A Practical Analysis of UEFI threats against Windows 11

Joshua Machauer

June 21, 2016
Version: My First Draft

Clean Thesis Style University

CleanThesis

Department of Clean Thesis Style
Institute for Clean Thesis Dev
Clean Thesis Group (CTG)

Documentation

A Practical Analysis of UEFI threats against Windows 11

Joshua Machauer

1. Reviewer	Jane Doe Department of Clean Thesis Style Clean Thesis Style University
2. Reviewer	John Doe Department of Clean Thesis Style Clean Thesis Style University
Supervisors	Jane Doe and John Smith

June 21, 2016

Joshua Machauer

A Practical Analysis of UEFI threats against Windows 11

Documentation, June 21, 2016

Reviewers: Jane Doe and John Doe

Supervisors: Jane Doe and John Smith

Clean Thesis Style University

Clean Thesis Group (CTG)

Institute for Clean Thesis Dev

Department of Clean Thesis Style

Street address

Postal Code and City

Abstract

In Computer Security one of the most feared threats is a bootkit, executing at the beginning of a computers boot chain, before the operating system and accompanying antivirus programs. With the wide spread adaption of standardized UEFI firmware these threats have become less machine dependent and can now target a host of systems at once. Past analyses about bootkits have been case studies of their appearences in the wild, this thesis instead aims to be a more practical approach by developing a bootkit and analysing the challenges doing so. We restrict our analysis by assuming an attacker has already gained read and write access to the BIOS image and is thus only facing security mechanisms involved during and with execution of the bootkit. Our bootkit was able to achieve elevated execution on Windows 11 by exploiting unrestricted hard drive access to edit Windows Registries, this was also possible on Bitlocker encrypted hard drives by keylogging the Recovery Key. UEFI makes it very easy for an attacker who has gained access to the System Firmware to leverage it's powers and gain full control over the system.

Abstract (different language)

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Acknowledgement

Contents

1 Introduction **1**

2 Related Work **3**

3 Background **5**

3.1 UEFI 5

3.1.1 Boot Sequence 5

3.1.2 UEFI/PI Firmware Images 6

3.1.3 UEFI Images 6

3.1.4 Firmware Core 7

3.1.5 edk2 7

3.1.6 Security 8

3.2 Windows 8

3.2.1 User Access Control (UAC) 8

3.2.2 Signing 8

3.2.3 Bitlocker 8

4 Attacks **9**

4.1 Neither Secure Boot nor Bitlocker 9

4.2 Secure Boot 10

4.3 Secure Boot and Bitlocker 10

5 Discussion **13**

5.1 Rootkit classification 13

5.2 Mitigations 13

5.2.1 User awareness 13

6 Conclusion **15**

A Example Appendix **23**

A.1 Appendix Section 1 23

A.2 Appendix Section 2 23

Declaration **25**

1 Introduction

definition of rootkit/bootkit

persistence

2 Related Work

3 Background

3.1 UEFI

general introduction to uefi interface between operating system and platform firmware data datables containing platform-related information boot- and runttime service functions for the bootloader and os to call pure interface specification merely states what interfaces to offer and not how they are implemented nor which or how they are used goals: - complete solution describing all features and capabilities - abstract interfaces to support a range of processors without the need for knowledge about underlying hardware for the bootloader - sharable persistent storage for platform support code replace bios but also backwards compatible with Compability Support Module (CMS) supports boot from media containing UEFI OS loader or UEFI System Partition does not require changes to the first sector, this allows media to boot legacy and uefi at the same time security

3.1.1 Boot Sequence

focus will be on dxe and transient system load

1. Security (SEC)

establishment of root of trust

2. Pre-EFI (PEI)

3. Driver Execution Environment (DXE)

dxe core dxe dispatcher depex dxe drivers

4. Boot Device Selection (BDS)

5. Transient System Load (TSL)

boottime and runtime services/driver bootloader ExitBootServices()

6. Runtime (RT)

runtime services/driver

7. Afterlife (AL)

hibernation sleep

3.1.2 UEFI/PI Firmware Images

flash device flash volume flash file system file sections depex

3.1.3 UEFI Images

executable subset of PE32+ file format with modified header signature to distinguish from normal PE32 Images + stands addition of 64-bit relocation fix-up extension fixed and dynamic address loading relocatable boot and runtime memory application vs os loader vs driver loaded fully into memory and reloaction fix ups memory marked as code and data jump to entry point what is the boot manager

UEFI Applications

example efi shell loaded by boot manager or other applications return or calling exit specifically always unloaded from memory

UEFI OS Loaders

example windows boot manager normally take over control from the firmware upon load behaves like a normal UEFI application - only use memory allocated from the firmware - only use services/protocols to access devices that the firmware exposes - conform to driver specifications to access hardware on error can return allocated resources with Exit boot service with error specific information given in ExitData on success take full control with ExitBootServices boot service all boot services in the system are terminated, including memory management UEFI OS loader now responsible

UEFI Drivers

loaded by boot manager, UEFI firmware (DXE foundation), or other applications example payload unloaded only when returning error code presistent on success boot and runtime drivers only difference is that runtime are available after Exit-BootServices was called boottime drivers are terminated and memory is released runttime drivers are fixed up with virtual mappings upon SetVirtualAddressMap call has to convert its allocated memory

3.1.4 Firmware Core

boot and runtime services boot service table guids handles and protocols protocols

3.1.5 edk2

build system

3.1.6 Security

Secure Boot

Signed Capsule Update

3.2 Windows

3.2.1 User Access Control (UAC)

3.2.2 Signing

3.2.3 Bitlocker

how does it work explain TPM

Attacks

4

attacks with escalating security mechanisms assumptions: read/write access to firmware image through exploit or physical access (spi clamp override)

how does one compile uefi application with edk2 it's open source so we can look up examples for most stuff

4.1 Neither Secure Boot nor Bitlocker

use read access to dump image open with UEFITool remove previous NTFS driver if present, for full control, might be read only etc in UEFITool search and remove add in NTFS driver use write access

try in EFI shell navigate to Windows folder create folder

try in code compile dxe driver within ovmf to receive .ffs file with version depex user interface section SimpleFileSystem Protocol iteration write failed on hibernated file patch to allow write on hibernated drivers pack executable binary as uefi module iterate over firmware volume protocols search for payload guid check size match override notepad works

but no automatic execution nor elevated privileges dll proxying dll hijacking

Task Scheduler defined in xml cached in registry edit with start cmd.exe and trigger manually whoami

chntpw and reged port to uefi edit Task in machine under Control maybe look if just adding a key would have also worked export target registry key modify so that registry key can differ and found via matching values import and override registry key on target machine payload whoami

4.2 Secure Boot

how does one enable it expect not to boot no difference secure boot default policy snippet option roms and bootloader instead relies on Signed Capsule Updates

4.3 Secure Boot and Bitlocker

assumptions: secure boot or not bitlocker enabled with TPM auto decryption

observation: boot execution differs from executing reootkit tpm values different bitlocker auto decryption fails recovery key prompt

what is the reaction of the average user (ask admin for recovery password) type in recovery password alternative would be to remove drive and insert into safe device

prompt is done by the OS Loader ergo still during transient system load phase required to use protocol services therefor uses uefi services for IO such as SimpleTextInputEx Protocol go over the two different input protocols find out which one is used

explain more in depth how protocols are returned to the end user one instance per controller/handle

explain basic hooking explain how we retain information of the hook in question map protocol pointer to hook information keylog recovery key key input advancment is weird and makes tracking tricky

alternatively screen shot still need hook to find when enter is pressed explain how screenshotting works some basic compression

on real hardware network stack wasn't installed onto handles when boot over ip was disabled compared loaded dxe drivers between both configurations with efi shell Realtek Family driver not loaded load manually reinstall all handle to controllers to enable network stack regardless

sending key out is only good for physical access attack vector dislocker linux utility mount encrypted drive with decryption mean read and write access dual boot in vm enter password and it works port to uefi bitlocker encrypts block-wise uefi protocol stack hook block io again hook data mapping wait for recovery key send recovery key on enter press hook ExitBootServices enable hook write payload import registry key disable hook

next boot would require to input tpm values again update tpm values in payload caveat pin? look into this

persistence when part of root of trust fresh install / tpm update values hook Trusted Copmuting Group 2 (TCG2) Protocol TPM communication receive bitlocker vmk key and send to dislocker

5 Discussion

attack assumption reflected to real world aplicability

social engineering aspekt

driver vorhanden und was mitbringen, debloating

5.1 Rootkit classification

statisken zu bilocker und secureboot auf systemen

industrie standard zur system security in firmen

5.2 Mitigations

hardware validated boot

inaccessible spi flash

tpm + pin detectability

googeln wie legitime recovery key prompt reaktion aussieht

enterprise policy on reovery key loss

5.2.1 User awareness

vermitteln was das prompt bedeuten koennte

aber kann man einfach nicht anzeigen lassen

Security Flaw of entering a Recovery Password in an inheritly unsafe System

enterprise doesnt hand out recovery keys and instead receives hard drive

!!!!!!!!!!!!!!!!!!!!!!!!!! without hardware chain of trust a compromised system can patch/change any software and fixes are impossible

phishing prompts on their own

6 Conclusion

when we are already in the image we can gain full control over the system system cant be trusted anymore e.g. uefi services full file access escalate it to local system level execution bitlocker has the flaw of allowing to enter criticial information into an inherently untrustable system on the other hand one could force such a prompt themselves mere exisitence of a recovery key is a security flaw

List of Figures

List of Tables

A.1 This is a caption text. 23
A.2 This is a caption text. 24

List of Listings

Example Appendix A

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

A.1 Appendix Section 1

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Alpha	Beta	Gamma
0	1	2
3	4	5

Tab. A.1.: This is a caption text.

A.2 Appendix Section 2

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Alpha	Beta	Gamma
0	1	2
3	4	5

Tab. A.2.: This is a caption text.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Declaration

You can put your declaration here, to declare that you have completed your work solely and only with the help of the references you mentioned.

City, June 21, 2016

Joshua Machauer