /etc/apt/sources.list.d/security:zeek.list
```

```
sudo apt update -y
```

```
sudo apt install zeek -y
```

Zeek's capabilities go beyond simple IDS and include detailed logs for forensic analysis.

4.3. Snort: Snort is another highly effective IDS/IPS tool for real-time traffic analysis. It is particularly useful for detecting and blocking malicious network activity, such as exploitation attempts, malware traffic, and denial-of-service (DoS) attacks.

```
sudo apt install snort -y
```

```
sudo snort -c /etc/snort/snort.conf -i eth0
```

To enable community rules for Snort:

```
wget
https://www.snort.org/downloads/community/community-rules.tar.gz
```

```
tar -xvzf community-rules.tar.gz -C /etc/snort/rules/
```

---

5. Malware Detection and Prevention

5.1. ClamAV: ClamAV is an open-source antivirus engine for detecting trojans, viruses, and other malicious threats. Regular scanning and automatic removal of detected files can significantly enhance the server's security posture.

```
sudo apt install clamav -y
```

```
0 2 * * * /usr/bin/clamscan --remove --recursive --infected /
--exclude-dir="^/sys" --exclude-dir="^/proc"
--log=/var/log/clamav-scan.log
```

5.2. Maldet (Linux Malware Detect): Maldet is another useful tool designed to scan for malware. It provides additional features such as email notifications and quarantine for infected files.

```
cd /tmp && wget
http://www.rfxn.com/downloads/maldetect-current.tar.gz && tar
xfz maldetect-current.tar.gz && cd maldetect-1.6.* && sudo
./install.sh
```

---

6. Brute Force Protection

6.1. Fail2ban: Fail2ban helps mitigate brute force attacks by monitoring log files for suspicious login attempts and banning the corresponding IP addresses after a configurable number of failed attempts.

```
sudo apt install fail2ban -y
```

```
sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
```

The following configuration blocks IPs after five failed login attempts within 10 minutes:

```
ignoreip = 127.0.0.1/8 ::1 192.168.1.0/24
bantime  = 1d
findtime  = 10m
maxretry = 5
```

---

7. System Hardening

7.1. Secure SSH Configuration: To protect the firewall from unauthorized SSH access, it is important to disable root login.

```
sudo vi /etc/ssh/sshd_config
```

```
PermitRootLogin no
```

7.2. Unattended Security Updates: Enabling automatic security updates ensures that the system remains protected against known vulnerabilities without requiring manual intervention.

```
sudo apt install unattended-upgrades -y
sudo dpkg-reconfigure unattended-upgrades
```

---

8. Monitoring and Logging

8.1. Netdata Monitoring: Netdata provides real-time monitoring of system performance, including CPU, RAM, network traffic, and disk usage. Its detailed visual interface helps identify issues before they become critical.

sudo bash <(curl -Ss https://my-netdata.io/kickstart.sh)

8.2. Log Management: Ensure that logs from security tools like `iptables`, `psad`, and `fail2ban` are centralized and stored securely for audit and forensic analysis.

---

9. Conclusion

Repurposing old servers into next-generation firewalls can effectively safeguard network infrastructures without incurring significant costs. By implementing a robust suite of security tools, such as `iptables`, **PSAD**, **Zeek**, **Snort**, and **Fail2ban**, the server can become a proactive defender against cyber threats. Additionally, regular updates, malware scanning, and system hardening ensure that the firewall remains resilient in the face of evolving security challenges.

This approach not only maximizes the use of legacy hardware but also provides an efficient, cost-effective solution to network security, leveraging the power of open-source tools and Linux’s inherent flexibility and robustness.