



Hijacking Arbitrary .NET Application Control Flow

Topher Timzen

#whoami

Security Researcher, Intel

Security Trainer

TopherTimzen.com

[@TTimzen](https://twitter.com/TTimzen)

Overview

✦ .NET?

✦ Runtime Attacks

✦ Modify Control Flow

✦ Machine Code Editing

✦ Managed Heap

Why are we Here?



Tools Released

Use .NET to attack

Using Objects on the Heap

CLR Attacks

Controlling the Common Language Runtime

Accessing raw objects on Managed Heap

Manipulate AppDomains

- **Controlling all Loaded Code**
- **Controlling Just-In-Time Compilation**

Attack With ASM

Manipulate Resources

Attack methods at ASM level

Alter application control flow

Runtime

.NET Process

CLR (2.0/4.0) & AppDomains

Assemblies (.EXE and .DLL(s))

Objects

Properties

Fields

Instance Methods

Classes

Methods

Logic

The Tools

Gray Frost & Gray Storm



Gray Frost



Gray Frost

Payload delivery system

C++ .NET CLR Bootstrapper

Creates or injects 4.0 runtime

Capability to pivot into 2.0 runtime

Contains raw payload

2 Rounds

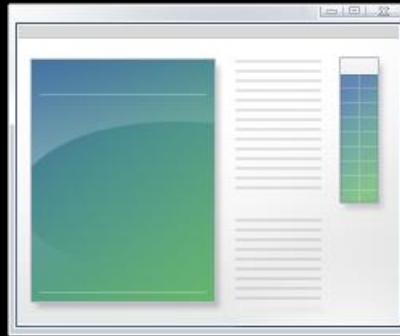
- ✦ **GrayFrostCpp**

- ✦ **GrayFrostCSharp**

 - **C# Payload**

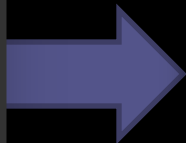
Round 1

.NET Process



Round 1

GrayFrostCpp

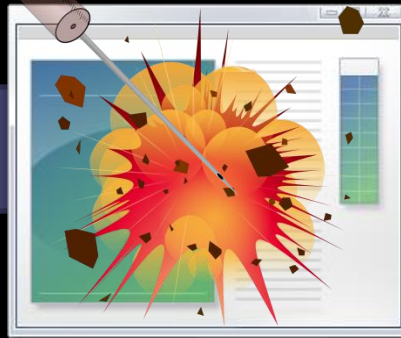


Mscoree



Round 1

GrayFrostCpp



Round 1

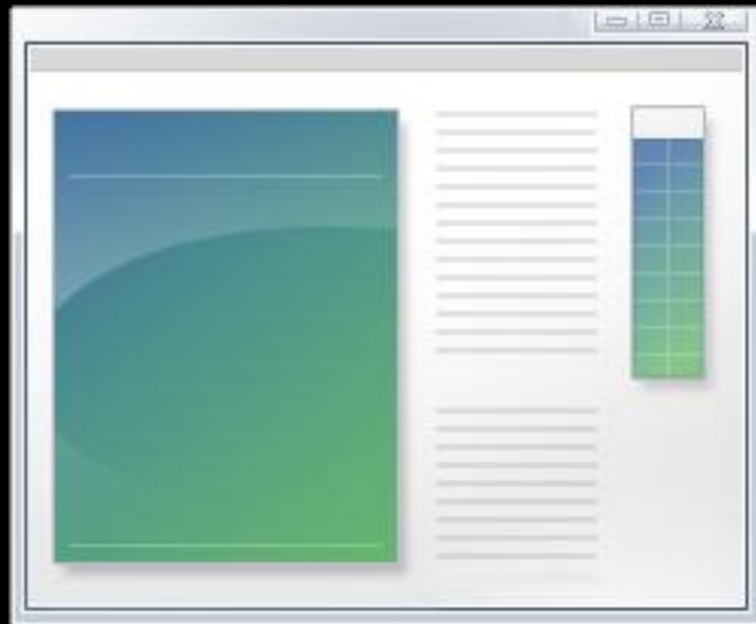
GrayFrostCpp



GrayFrostCSharp

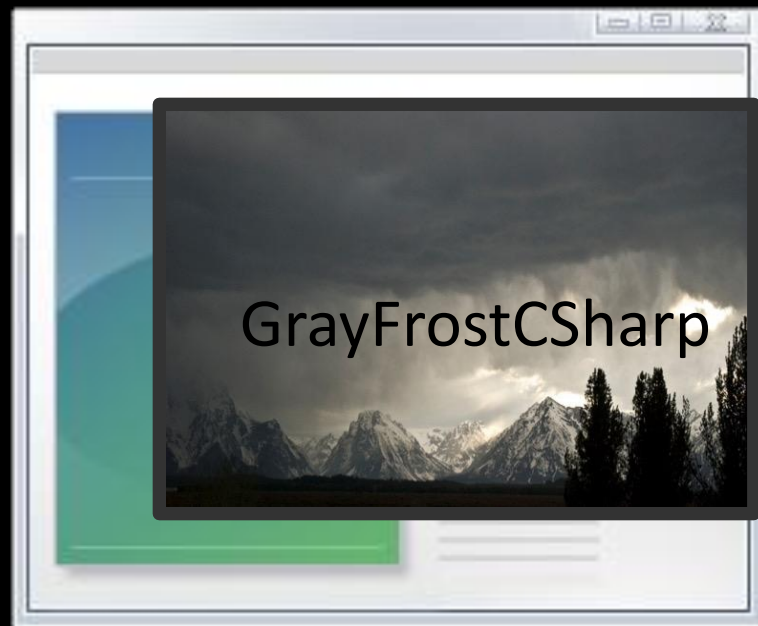
Round 2

.NET Process



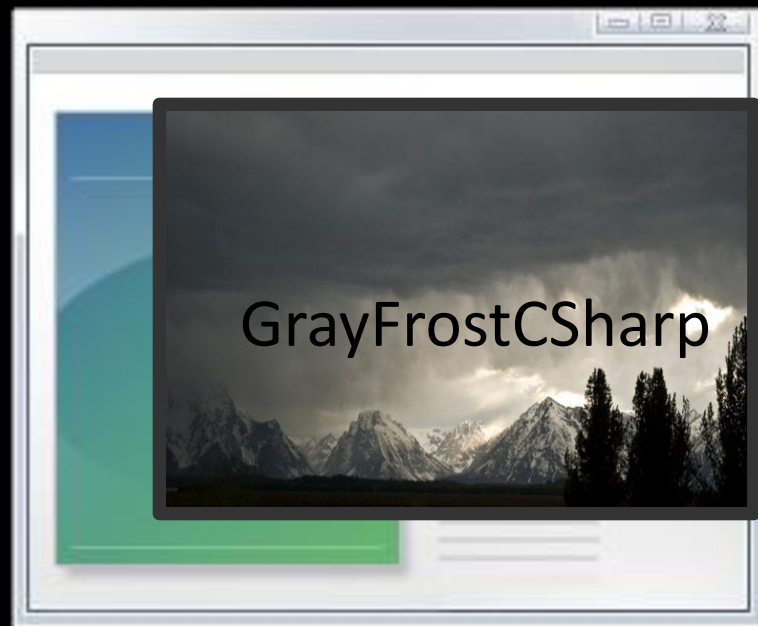
Round 2

.NET Process



Round 2

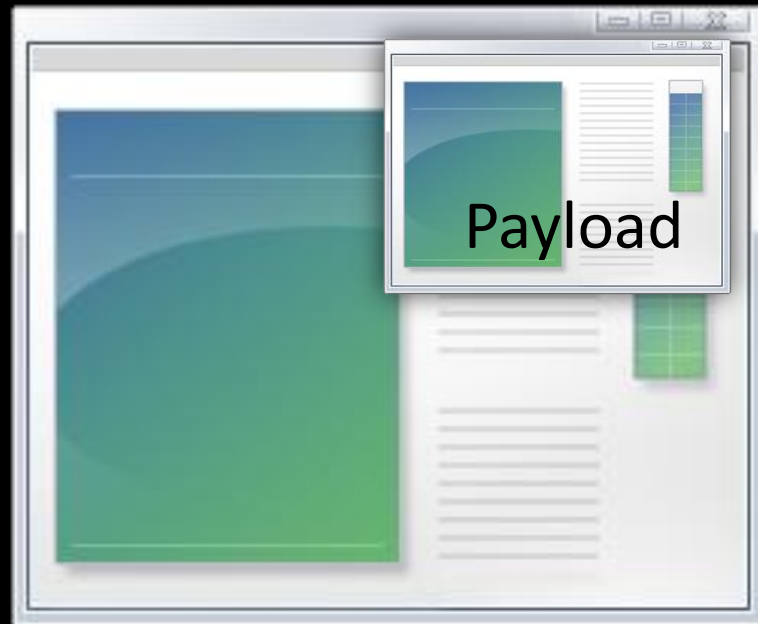
.NET Process



payload void
main()

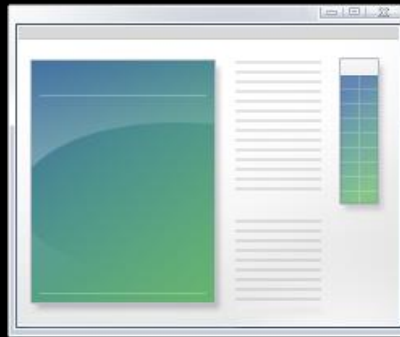
Round 2

.NET Process

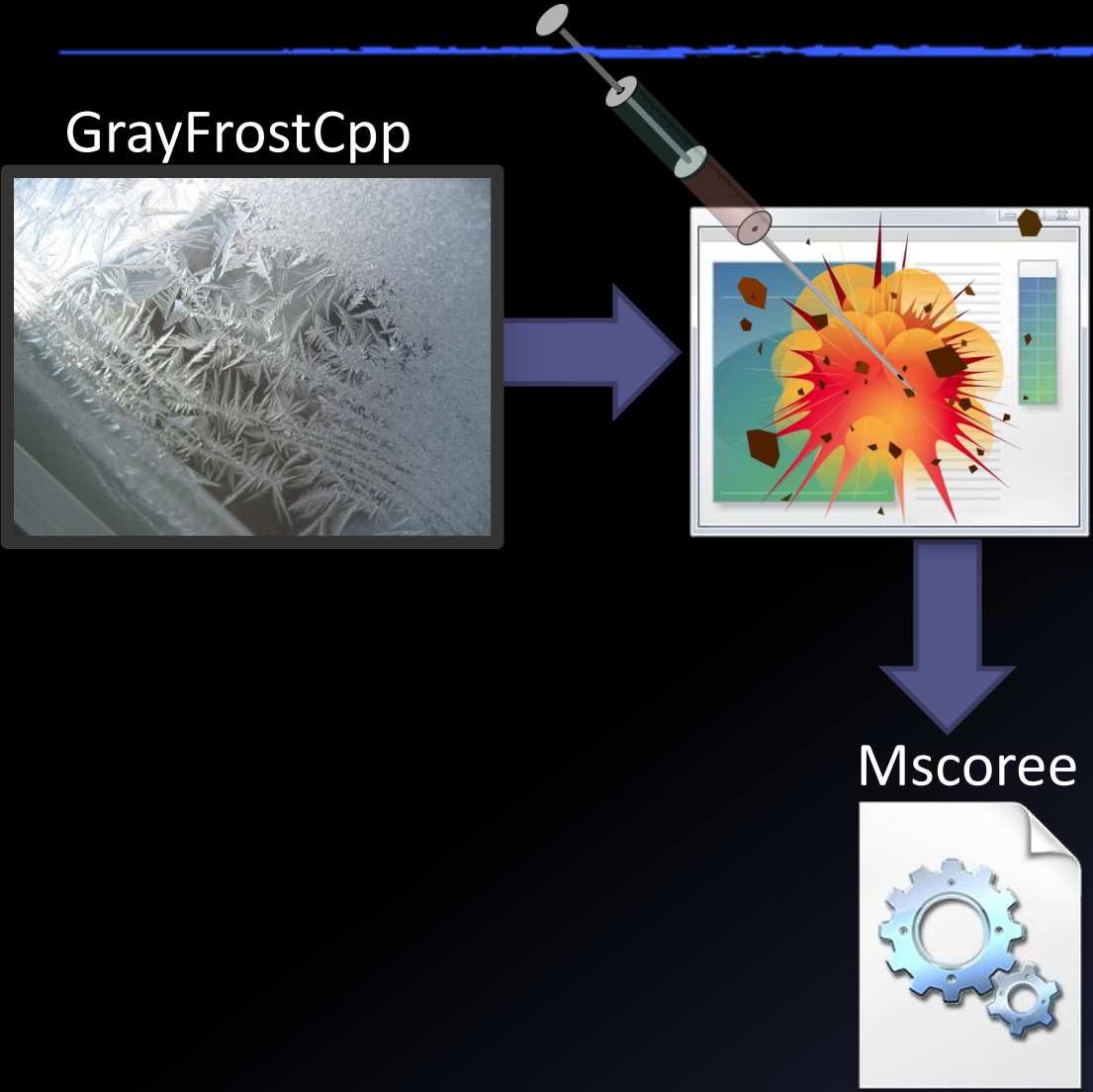


Pivoting Between runtimes

.NET Process



Pivoting Between runtimes



Pivoting Between runtimes

GrayFrostCpp



Pivoting Between runtimes

GrayFrostCpp



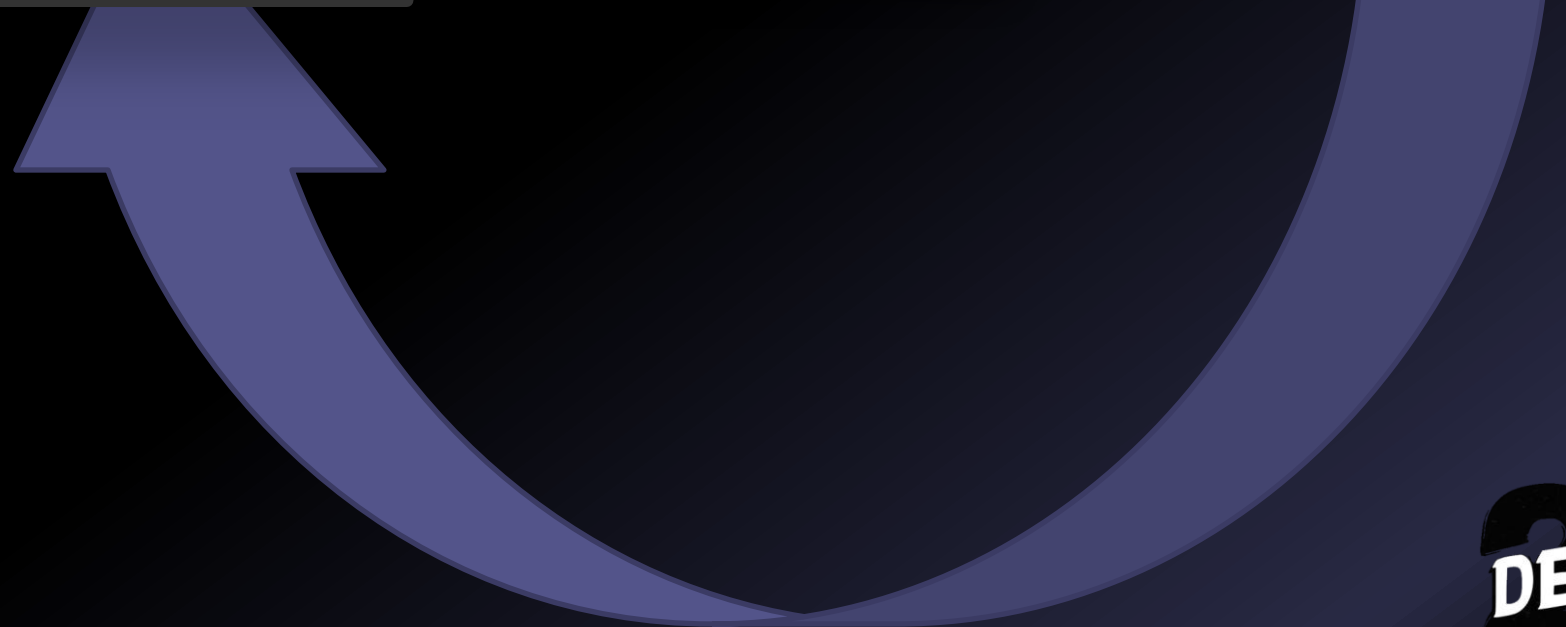
GrayFrostCSharp

Pivoting Between runtimes

GrayFrostCpp



GrayFrostCSharp



Pivoting Between runtimes

GrayFrostCpp



Pivoting Between runtimes

GrayFrostCpp



GrayFrostCSharp

Pivoting Between runtimes

GrayFrostCpp



GrayFrostCSharp

Gray Storm



Gray Storm

**Reconnaissance and In-memory attack
payload**

Features

- ✦ **Attacking the .NET JIT**
- ✦ **Attacking .NET at the ASM level**
- ✦ **ASM and Metasploit payloads**
- ✦ **Utilize objects on the Managed Heap**

Gray Storm Usage

Controlling the JIT

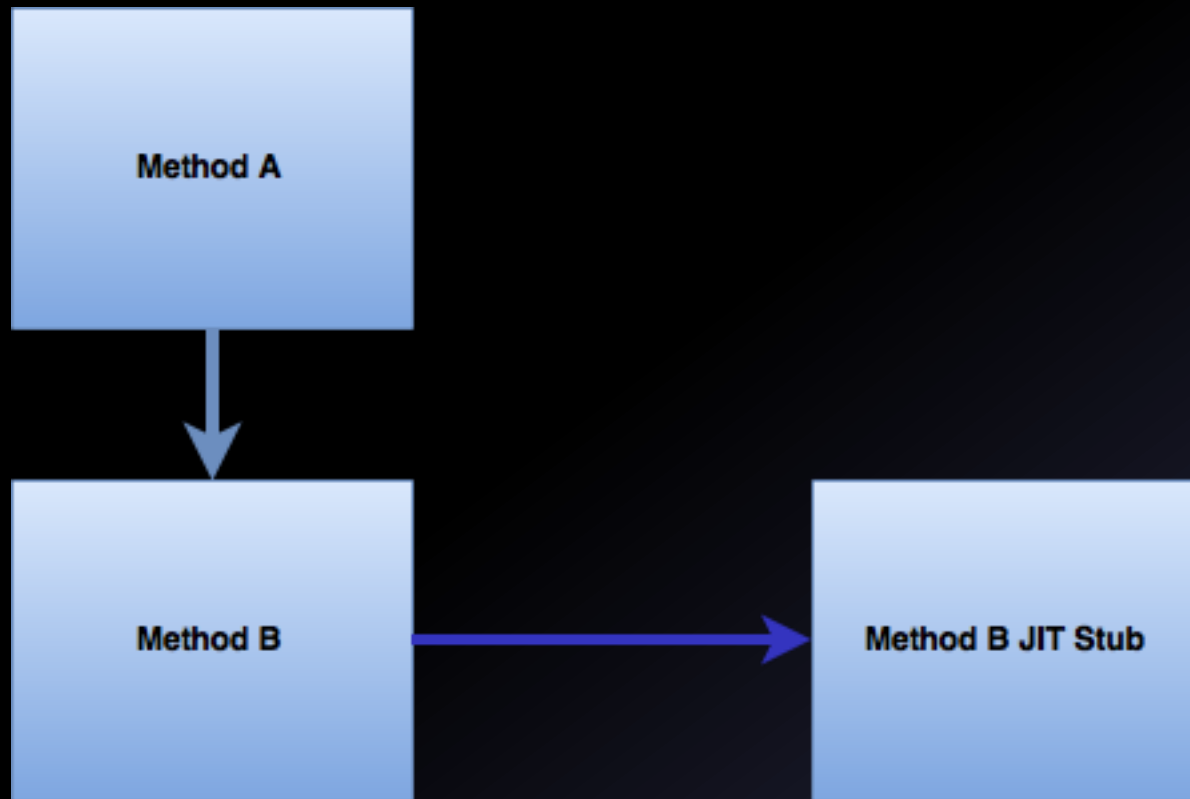
Method Tables contain address of JIT stub for a class's methods.

During JIT the Method Table is referenced

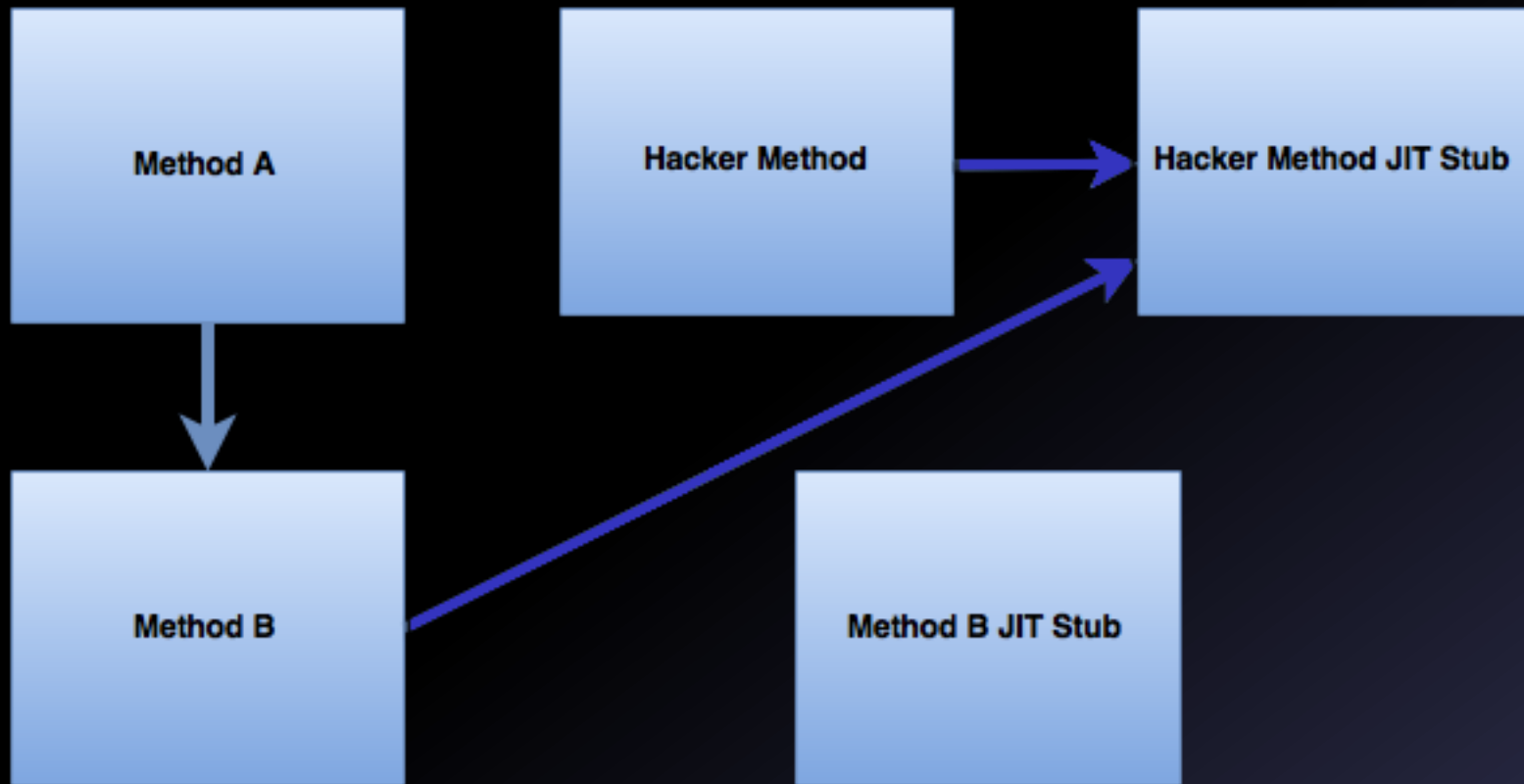
We can control the address

Lives after Garbage Collection

Controlling the JIT



Controlling the JIT



Control Flow Attacks

.NET uses far and relative calls

✦ **0xE8; Call [imm]**

✦ **0xFF 0x15; Call dword
segmentRegister[imm]**

relCall = dstAddress - (currentLocation+ lenOfCall)

ASM Payloads

Address of a method known through Reflection

Overwrite method logic with new ASM

Steal stack parameters

Change events

ASM Payloads

Change return TRUE to return FALSE

- ✦ Password validation
- ✦ Key & Licensing validation
- ✦ SQL Sanitization

Destroy security Mechanisms

Overwrite logic

Update Mechanisms

ASM Payloads

```
public bool VerifyLicense(string licenseKey, string email)
{
    if (string.IsNullOrEmpty(licenseKey) || string.IsNullOrEmpty(email))
    {
        return false;
    }

    this.License = this.Decrypt(licenseKey);

    if (this.License != null && this.License.Email != null &&
        this.License.Product != null)
    {
        bool flag = this.License.Email.Equals(email,
            StringComparison.OrdinalIgnoreCase);
        bool flag2 = this.License.Product == "Licensed";
        return flag && flag2;
    }
    return false;
}
```

```
0x31 xor eax , eax
0x40 inc eax
0xC3 ret
```

```
public bool VerifyLicense(string licenseKey, string email)
{
    return true;
}
```

```
0x55 push ebp
0x8B mov ebp , esp
0x57 push edi
0x56 push esi
0x53 push ebx
0x83 sub esp , 00000014h
0x89 mov dword [ebp-14h] , ecx
0x89 mov dword [ebp-18h] , edx
[snip]
0x8B mov ecx , esi
0xE8 call 53A7CE80h
0x85 test edi , edi
0xF95 setne al
0xFB6 movzx eax , al
0x8D lea esp , dword [ebp-0Ch]
0x5B pop ebx
0x5E pop esi
0x5F pop edi
0x5D pop ebp
0xC3 ret
```

ASM Payloads

Metasploit

Hand Rolled

Portable Environment Block (PEB) changes

Portable Environment Block

```
0:005> !peb
PEB at 7efde000
  InheritedAddressSpace:      No
  ReadImageFileExecOptions:  No
  BeingDebugged:              Yes
  ImageBaseAddress:           012b0000
  Ldr                         77b40200
  Ldr.Initialized:            Yes
  Ldr.InInitializationOrderModuleList: 00273a80 . 00301d48
  Ldr.InLoadOrderModuleList:   002739e0 . 00301d38
  Ldr.InMemoryOrderModuleList: 002739e8 . 00301d40
      Base TimeStamp           Module
  12b0000 54f4a118 Mar 02 09:42:48 2015 C:\\Users\\Blob\\DllInjector.exe
  77a40000 521ea8e7 Aug 28 18:50:31 2013 C:\\Windows\\SysWOW64\\ntdll.dll
  73a10000 4b90752b Mar 04 19:06:19 2010 C:\\Windows\\SYSTEM32\\MSCOREE.DLL
  75fc0000 53159a85 Mar 04 01:19:01 2014 C:\\Windows\\syswow64\\KERNEL32.dll
```

Object Hunting in Memory

Managed Heap

Storage point for .NET Objects

New reference objects added to heap

**Garbage Collector removes dead
objects**

Managed Heap

Storage point for .NET Objects

New reference objects added to heap

Garbage Collector removes dead objects

Let's manipulate it!

Object Hunting in Memory

Objects are IntPtrs

Point to Object Instance on Managed Heap

All instantiated objects of the same class share the same Method Table

Reflection



Object Hunting



Win



Finding Objects at Runtime

- i. Construct an object and find location of Managed Heap
- ii. Signature instantiated type
- iii. Scan Managed Heap for object pointers
- iv. Convert object pointers to raw objects
- v. ????
- vi. PROFIT

Finding Objects at Runtime

- i. Construct an object and find location of Managed Heap**
- ii. Signature instantiated type**
- iii. Scan Managed Heap for object pointers**
- iv. Convert object pointers to raw objects**
- v. ????**
- vi. PROFIT**

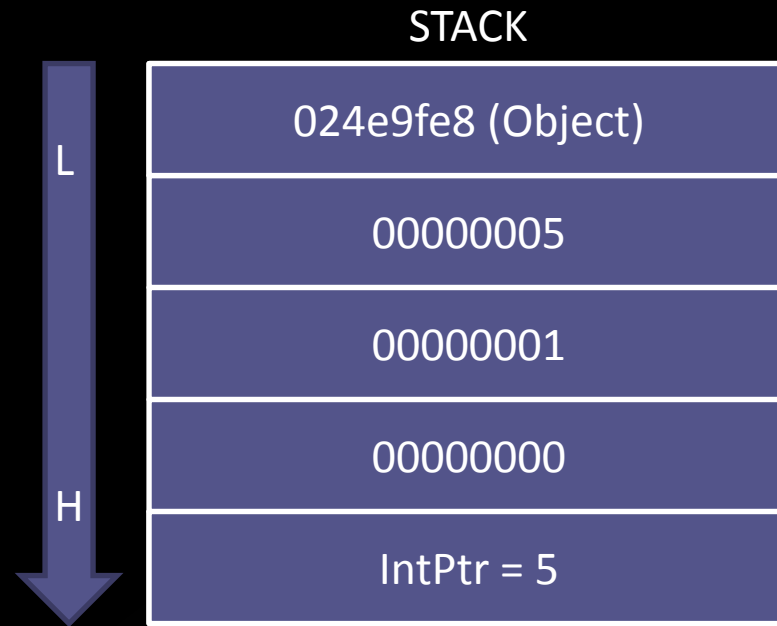
Construct an Object

Use Reflection to invoke a constructor

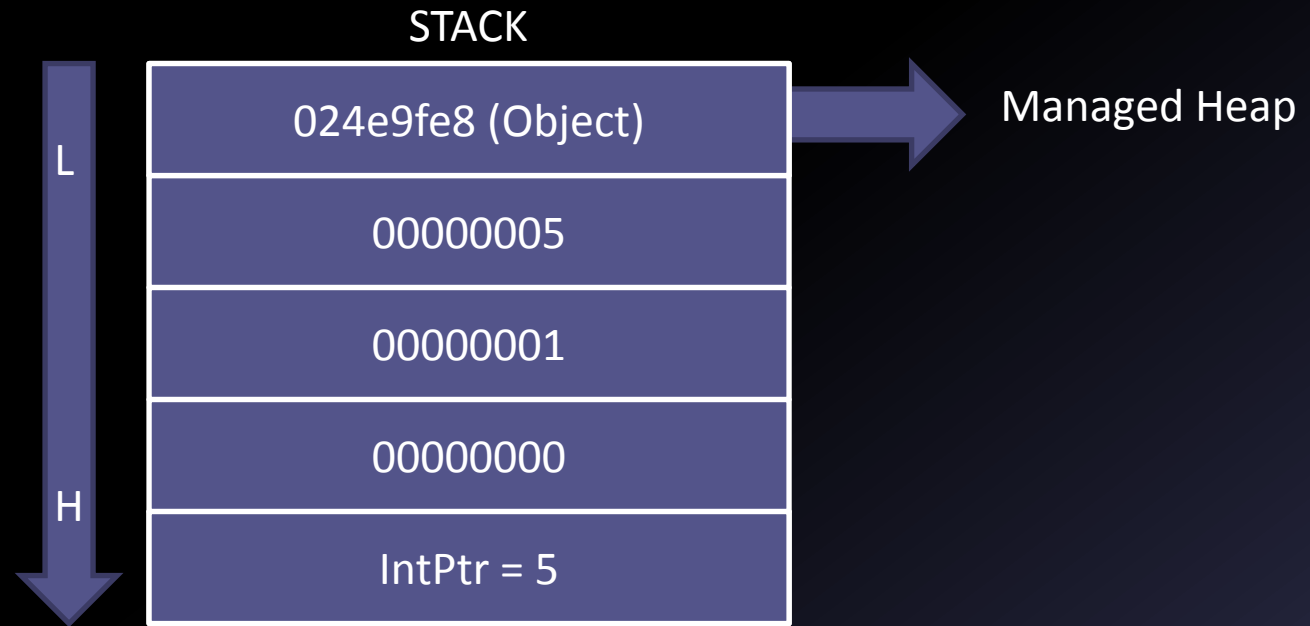
Can instantiate any object

**If a constructor takes other objects,
nullify them**

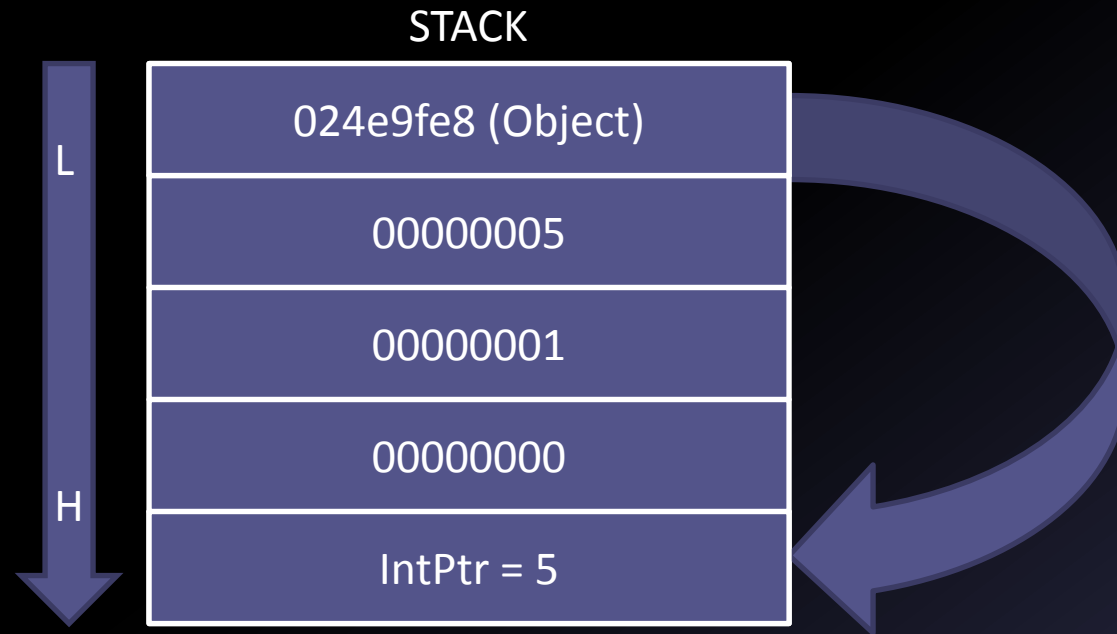
Find location of Managed Heap



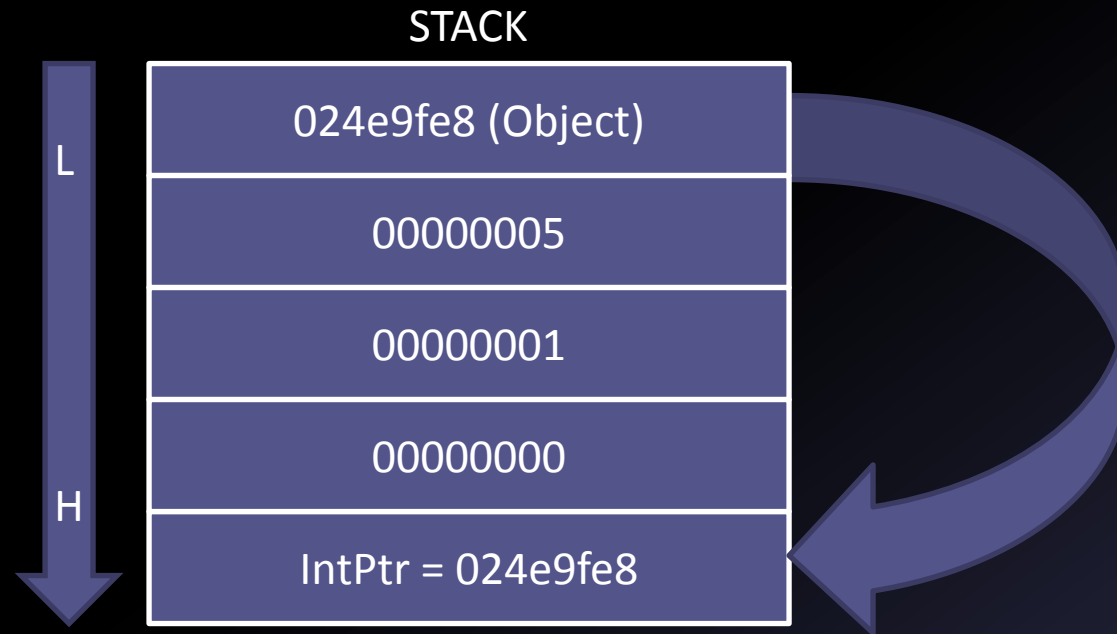
Find location of Managed Heap



Find location of Managed Heap



Find location of Managed Heap



Finding Objects at Runtime

- i. Construct an object and find location of Managed Heap
- ii. Signature instantiated type
- iii. Scan Managed Heap for object pointers
- iv. Convert object pointers to raw objects
- v. ????
- vi. PROFIT

Signature instantiated type

Object Instances contain a Method Table pointer to their corresponding type.

```
0:009> dd 024e9fe8
024e9fe8 00774828 0000038c 00000001 00000000
```

(x86)

Bytes 0-3 are the Method Table (MT)

Bytes 4-7 in MT is Instance Size

Signature instantiated type

Object Instances contain a Method Table pointer to their corresponding type.

```
0:008> dd 00000000024e9fe8  
00000000`0286b8e0 ea774828 000007fe
```

(x64)

Bytes 0-7 are the Method Table (MT)

Bytes 8-11 in MT is Instance Size

Finding Objects at Runtime

- i. Construct an object and find location of Managed Heap
- ii. Signature instantiated type
- iii. Scan Managed Heap for object pointers
- iv. Convert object pointers to raw objects
- v. ????
- vi. PROFIT

Scan Managed Heap

Scan down incrementing by size of object

Scan linearly up to top of heap

Compare object's Method Table to the reference

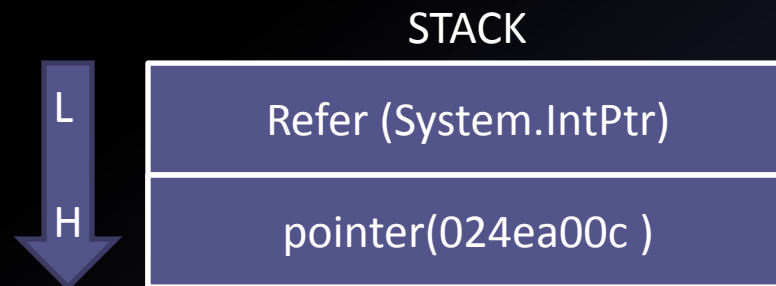
If they match, get IntPtr address of object

Finding Objects at Runtime

- i. Construct an object and find location of Managed Heap
- ii. Signature instantiated type
- iii. Scan Managed Heap for object pointers
- iv. Convert object pointers to raw objects
- v. ????
- vi. PROFIT

Convert object ptr -> raw obj

```
public static object GetInstance(IntPtr ptrIN)
{
    object refer = ptrIN.GetType();
    IntPtr pointer = ptrIN;
    unsafe
    {
        *(&pointer - 1) = *(&pointer);
    }
    return refer;
}
```



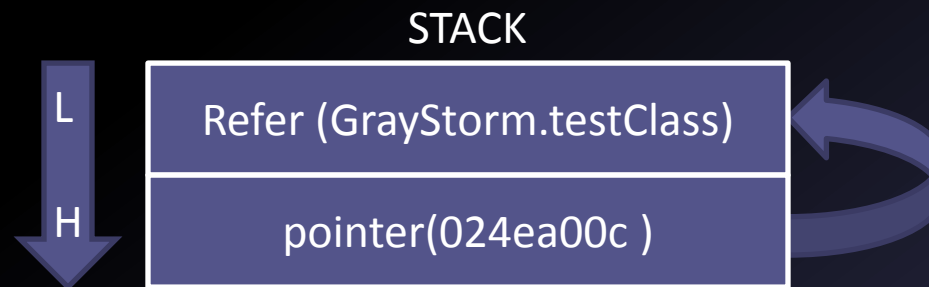
Convert object ptr -> raw obj

```
public static object GetInstance(IntPtr ptrIN)
{
    object refer = ptrIN.GetType();
    IntPtr pointer = ptrIN;
    unsafe
    {
        *(&pointer - 1) = *(&pointer);
    }
    return refer;
}
```



Convert object ptr -> raw obj

```
public static object GetInstance(IntPtr ptrIN)
{
    object refer = ptrIN.GetType();
    IntPtr pointer = ptrIN;
    unsafe
    {
        *(&pointer - 1) = *(&pointer);
    }
    return refer;
}
```



Finding Objects at Runtime

- i. Construct an object and find location of Managed Heap
- ii. Signature instantiated type
- iii. Scan Managed Heap for object pointers
- iv. Convert object pointers to raw objects
- v. ????
- vi. PROFIT

????

LEARN



PROFIT



Superpowers and Things?

✦ Change Keys

✦ Change Fields / Properties

✦ Call Methods

✦ With arguments!

Automation

Automation

GrayFrost can be used with automated payloads

```
GrayKernel>autoFrost.py
Usage: autoFrost.py <C# Payload.exe> <GrayFrost.sln file>

GrayKernel>autoFrost.py autoThink.exe grayfrost\\GrayFrost.sln
[+] Building Payload into embeddable array
[+] Writing GrayFrostCSharp\\payload.cs
[+] Building GrayFrostCSharp
[+] Writing Slate.h
[+] Building GrayFrost{32,64}.dll
[+] GrayFrost finished building
```


Constructing Attack Chains

How to construct attack chains

Gray Wolf / IL Decompiler

- Find Methods, Fields & Properties of interest
- Locate meaningful objects
- Discover high level control flow

Gray Storm “Debugging” functionality

- Breakpoint at constructors or methods from Method Pointers
- Use with WinDbg

Utilize DLL Hijacking!

Hybrid .NET/ASM Attacks

- ✦ Hybrid C#/ASM code in .NET
- ✦ Encrypting .NET payloads and unwinding
- ✦ Encrypting ASM Payloads

Payload System

C# is easy

**Can use Gray Frost in any
application**

Low and High level gap is easy

.NET Hacking Space

Small

Few tools

Mostly hacking WoW

Lots of PowerShell

Previous DEF CON talks

DEF CON 18 & 19 - Jon McCoy

Conclusion

- ✦ Arbitrary .NET applications can be injected and changed
- ✦ New .NET attack possibilities
- ✦ New tools that support automation
- ✦ Get Gray Frost and Storm
github.com/graykernel

Questions?

Contact Me

✦ [@TTimzen](https://twitter.com/TTimzen)

✦ <https://www.tophertimzen.com>

Get Gray Frost and Storm

✦ github.com/graykernel

White Papers

✦ Hijacking Arbitrary .NET Application Control Flow

✦ Acquiring .NET Objects from the Managed Heap