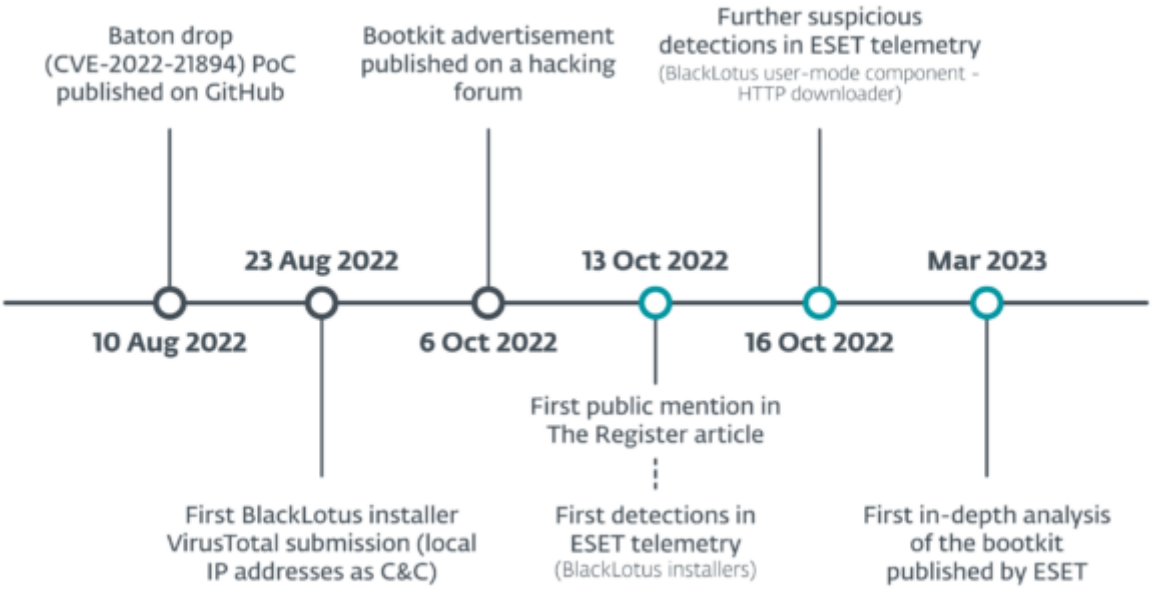
Black lotus - UEFI Boot attack

Historique



Description

There are three main sections in the chain:

1. An installer deploys files to the ESP, as shown in step 1 in the above figure. The installer then disables HVCI and BitLocker and reboots the device. The installer appears to have two versions—one with embedded vulnerable binaries and another that downloads them directly from Microsoft. The latter installer version downloads binaries, including:

- https://msdl.microsoft.com/download/symbols/bootmgfw.efi/7144BCD31C0000/bootmgfw.efi
- https://msdl.microsoft.com/download/symbols/bootmgr.efi/98B063A61BC000/bootmgr.efi
- https://msdl.microsoft.com/download/symbols/hvloader.efi/559F396411D000/hvloader.efi

If the installer doesn’t already have administrator system permissions, it tries to elevate its current permissions by using this method for bypassing the Microsoft User Account Control, a security protection designed to prevent unauthorized changes to the OS unless they’re approved by an account with administrative rights.

The installer disables HVCI by setting the enabled registry value under the HypervisorEnforcedCodeIntegrity registry key to zero, as described here. The HVCI ensures that all kernel-mode drivers and binaries are signed before they can run. The installer disables it so that the custom unsigned kernel mentioned earlier can be installed later in the execution chain.

The installer must also disable BitLocker because it can be used in combination with a Trusted Platform Module to ensure that Secure Boot hasn't been tampered with. To do this, the installer calls the DisableKeyProtectors method, with the DisableCount parameter set to zero.

The MOK process.

Enlarge / The MOK process.

2. Once the device restarts, BlackLotus gains persistence, meaning it will run each time the device starts. The malware does this by exploiting CVE-2022-21894 and, when Secure Boot is enabled, registering an attacker-designated machine owner key (MOK). A MOK allows owners of devices running non-Windows OSes to generate keys that sign non-Microsoft components during the boot process. The MOK is used in combination with what's known as a shim loader, which is signed by various Linux distributors. This MOK process is illustrated in the image to the right.

Steps 2 through 4 of the figure above show this fits into the overall BlackLotus execution chain. The image below shows the self-signed certificate corresponding to the MOK.

A self-signed certificate for the BlackLotus malware. Note the Issuer NM "When they cry CA," a

Enlarge / A self-signed certificate for the BlackLotus malware. Note the Issuer NM "When the CA," a reference to the *Higurashi When They Cry* anime series.
ESET

ESET’s Smolár explained:

> In a nutshell, this process consists of two key steps:
>
> 1. Exploiting CVE-2022-21894 to bypass the Secure Boot feature and install the bootkit. This allows arbitrary code execution in early boot phases, where the platform is still owned by firmware and UEFI Boot Services functions are still available. This allows attackers to do many things they should not be able to do on a machine with UEFI Secure Boot enabled without having physical access to it, such as modifying Boot-services-only NVRAM variables. And this is what attackers take advantage of to set up persistence for the bootkit in the next step.
> 2. Setting persistence by writing its own MOK to the `MokList`, [in the] boot-services-only NVRAM variable. By doing this, it can use a legitimate Microsoft-signed shim for loading its self-signed (signed by the private key belonging to the key written to `MokList`) UEFI bootkit instead of exploiting the vulnerability on every boot.

The ESET post provides more granular descriptions of the exploitation of CVE-2022-21894 and gaining persistence here and here.

3. From then on, each time the device boots, the attacker's self-signed bootkit is executed. As explained earlier, the bootkit ensures that both the kernel driver preventing file deletion and the HTTP downloader are installed (steps 5 through 9). From the post:

> "The kernel driver is responsible for:
>
> - Deploying the next component of the chain—an HTTP downloader
> - Keeping the loader alive in case of termination
> - Protecting bootkit files from being removed from ESP
> - Executing additional kernel payloads, if so instructed by the HTTP downloader
> - Uninstalling the bootkit, if so instructed by the HTTP downloader
>
> The HTTP downloader is responsible for:
>
> - Communicating with its C&C
> - Executing commands received from the C&C
> - Downloading and executing payloads received from the C&C (supports both kernel payloads and user-mode payloads)

Here is a diagram showing the execution of the UEFI bootkit:

Diagram showing the execution of the BlackLotus bootkit.

Enlarge / Diagram showing the execution of the BlackLotus bootkit.
ESET

It's not known who is behind BlackLotus. One clue, however, may be in the restrictions found in some of the samples that prevent execution if a device is located in:

- Moldova (Romanian), ro-MD
- Moldova (Russian), ru-MD
- Russia (Russian), ru-RU
- Ukraine (Ukrainian), uk-UA
- Belarus (Belarusian), be-BY
- Armenia (Armenian), hy-AM
- Kazakhstan (Kazakh), kk-KZ

Often, attackers in one of these countries take pains not to infect devices there to prevent being arrested and prosecuted since these places have treaties allowing for extradition, though they generally don't have extradition treaties with the US and other Western countries.

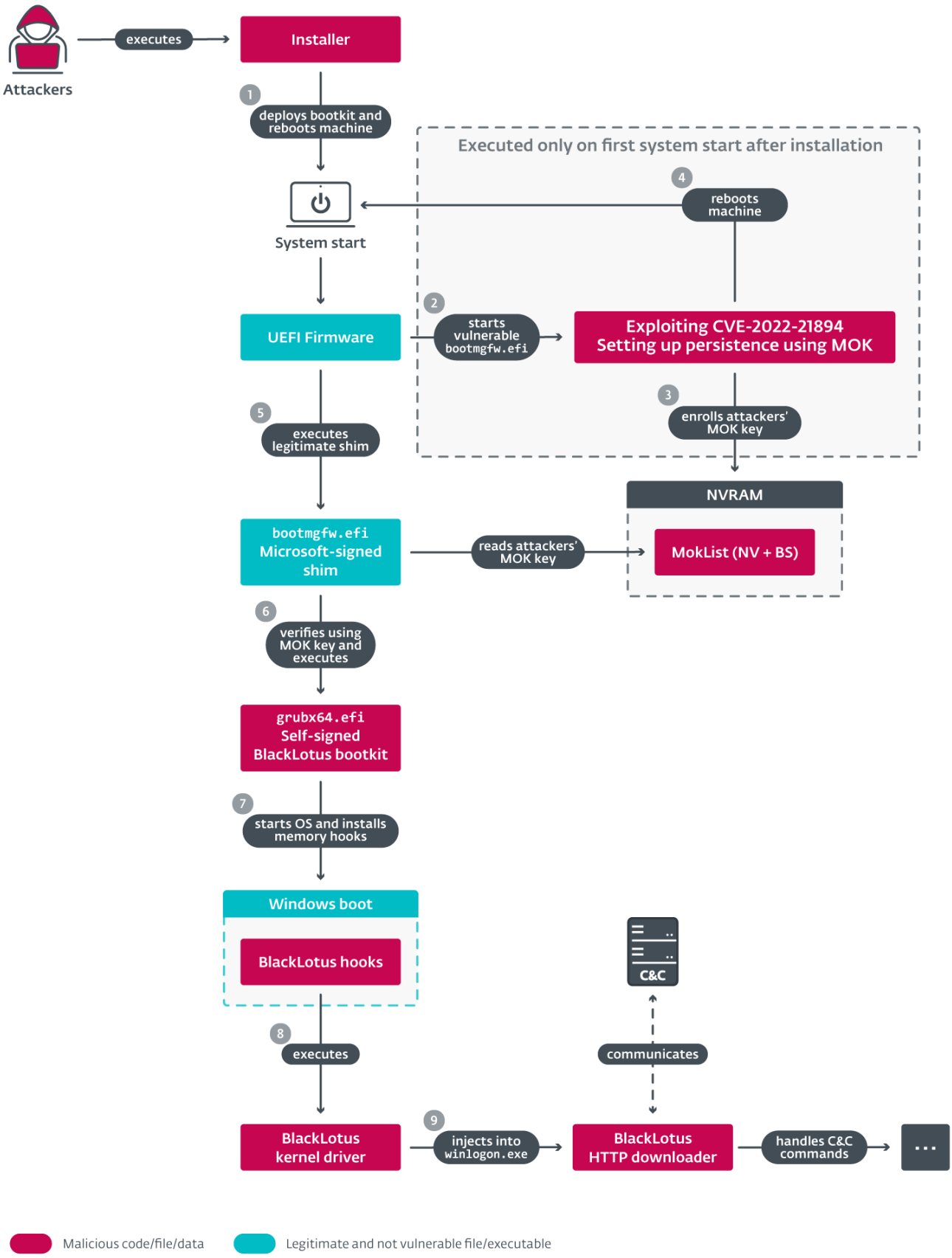
It's also not clear how many devices have been infected by BlackLotus or how it gets installed. As mentioned earlier, the installer must gain administrator permissions to run. That's a high bar that means a computer is already fully compromised. In a statement, Microsoft officials wrote, "This technique [for exploiting CVE-2022-21894] requires administrative access for remote attacks or physical access for local attacks. We are investigating further and will do what is necessary to help keep our customers safe and protected."

For now, the only way to prevent infections by BlackLotus is to ensure that all available OS and app patches have been installed. This won't prevent the bootkit from running, but it will make it harder for the installer to gain the administrative privileges it needs. Antivirus products that monitor firmware for malicious tampering might also provide some level of protection.

Despite the high bar, BlackLotus could prove useful as an alternative to more traditional forms of backdoor malware, which also require administrator permissions. BlackLotus is harder to detect than many pieces of traditional malware. Fortunately, unlike many UEFI bootkits, it can be removed by reinstalling the OS, Boutin.

The handful of previously discovered bootkits in the wild—including CosmicStrand, MosaicRegressor, FinSpy, and MoonBounce (all four discovered by security firm Kaspersky) and ESPecter (like BlackLotus discovered by ESET)—provide the same benefits, but they were easily defeated by enabling Secure Boot. BlackLotus represents a major milestone in the continuing evolution of UEFI bootkits and signals the world's continuing susceptibility to them.

Schéma


Attackers
executes
Installer
1
deploys bootkit and reboots machine
System start
Executed only on first system start after installation
4
reboots machine
2
starts vulnerable bootmgfw.efi
Exploiting CVE-2022-21894
Setting up persistence using MOK
3
enrolls attackers' MOK key
UEFI Firmware
5
executes legitimate shim
NVRAM
bootmgfw.efi
Microsoft-signed shim
reads attackers' MOK key
MokList (NV + BS)
6
verifies using MOK key and executes
grubx64.efi
Self-signed
BlackLotus bootkit
7
starts OS and installs memory hooks
Windows boot
BlackLotus hooks
C&C
8
executes
communicates
BlackLotus
kernel driver
9
injects into winlogon.exe
BlackLotus
HTTP downloader
handles C&C commands
...
Malicious code/file/data
Legitimate and not vulnerable file/executable

Sources

https://arstechnica.com/information-technology/2023/03/unkillable-uefi-malware-bypassing-secure-boot-enabled-by-unpatchable-windows-flaw/

Revision #1
Created 2026-07-17 15:09:51 UTC by Olivier
Updated 2026-07-17 15:09:52 UTC by Olivier