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

February 23, 2021

Abstract

Write something very clever here and read it through 10000 times

Table of contents

Introduction

Introduce something here

Tracing and logging

2.1 Windows telemetry

2.2 Event Tracing for Windows (ETW)

Vulnerability analysis

3.1 CVE-2020-17140.tex

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

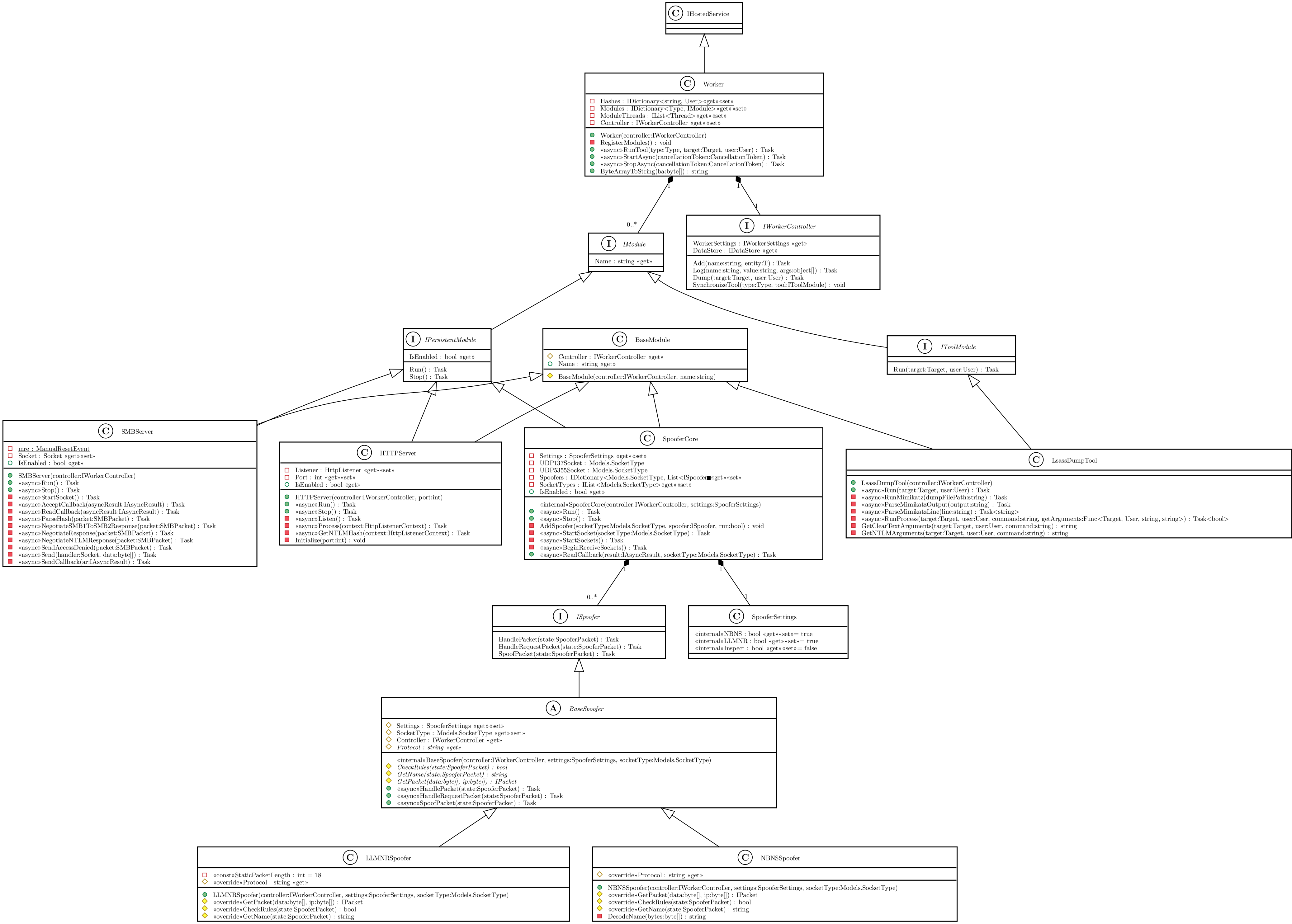
List of Figures

List of code snippets

Appendices

.1 Class Diagrams

.1.1 Worker class diagram


C IHostedService
C 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..*
1
I IModule
Name : string «get»
I 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
I IPersistentModule
IsEnabled : bool «get»
Run() : Task
Stop() : Task
C BaseModule
Controller : IWorkerController «get»
Name : string «get»
BaseModule(controller:IWorkerController, name:string)
I IToolModule
Run(target:Target, user:User) : Task
C 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
C 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
C 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
C 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..*
1
I ISpoofer
HandlePacket(state:SpooferPacket) : Task
HandleRequestPacket(state:SpooferPacket) : Task
SpoofPacket(state:SpooferPacket) : Task
C SpooferSettings
«internal»NBNS : bool «get»«set»= true
«internal»LLMNR : bool «get»«set»= true
«internal»Inspect : bool «get»«set»= false
A 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
C 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
C 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