IP Shuffle:
Random IP Address Assignment for Network Interfaces

Hunter Thompson
Eastern Washington University

Chelsea Edwards
Eastern Washington University

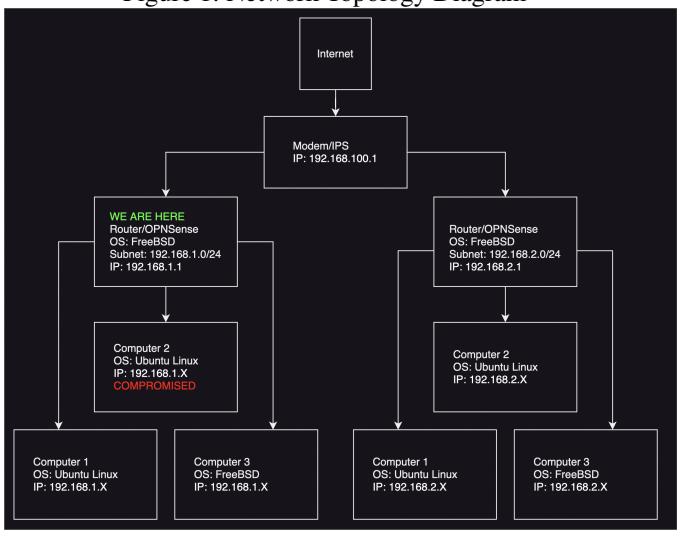
Abstract

This paper introduces a Bash script designed as a Moving Target Defense (MTD) mechanism for dynamically assigning a random IP address to a computer's network interface, thereby complicating lateral movement reconnaissance. The IP-shuffle script generates a random IP address within a specified range, verifies its availability, and ensures proper configuration. By unpredictably rotating IP addresses within a subnet, the IP-shuffle script disrupts attackers' ability to establish a static view of the network, making reconnaissance challenging. This paper evaluates the impact of IP-shuffle in mitigating network reconnaissance and lateral movement, offering insights into its effectiveness as a proactive defense strategy.

1 Introduction

In the rapidly evolving field of cybersecurity, attackers constantly refine their reconnaissance and lateral movement techniques to compromise networked systems. Moving Target Defense (MTD) strategies have emerged as a proactive solution to complicate and thwart such attacks by introducing uncertainty and unpredictability into network operations. One such MTD technique is known as IP shuffling, which involves dynamically changing the IP addresses of systems within a network to impede reconnaissance. This paper introduces the `ip-shuffle` script, a Bash-based tool that dynamically assigns random IP addresses to a computer's network interface within a specified range, verifies its availability, and ensures proper configuration. By unpredictably rotating IP addresses within a subnet, the `ip-shuffle` script disrupts attackers' ability to establish a static view of the network, making reconnaissance challenging. The script achieves efficient and reliable IP address assignment through distinct functions for IP address generation, availability verification, network configuration validation, and gateway reachability testing. The `ip-shuffle` script incorporates comprehensive error handling and compatibility with Linux and BSD systems to provide a basic solution for scenarios requiring dynamic IP address allocation.

Figure 1: Network Topology Diagram



2 Threat Model

Our threat model involves a scenario where an adversary has successfully compromised a company device within a subnet on the company network. Figure 1 depicts the network topology in which this threat model takes place: three interconnected computers form a subnet, with one of these computers already compromised. In this scenario, the attacker is assumed to possess the following capabilities:

- **Basic User Access:** The attacker has basic user privileges on the compromised system.
- **Network Reconnaissance:** The attacker can perform network reconnaissance by scanning the network.

- **System Persistence:** The attacker can maintain persistent access to the compromised device.

Given these capabilities, the attacker seeks to gain valuable reconnaissance information, identify other devices on the network, and exploit any discovered vulnerabilities that could facilitate lateral movement. The `ip-shuffle` script aims to counter these activities by dynamically assigning random IP addresses to network interfaces, making it difficult for the attacker to establish a static view of the network and impeding their ability to conduct effective reconnaissance.

3 System Design

The `ip-shuffle` script provides a systematic approach to dynamic IP address assignment for network interfaces in Linux and FreeBSD environments. Built around Bash scripting, it orchestrates the IP address allocation process seamlessly. By default, the program runs every three minutes based on a cronjob, dynamically configuring the IP address, gateway, and network interface details. This ensures an efficient and flexible network configuration.

3.1 Components and Functions

The core of the `ip-shuffle` script relies on modular functions that handle IP address generation, availability verification, and network configuration validation. The primary functions include:

- **`generate_random_ip()`:** Generates a random IP address within a specified subnet range.
- **`check_ip_availability()`:** Verifies whether the generated IP address is available using the `ping` command.
- **`validate_network_config()`:** Validates that the newly assigned IP address works correctly within the network by checking gateway reachability.
- **`reset_network()`:** Resets network configurations in case of errors to restore connectivity.

3.2 IP Address Assignment Workflow

The workflow of the `ip-shuffle` script follows these steps:

1. **Generate a Random IP Address:** The script uses `generate_random_ip()` to produce a new IP address.
2. **Check Availability:** The script verifies if the generated IP address is available using `check_ip_availability()`.
3. **Configure Network Interface:** Depending on the OS, the script will either configure a specified interface using `ip` or `ifconfig`.
4. **Validate Network Configuration:** The script ensures proper network configuration by testing gateway reachability via `validate_network_config()`.
5. **Reset Network if Needed:** If the new IP address is invalid or unreachable, the `reset_network()` function is called to restore network connectivity.

3.3 Scheduling with Cron

To ensure regular IP address rotation, the `ip-shuffle` script is scheduled with a cronjob that runs every three minutes:

```
# Recommended placement: /usr/local/sbin/ip-shuffle
*/3 * * * * /path/to/ip-shuffle
```

This systematic rotation of IP addresses complicates reconnaissance and lateral movement for potential attackers, forcing them to continually rescan the network.

3.4 Error Handling and Signal Support

The `ip-shuffle` script incorporates error-trapping mechanisms and Unix signal support to enhance reliability and resilience. Common Unix signals such as `SIGINT` and`SIGTERM` are handled gracefully, allowing the script to clean up network configurations if interrupted unexpectedly. This ensures that network configurations remain intact even in the face of unexpected interruptions.

4 Evaluation

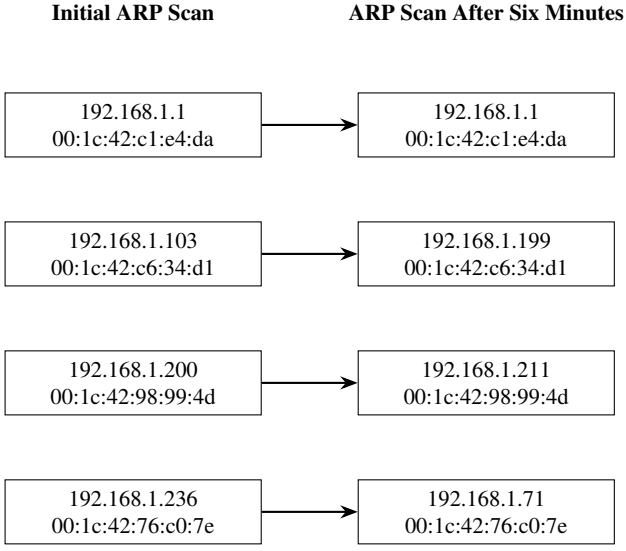
To evaluate the effectiveness of our IP shuffling script, we'll be adding an extra system to the virtual network with a static IP address of `192.168.1.10`. We'll set up an OPNsense instance as the default gateway for all virtual machines. This instance will be assigned the IP address `192.168.1.1`, acting as a simulated router that provides the DHCP service. As shown in Figure 1, there will be two Ubuntu Linux machines and one instance of FreeBSD. Each machine will receive an IP address on initial startup starting at `192.168.1.100`, assigned by OPNsense. We'll give each system six minutes after startup to begin changing its IP address, after which we'll use an arp-scan using our extra machine to obtain the following output:

```
192.168.1.1     00:1c:42:c1:e4:da       (Unknown)
192.168.1.103   00:1c:42:c6:34:d1       (Unknown)
192.168.1.200   00:1c:42:98:99:4d       (Unknown)
192.168.1.236   00:1c:42:76:c0:7e       (Unknown)
```

We can observe that the IP addresses, with the exception of the OPNsense instance, have been altered from their initially assigned addresses through DHCP. To check if the systems have changed their IPs once more, let's wait for another six minutes and then conduct another ARP scan. Here are the results of the said scan:

```
192.168.1.1     00:1c:42:c1:e4:da       (Unknown)
192.168.1.71    00:1c:42:76:c0:7e       (Unknown)
192.168.1.199   00:1c:42:c6:34:d1       (Unknown)
192.168.1.211   00:1c:42:98:99:4d       (Unknown)
```

Let's cross-check the modified IP addresses with their respective MAC addresses to better understand their altered IPs.



4.1 Analysis and Observations

Analyzing the IP shuffling technique involves evaluating its ability to prevent attackers from gaining valuable network reconnaissance information. Here's a summary of our observations:

- **IP Address Changes:** The IP addresses changed significantly over the two arp-scans, making it difficult for an attacker to establish a static network view.
- **MAC Address Consistency:** Each machine retained its MAC address throughout the scans, but the association between MAC addresses and IP addresses changed dynamically.
- **Impact on Reconnaissance:** An attacker may struggle to perform effective reconnaissance and lateral movement as the IP addresses of potential targets keep changing, requiring constant rescanning of the subnet.

4.2 Future Work and Limitations

- **MTD Evasion Analysis:** Attackers may attempt to fingerprint devices based on other network characteristics like latency or open ports. Future work could include an analysis of such evasion techniques.
- **MAC Address Fingerprinting Mitigation:** Although the IP addresses of the machines are shuffled, each machine retains its MAC address, which can be used to fingerprint it. Investigating ways to obscure MAC addresses or to randomly change them in addition to IP shuffling could significantly improve the effectiveness of the Moving Target Defense (MTD) strategy.
- **Impact on Legitimate Users:** Changing IP addresses might also impact legitimate network users. Evaluating how legitimate users handle these changes could be a valuable direction for research.
- **Integration with SDN:** Integration of this MTD technique with Software Defined Networking (SDN) could provide more robust and flexible defense mechanisms.

5 Conclusion

Moving Target Defense (MTD) has been hailed as a revolutionary strategy in cybersecurity that increases complexity and costs for attackers while reducing the exposure of vulnerabilities and enhancing system resilience [**?**]. This paper introduced the `ip-shuffle` script, a robust solution for dynamically allocating random IP addresses to network interfaces, thereby impeding attackers' reconnaissance efforts. The `ip-shuffle` script provides a systematic approach to dynamic IP address assignment through its modular design and comprehensive functionalities, including generating random IP addresses, verifying availability, and validating network configurations. By leveraging error-handling mechanisms and Unix signal responsiveness, the script ensures reliable execution and strengthens network resilience. The evaluation demonstrated the impact of `ip-shuffle` in complicating reconnaissance and lateral movement by continually altering IP addresses within a subnet, making it challenging for attackers to establish a static network view. In future work, the potential of integrating this technique with Software Defined Networking (SDN) could offer more robust and flexible defense mechanisms. Additionally, addressing the limitations of MAC address fingerprinting and evaluating the impact on legitimate network users will further improve this Moving Target Defense strategy. Overall, the `ip-shuffle` script exemplifies proactive defense strategies that make it increasingly difficult for attackers to identify and exploit vulnerabilities.

References