Dynamic Detection of Vulnerability Exploitation in Windows

Dynamisk detektion af udnyttelse af sårbarheder i Windows

Author:
Søren Fritzbøger
s153753@student.dtu.dk

Supervisor:
Christian D. Jensen
cdje@dtu.dk

A thesis presented for the degree of
Master of Science in Computer Science and Engineering

DTU Compute
Danmarks Tekniske Universitet

May 4, 2021

Todo list

Abstract

Write something very clever here and read it through 10000 times

Table of contents

Introduction

Introduce something here

1.1 Purpose

Purpose

1.2 Thesis overview

Thesis overview

1.3 Related work

Purpose

Tracing and logging

2.1 Windows telemetry

2.2 Event Tracing for Windows (ETW)

Add proper description of what ETW is.

In the architecture of Event Tracing for Windows (ETW) events are at the centerpiece where they are created, managed and consumed by different event components[1]. These differentiate between event *providers*, event *consumers*, and event *controllers*. All of these event components handle the workflow of ETW, either by reading or writing, or by controlling the events in some way.

2.2.1 Event components

figure out if the components should just be subsections

Controllers

Providers

Providers are the system- and userland applications that provide events and data. They do so by registering themselves as a provider, allowing a controller to enable or disable events. By having the controller control whether events are enabled or not, allows an application to have tracing without generating alerts all the time. This is especially interesting for debugging purposes, which is usually not needed during regular usage.

Many different

Consumers

Vulnerability analysis

3.1 CVE-2020-24086

According to Microsoft[2] CVE-2021-24086 is a denial of service vulnerability with a CVSS:3.0 score of 7.5 / 6.5, that is a base score metrics of 7.5 and a temporal score metrics of 6.5. The vulnerability affects all supported versions of Windows and Windows Server. According to an accompanied blog post published by Microsoft [4] at the same time as the patch was released, details that the vulnerable component is the Windows TCP/IP implementation, and that the vulnerability revolves around IPv6 fragmentation. The Security Update guide and the blog post also present a workaround that can be used to temporarily mitigate the vulnerability by disabling IPv6 fragmentation.

Figure out if this should be here

3.1.1 Public information

Due to the Microsoft Active Protetions Program (MAPP)[3] security software providers are given early access to vulnerability information. This information often include Proof of Concept (PoC)s for vulnerabilities to be patched, in order to aid security software providers to create valid detections for exploitation of soon-to-be patched vulnerabilities. Due to MAPP, some security software providers publish relevant information regarding recently patched vulnerabilities. However, the information is usually very vague in details, and can therefore only aid in the initial exploration of the vulnerability. For CVE-2021-24086, both McAfee[6] and Palo Alto[5]

3.1.2 Patch diffing

3.1.3 Root-cause analysis

3.1.4 Triggering the vulnerability

Detection

4.1 Event Tracing for Windows (ETW)

4.2 Hooking and DTrace

4.3 Implementation

Scaling and extensibility

Conclusion

Conclude something please

Abbreviations

ETW Event Tracing for Windows. 3, 5

MAPP Microsoft Active Protetions Program. 4

PoC Proof of Concept. 4

List of Figures

List of code snippets

Bibliography

[1] Microsoft. *About Event Tracing - Win32 apps | Microsoft Docs*. URL: https://docs.microsoft.com/en-us/windows/win32/etw/about-event-tracing (visited on 05/31/2018).

[2] Microsoft. *CVE-2021-24086 - Security Update Guide - Microsoft - Windows TCP/IP Denial of Service Vulnerability*. URL: https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-24086 (visited on 02/09/2021).

[3] Microsoft. *Microsoft Active Protections Program*. URL: https://www.microsoft.com/en-us/msrc/mapp (visited on 02/09/2021).

[4] Microsoft. *Multiple Security Updates Affecting TCP/IP: CVE-2021-24074, CVE-2021-24094, and CVE-2021-24086 - Microsoft Security Response Center*. URL: https://msrc-blog.microsoft.com/2021/02/09/multiple-security-updates-affecting-tcp-ip/ (visited on 02/09/2021).

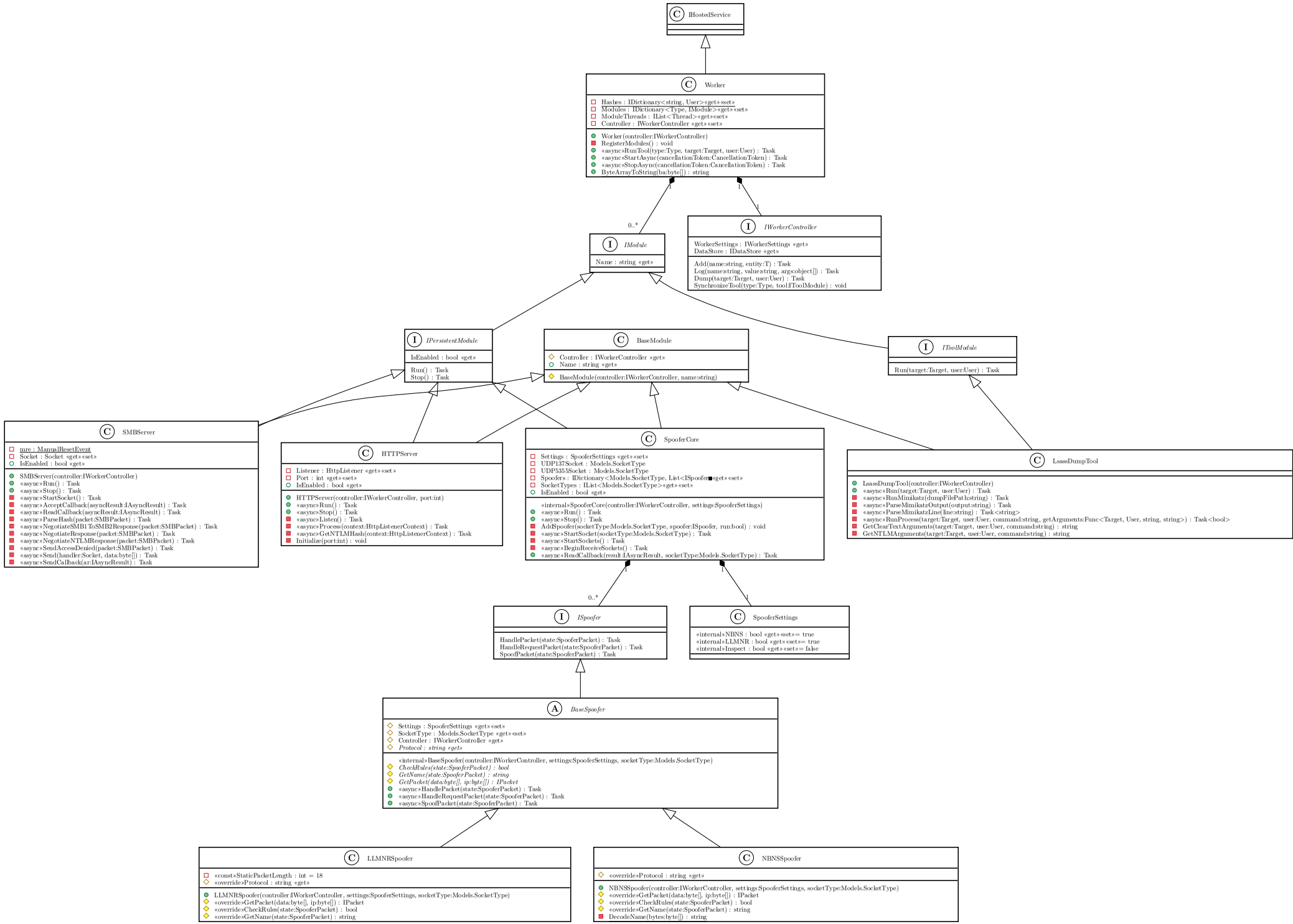
[5] Abisheik Ganesan from Palo Alto. *Threat Brief: Windows IPv4 and IPv6 Stack Vulnerabilities (CVE-2021-24074, CVE-2021-24086 and CVE-2021-24094)*. URL: https://unit42.paloaltonetworks.com/cve-2021-24074-patch-tuesday/ (visited on 02/09/2021).

[6] Steve Povolny et al. *Researchers Follow the Breadcrumbs: The Latest Vulnerabilities in Windows' Network Stack | McAfee Blogs*. URL: https://www.mcafee.com/blogs/other-blogs/mcafee-labs/researchers-follow-the-breadcrumbs-the-latest-vulnerabilities-in-windows-network-stack/ (visited on 02/09/2021).

Appendices

.1 Class Diagrams

.1.1 Worker class diagram


IHostedService
Worker
- Hashes : IDictionary<string, User>«get»«set»
- Modules : IDictionary<Type, IModule>«get»«set»
- ModuleThreads : IList<Thread>«get»«set»
- Controller : IWorkerController «get»«set»
+ Worker(controller:IWorkerController)
- RegisterModules() : void
+ «async»RunTool(type:Type, target:Target, user:User) : Task
+ «async»StartAsync(cancellationToken:CancellationToken) : Task
+ «async»StopAsync(cancellationToken:CancellationToken) : Task
+ ByteArrayToString(ba:byte[]) : string
1
1
0..*
IModule
Name : string «get»
IWorkerController
WorkerSettings : IWorkerSettings «get»
DataStore : IDataStore «get»
Add(name:string, entity:T) : Task
Log(name:string, value:string, args:object[]) : Task
Dump(target:Target, user:User) : Task
SynchronizeTool(type:Type, tool:IToolModule) : void
IPersistentModule
IsEnabled : bool «get»
Run() : Task
Stop() : Task
BaseModule
- Controller : IWorkerController «get»
+ Name : string «get»
+ BaseModule(controller:IWorkerController, name:string)
IToolModule
Run(target:Target, user:User) : Task
SMBServer
- mre : ManualResetEvent
- Socket : Socket «get»«set»
+ IsEnabled : bool «get»
+ SMBServer(controller:IWorkerController)
+ «async»Run() : Task
+ «async»Stop() : Task
- «async»StartSocket() : Task
- «async»AcceptCallback(asyncResult:IAsyncResult) : Task
- «async»ReadCallback(asyncResult:IAsyncResult) : Task
- «async»ParseHash(packet:SMBPacket) : Task
- «async»NegotiateSMB1ToSMB2Response(packet:SMBPacket) : Task
- «async»NegotiateResponse(packet:SMBPacket) : Task
- «async»NegotiateNTLMResponse(packet:SMBPacket) : Task
- «async»SendAccessDenied(packet:SMBPacket) : Task
- «async»Send(handler:Socket, data:byte[]) : Task
- «async»SendCallback(ar:IAsyncResult) : Task
HTTPServer
- Listener : HttpListener «get»«set»
- Port : int «get»«set»
+ IsEnabled : bool «get»
+ HTTPServer(controller:IWorkerController, port:int)
+ «async»Run() : Task
+ «async»Stop() : Task
- «async»Listen() : Task
- «async»Process(context:HttpListenerContext) : Task
- «async»GetNTLMHash(context:HttpListenerContext) : Task
- Initialize(port:int) : void
SpooferCore
- Settings : SpooferSettings «get»«set»
- UDP137Socket : Models.SocketType
- UDP5355Socket : Models.SocketType
- Spoofers : IDictionary<Models.SocketType, List<ISpoofer■«get»«set»
- SocketTypes : IList<Models.SocketType>«get»«set»
+ IsEnabled : bool «get»
«internal»SpooferCore(controller:IWorkerController, settings:SpooferSettings)
+ «async»Run() : Task
+ «async»Stop() : Task
- AddSpoofer(socketType:Models.SocketType, spoofer:ISpoofer, run:bool) : void
- «async»StartSocket(socketType:Models.SocketType) : Task
- «async»StartSockets() : Task
- «async»BeginReceiveSockets() : Task
+ «async»ReadCallback(result:IAsyncResult, socketType:Models.SocketType) : Task
LsassDumpTool
+ LsassDumpTool(controller:IWorkerController)
+ «async»Run(target:Target, user:User) : Task
- «async»RunMimikatz(dumpFilePath:string) : Task
- «async»ParseMimikatzOutput(output:string) : Task
- «async»ParseMimikatzLine(line:string) : Task<string>
- «async»RunProcess(target:Target, user:User, command:string, getArguments:Func<Target, User, string, string>) : Task<bool>
- GetClearTextArguments(target:Target, user:User, command:string) : string
- GetNTLMArguments(target:Target, user:User, command:string) : string
1
1
0..*
ISpoofer
HandlePacket(state:SpooferPacket) : Task
HandleRequestPacket(state:SpooferPacket) : Task
SpoofPacket(state:SpooferPacket) : Task
SpooferSettings
«internal»NBNS : bool «get»«set»= true
«internal»LLMNR : bool «get»«set»= true
«internal»Inspect : bool «get»«set»= false
BaseSpoofer
- Settings : SpooferSettings «get»«set»
- SocketType : Models.SocketType «get»«set»
- Controller : IWorkerController «get»
- Protocol : string «get»
«internal»BaseSpoofer(controller:IWorkerController, settings:SpooferSettings, socketType:Models.SocketType)
CheckRules(state:SpooferPacket) : bool
GetName(state:SpooferPacket) : string
GetPacket(data:byte[], ip:byte[]) : IPacket
+ «async»HandlePacket(state:SpooferPacket) : Task
+ «async»HandleRequestPacket(state:SpooferPacket) : Task
+ «async»SpoofPacket(state:SpooferPacket) : Task
LLMNRSpoofer
- «const»StaticPacketLength : int = 18
- «override»Protocol : string «get»
+ LLMNRSpoofer(controller:IWorkerController, settings:SpooferSettings, socketType:Models.SocketType)
- «override»GetPacket(data:byte[], ip:byte[]) : IPacket
- «override»CheckRules(state:SpooferPacket) : bool
- «override»GetName(state:SpooferPacket) : string
NBNSSpoofer
- «override»Protocol : string «get»
+ NBNSSpoofer(controller:IWorkerController, settings:SpooferSettings, socketType:Models.SocketType)
- «override»GetPacket(data:byte[], ip:byte[]) : IPacket
- «override»CheckRules(state:SpooferPacket) : bool
- «override»GetName(state:SpooferPacket) : string
- DecodeName(bytes:byte[]) : string