Taking the Elevator down to ring 0

Black Lotus Labs Posted On November 14, 2023

Executive Summary

The Black Lotus Labs team has discovered a highly unique piece of malware designed to compromise the security of the extended Berkeley Packet Filter (eBPF) functionality in the Linux kernel of container-based operating systems, like CoreOS. eBPF is a programmable

framework that allows users to run code within the kernel of Linux systems, without having to write a kernel-specific module. Named “Elevator” by the malware author, it was created to escape the security restrictions of containers and allow the attacker to escalate privileges. Attacking the container’s operating system in this manner allows the threat actor to get root access on the shared server.

Elevator appears to be a multi-stage attack, deployed against web-facing container applications. The first of two stages is designed entirely as a reconnaissance element, to determine which OS and environment it has encountered before sending the primary exploit. Once the exploit is introduced and deployed successfully, the file collects a myriad of host-based enumeration details and sends it back to a remote command-and-control (C2) node. From this point it spawns a remote shell and awaits additional tasking from the C2. We suspect that this tool was meant for use against containerized environments as a mechanism to gain initial access to a targeted network. With this means of entry, an attacker could expand access and move laterally into adjacent networks.

The first defining aspect of this file is that it was compiled to run on two versions of CoreOS and one version of Ubuntu. The CoreOS system was designed in 2013 as a “fundamental building block of distributed systems,” and is useful for deploying containers onto cluster nodes. CoreOS was developed by the company of the same name, and later acquired by RedHat in 2018 to build into the Kubernetes container system. To date there has only been one other instance of this type of attack occurring “in-the-wild,” known as SiloScape, the purpose of which was the targeting of Windows Servers.

This vulnerability follows a similar logic path described in CVE-2018-18445, its abuse allowed an attacker to gain access to the heap memory and escalate privileges to ring 0, or all the way to the highest privilege at the kernel level of the operating system. Investigating further, we found that a sample of Elevator was first uploaded to VirusTotal in December 2018. This sample was submitted just one month after a Linux kernel patch for an eBPF was posted for a similar exploit, described in CVE-2018-18445. This finding suggests that whoever developed and operationalized this sample based the exploit on the details in the CVE and operationalized the exploit prior to a patch being widely implemented.

Technical Details

Introduction

This is only the second instance of publicly reported in-the wild exploitation of container systems that we have discovered after Palo Alto’s report on SiloScape – which differed as they were escaping containers to gain access to Windows servers. The specialization caught our attention, as Kubernetes, or “K8s” systems have become a staple in many cloud computing environments for securely running applications, and, by their very nature, are designed to be exposed to the internet as a perimeter asset. We suspect that Elevator could

serve as an initial access vector that gets deployed once an attacker finds a vulnerability in an application. For an excellent reference on exploiting a vulnerability in the eBPF application, please see Chompie's blog on kernel exploits here.

One of the first things that we noticed about the malware is it implemented a multi-phase attack chain. There was an initial module that would seek out information about the kernel on the impacted machine. We suspect that if the kernel was vulnerable, it would deploy a second module dubbed Elevator. Once invoked, Elevator seeks out kernel information for the impacted machine and then tries to match it to three known operating systems; two of which were variants of CoreOS while the third was a variant of Ubuntu. If the kernel version does not match one of these three options, the file will cease execution. Since the malware hunts for this container-based operating system, we assess that the threat actor likely targeted web applications and then used this file as a tool to escape the container to gain access to the underlying Ubuntu server. The second feature of note is that the sample appears to implement a local privilege escalation exploit in the eBPF of the Linux kernel. This vulnerability shares a logic flow that was similar to the one publicly disclosed in late October 2018 as CVE-2018-18445.

Once the exploit chain was successful, the malware would then perform some host-based enumeration and spawn a remote shell which then connected back to a C2 server. This last command of spawning a remote bash shell suggests that the actor likely had the intention to run subsequent commands on the infected machine. This also indicates this malware family could either be used to gather information about applications running on the server or even be used as a pivot point into the network. We suspect that since the containers themselves are immutable and designed to deal with high bandwidth usage from the applications and customers, they are unlikely to receive the same level of introspection as other perimeter devices. This combination could lead to prolonged network access that is unlikely to be detected.

Malware Analysis

Initial Reconnaissance Module

We suspect this infection process utilized two unique files, both of which were uploaded by the same submitter within minutes of each other. The first module called "bpt.test" was run on the system to determine if the conditions for one of two different privilege escalation vulnerabilities exists. There are two vulnerability checks, the first is set up by calling "prog_load" and attaching to the eBPF program. It is then triggered by writing to the socket. The second vulnerability check only calls prog_load which will attempt to verify and load the program. The return value from the verify/load is what is checked. If the constant can be read from the heap, the preconditions needed to exploit CVE-2018-18445 are valid.

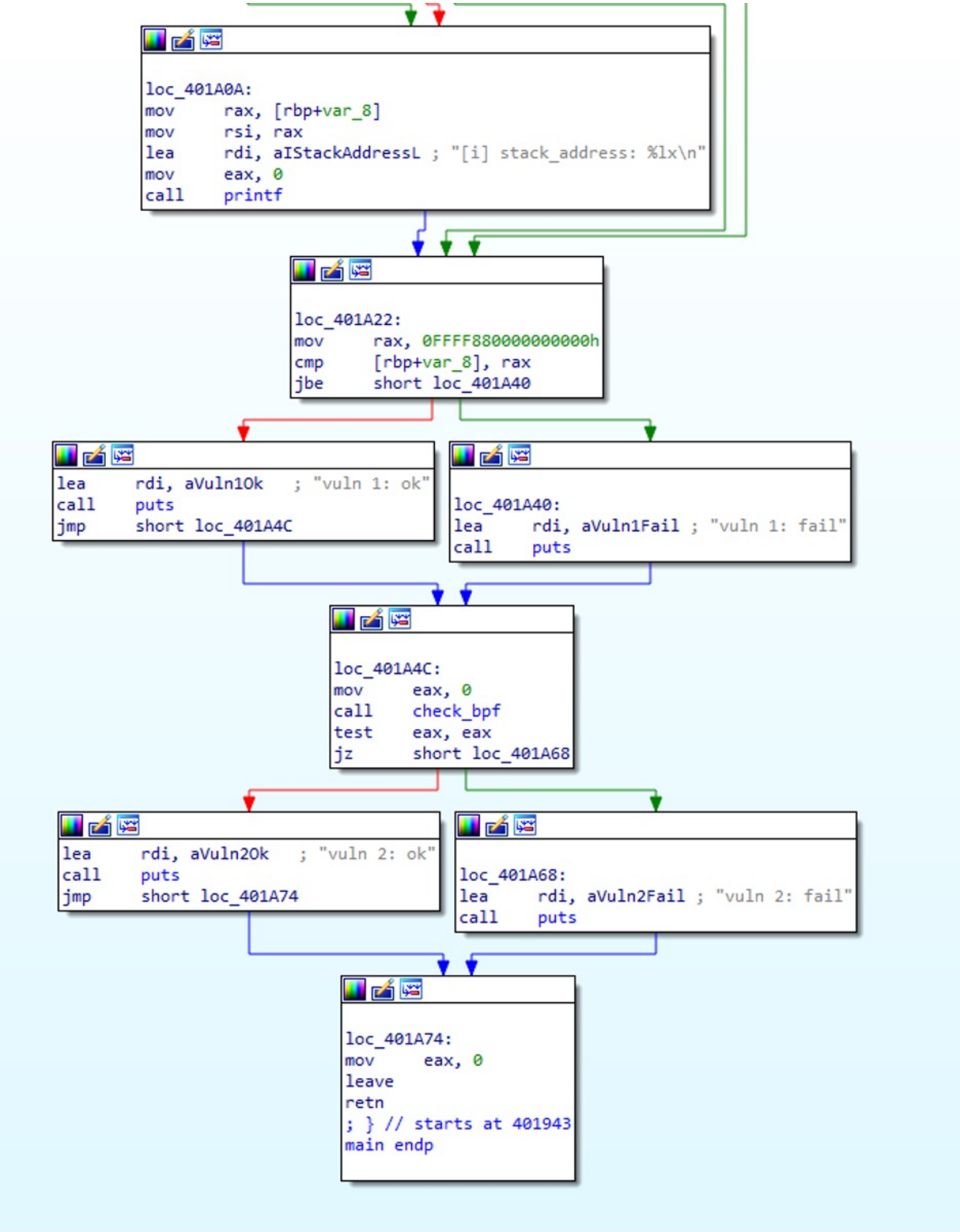

Figure 1: Reconnaissance module flow to check if the two conditions are present that would allow for exploitation

Once the file was run it would report back to confirm if the infected system would be vulnerable to privilege escalation exploit. We suspect that the threat actors employed this module to only deploy their malware on systems that were vulnerable, to limit detectability of their operations.

Elevator: Exploitation of eBPF, likely CVE-2018-18445

Once the binary began its execution chain, one of the first things that it did was call “uname” to gather information about the kernel version running on the infected hosts. It would then check for three specific versions of Linux: 4.15.0-42-generic (Ubuntu), 4.14.63-coreos (CoreOS 1800.7), 4.14.48-coreos-r2 (CoreOS 1745.7). If it is found to be running on one of these systems, it will save the offset to a structure with information related to the version.

Next it creates two bpf_maps and stores the file descriptors to reference eBPF helper functions, which will return pointers to values within the map. Both maps are of type “BPF_MAP_TYPE” with a key size of 4 and a value size of 8, but one has max_entries of 1 and the other has 3. The map with 3 entries does not appear to be referenced anywhere else in the binary. It checks for the environment variable “TEST_ADDR” and if it is set it will create an eBPF program using the variable stored in TEST_ADDR. The eBPF program attempts to call “bpf_get_current_comm” to retrieve the name of the current file and then calls “bpf_map_lookup_elem.”

If TEST_ADDR is not set, the binary will call “capget” to check thread capabilities and echo them: “

[i] admin:0 module:0.” Next, it creates the first eBPF program and attaches the first file descriptor from the socketpair to the eBPF program using setsockopt.

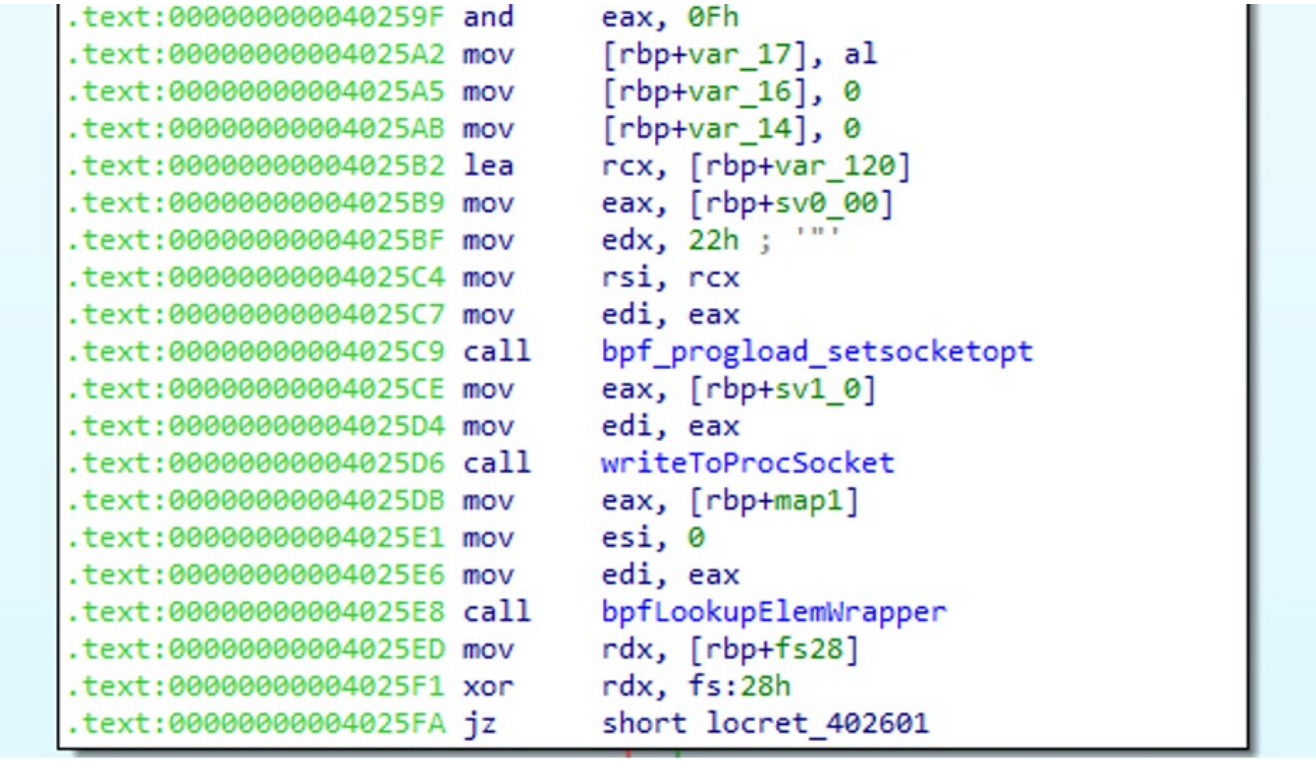

Figure 2:The program creates the eBPF program and attaches the socketpair

To execute the eBPF program, the second file descriptor wrote the letter ‘X’. This program appears to get the kernel stack pointer, based on strings in the binary.

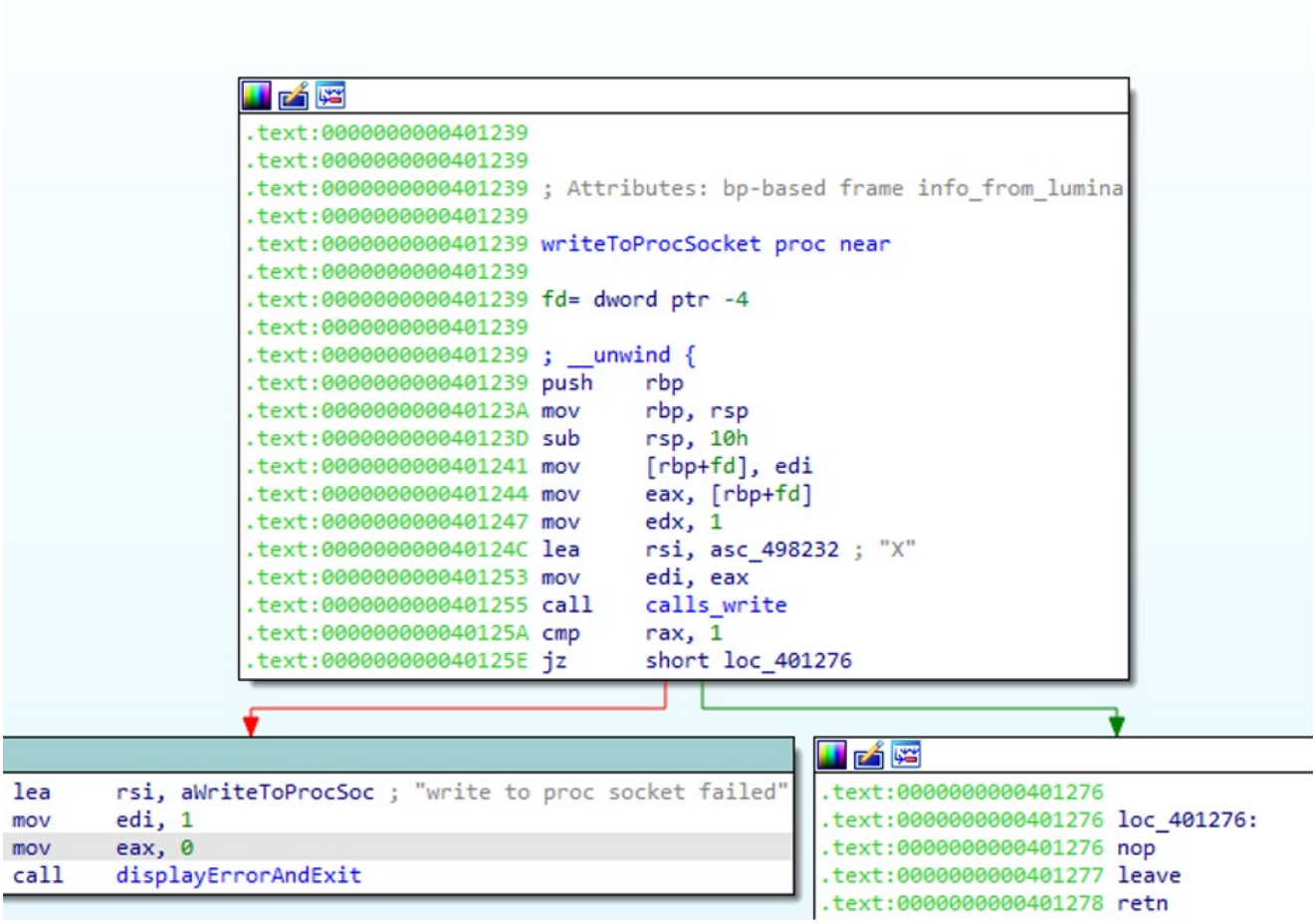

Figure 3: The program executes eBPF by writing the letter "X"

A second eBPF program is loaded and executed using the same method as above. This second program attempts to verify it has found the stack end by comparing the return value with "0x57AC6E9D (STACK_END_MAGIC)." If it finds the stack end it will now check that it did find one of the three Linux versions earlier, and if not exits with message:

[-] unknown kernel

[!] please exec: cat /proc/kallsyms|egrep " init_task| _text"

If STACK_END_MAGIC is found and the binary is running on one of the three Linux versions, it will attempt to leak the kernel base using the two previous eBPF programs. Once the kernel base has been located, it will check for environment variables, "DUMP_ADDR," "DUMP_BY_OFFSET," or "DUMP_CREDS" and will attempt to leak different kernel info depending on the environment variable set.

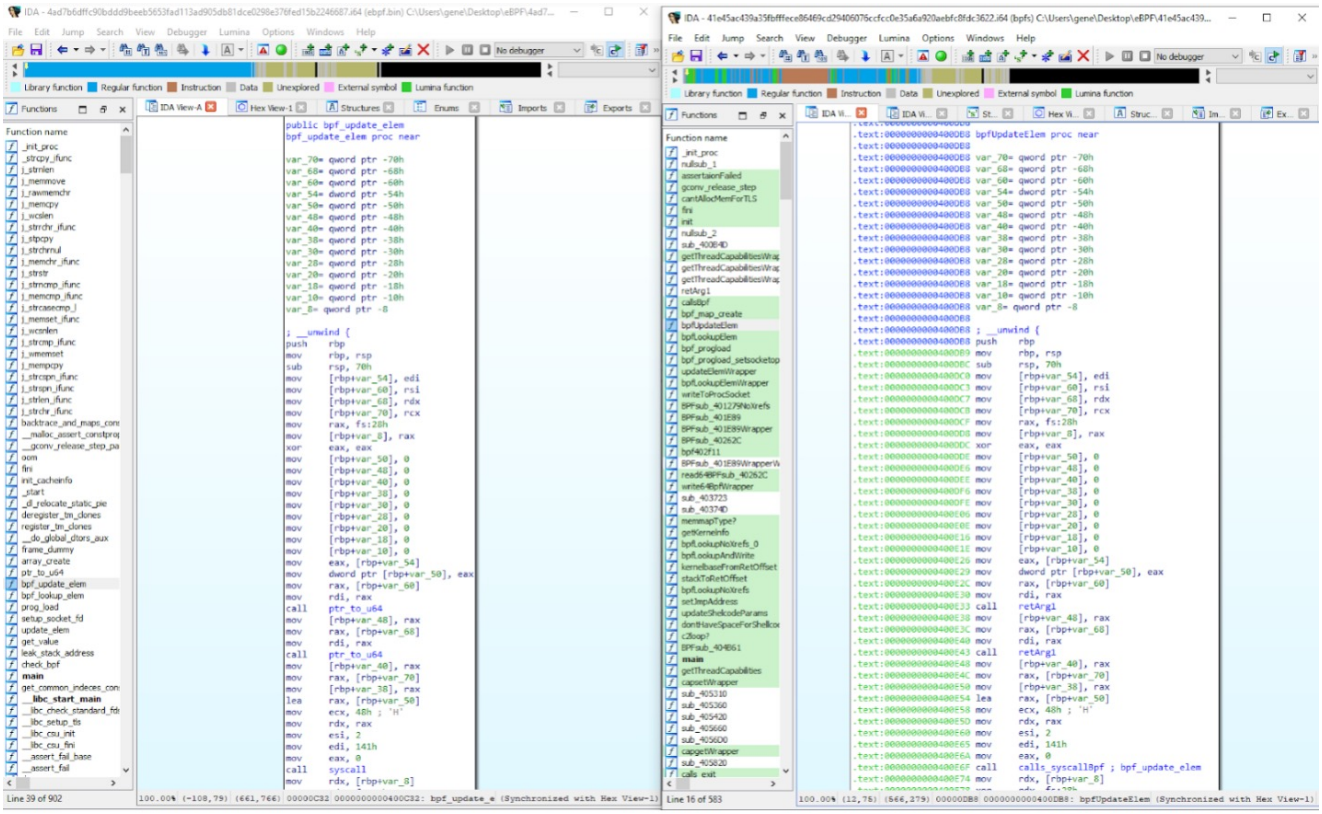

Figure 4: Exploitation logic observed in both the recon and Elevator module

If none of the environment variables are set, it will attempt to elevate privileges by setting UID to 0.

Elevator: Host-Based Enumeration and Remote Shell

If the exploit succeeded in elevating the privilege on the host machine down to ring 0, it would run a small Python script that would enumerate the infected host machine and establish a remote shell. It would gather the following information:

- The main board product UUID, as set by the board manufacturer and encoded in the BIOS DMI information.
- A list of the keys for which the reading thread has view permission, providing various information about each key.
- The machine's hostname.
- A list of running processes on the machine.
- The public IP address of the machine.
- A list of open files on the infected machine.
- Information about the various network interfaces.
- A list of machines connected to the machine on the layer 2 level, along with their MAC addresses.
- Information about the hardware configuration.
- Information about all the PCI buses and devices connected to the system.

- Information about all the USB buses and devices connected to the system.

The exact command line syntax employed by malicious script can be found in the IOC section. Once the data was gathered it was transmitted to a threat actor controlled Virtual Private Server (VPS) located at IP address 198.211.118[.]121:8081. Finally, the Python script would start a remote shell using bash, this allowed the threat actor to remotely interact with the infected systems to run arbitrary commands.

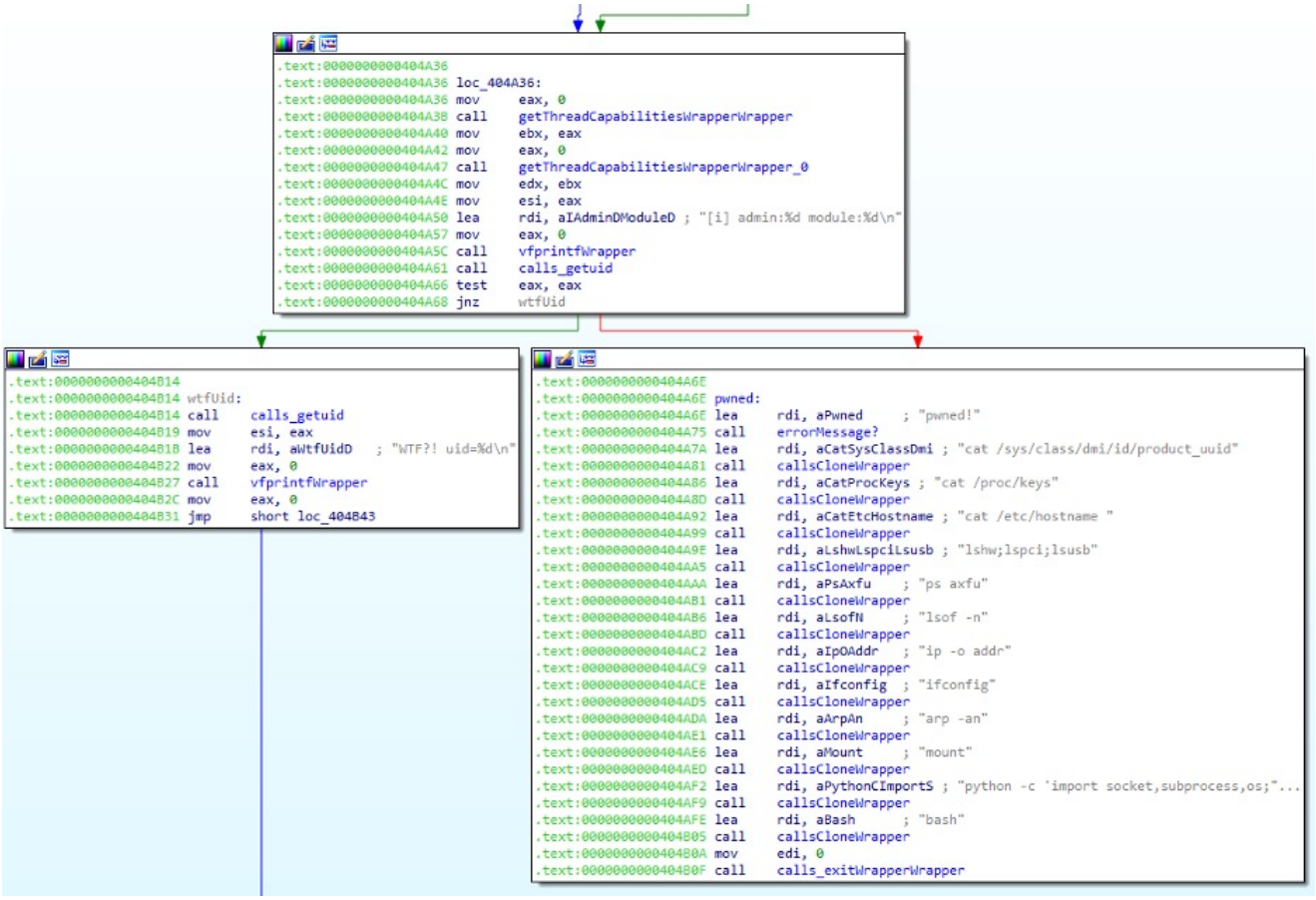

Figure 5: Showing the enumeration performed by the threat actor on the infected machine

Conclusion

While the exploit that was utilized as part of this attack chain was older, we believe that this methodology is still relevant. Since this malware sample was compiled we have seen subsequent research into exploiting eBPF from other security researchers, one of the most notable articles was from “Kernel Pwning with eBPF” that was released in July 2021 discussing CVE-2021-3490. CrowdStrike later theorized that eBPF exploits, such as CVE-2021-3490, could also be used for a container escape to gain access to the host system. We suspect that this fact would hold true for other eBPF vulnerabilities such as CVE-2022-23222. While these priv-esc exploits can be swapped, once exploited we anticipate actors would likely follow the same host-based enumeration pattern. Secondly, it showcases the

ever-present threat to perimeter devices, even when proper steps such as containerization are in place. While the use of containers and K8 reduce the overall attack surface, they still pose a risk as a "single point of failure." In 2023 alone, there are 10 CVEs related to eBPF exploits.

This attack scenario showcases the need for behavior-based detection through applications such as Sysmon for Linux, even for machines traditionally thought of as a security appliance. As this sample managed to evade EDR-based detection for several years until the time of this publication, it demonstrates the need for proper network segmentation for internet facing devices to impede lateral movement into the adjacent LAN. These steps – behavioral based monitoring and network segmentation – help defenders identify this type of malicious activity and prevent the infection from becoming worse even when an attacker employs a known or unknown exploit. We suspect that the malware's capability to gather information about the network interfaces and the contents of the ARP cache, suggests the intention is to move laterally within a target network.

Finally, we recommend fastidious patching of all devices, especially those associated with perimeter facing servers as they are more likely to be probed by attackers. If running or maintaining a K8s cluster, make sure that you are utilizing a supported operating system such as Fedora-CoreOS which is the current open-source alternative.

Black Lotus Labs will continue proactively hunting for interesting malware samples in both public and private repositories. If we can associate this file to any publicly reported activity cluster, we will continue to alert the information security community to these developments. We encourage other organizations to alert on this and similar activity in their environments.

For additional IOCs such as file hashes associated with these campaigns, please visit our GitHub page.

If you would like to collaborate on similar research or have additional information on this activity cluster described above, please contact us on Twitter and Mastodon @BlackLotusLabs.

This analysis was performed by Danny Adamitis and Steve Rudd, technical editing by Ryan English.

Black Lotus Labs

Author

Black Lotus Labs

The mission of Black Lotus Labs is to leverage our network visibility to help protect customers and keep the internet clean.

Trending Now

You may also like

Services not available everywhere. ©2022 Lumen Technologies. All Rights Reserved.