AVENGERCON II

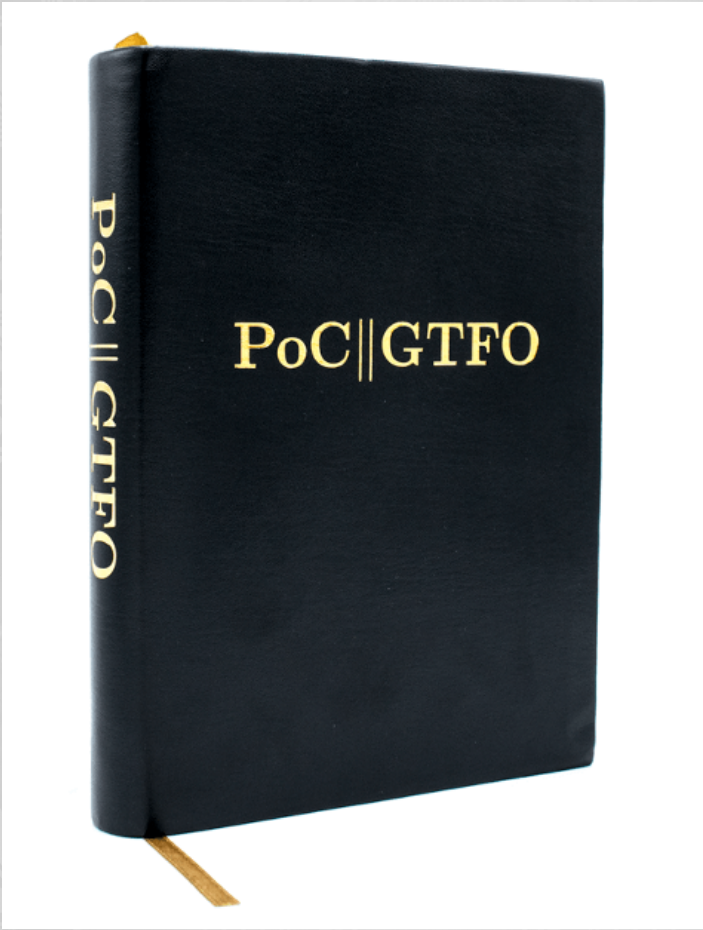
Exploit Mitigations in Windows 10 Creators Update

Christian Sharpsten

November 15, 2017

Table of Contents

POC || GTFO



https://github.com/shareef12

Exploit Primitives

- Some kind of write or memory corruption
- Arbitrary write
- Info leak or arbitrary read?
- ROP/limited code execution
- Arbitrary code execution

Pre Windows 10

- EMET - DEP
- EMET - Structured Exception Handling Overwrite Protection (SEHOP)
- EMET - Mandatory ASLR
- EMET - ROP (detects stack pivot)
- EMET - NullPage (farewell null-ptr deref exploits)
- EMET - Export Address Table Filtering (hw bp on EAT)
- Control Flow Guard - breaks vtable overwrites to ROP

Windows 10 1703

- EMET built in
- Better Control Flow Guard (Export Suppression)
- No child processes
- Disallow win32k syscalls
- Signing policies (DeviceGuard, UMCI)
- Code Integrity Guard
- Arbitrary Code Guard
- Return-Flow Guard??

Why do we care about CIG/ACG?

- Modular exploitation payloads
- Code injection

How to enable CIG

There are three ways to enable CIG, in order of increasing precedence.

1. Runtime (current process)

   ```
   SetProcessMitigationPolicy(
   ProcessSignaturePolicy, ...)
   ```

2. Runtime (child processes)

   ```
   UpdateProcThreadAttribute(
   PROC_THREAD_ATTRIBUTE_MITIGATION_POLICY,
   PROCESS_CREATION_MITIGATION_POLICY_\
   BLOCK_NON_MICROSOFT_BINARIES_ALWAYS_ON, ...)
   ```

3. Registry - `MitigationOptions` Image File Execution Option for an EXE

How to enable ACG

There are three ways to enable ACG, in order of increasing precedence.

1. Runtime (current process)

   ```
   SetProcessMitigationPolicy(
   ProcessDynamicCodePolicy, ...)
   ```

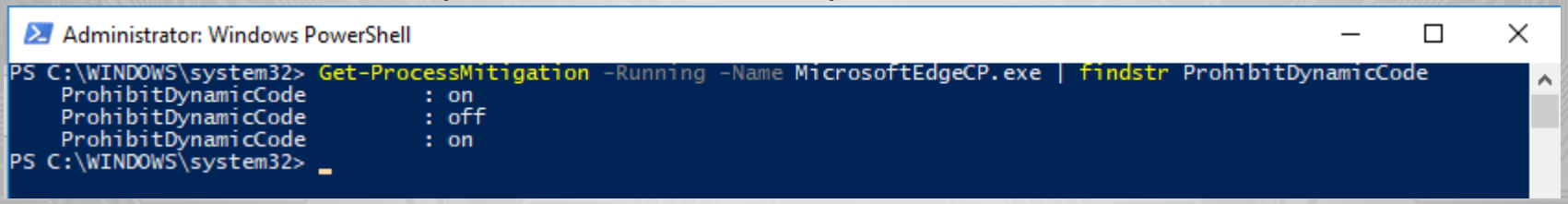
2. Runtime (child processes)

   ```
   UpdateProcThreadAttribute(
   PROC_THREAD_ATTRIBUTE_MITIGATION_POLICY,
   PROCESS_CREATION_MITIGATION_POLICY_\
   PROHIBIT_DYNAMIC_CODE_ALWAYS_ON, ...)
   ```

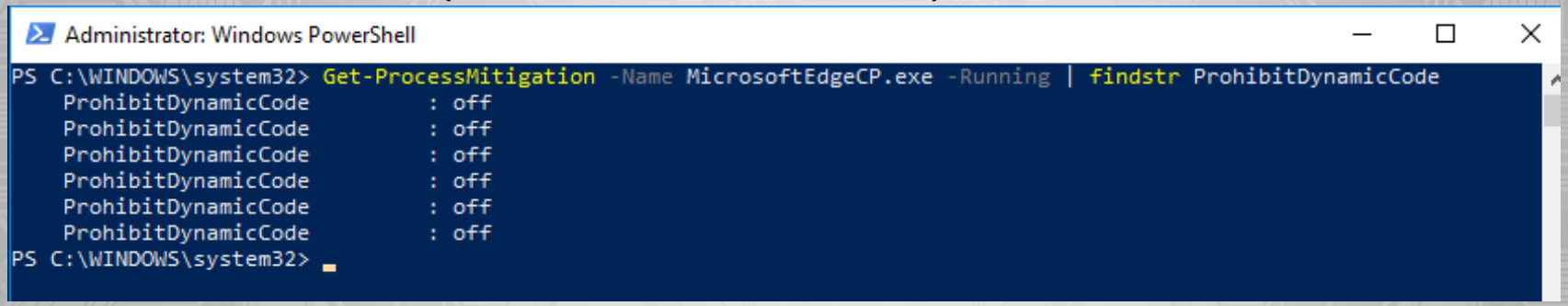
3. Registry - `MitigationOptions` Image File Execution Option for an EXE

Where is ACG enabled

Windows 10 1703 (Creators Update)



Windows 10 1709 (Fall Creators Update)



CIG/ACG Implementation Details

```
0: kd> dt _EPROCESS
   ...
   +0x828 MitigationFlags  : Uint4B
   +0x828 MitigationFlagsValues : <unnamed-tag>
   +0x82c MitigationFlags2 : Uint4B
   +0x82c MitigationFlags2Values : <unnamed-tag>
```

CIG/ACG Implementation Details

```
0: kd> dt _EPROCESS MitigationFlagsValues.
nt!_EPROCESS
    +0x828 MitigationFlagsValues  :
        ...
        +0x000 DisableDynamicCode     : Pos 8, 1 Bit
        +0x000 DisableDynamicCodeAllowOptOut : Pos 9, 1 Bit
        +0x000 DisableDynamicCodeAllowRemoteDowngrade : Pos 10, 1 Bit
        +0x000 AuditDisableDynamicCode : Pos 11, 1 Bit
        ...
        +0x000 SignatureMitigationOptIn : Pos 23, 1 Bit
        +0x000 AuditBlockNonMicrosoftBinaries : Pos 24, 1 Bit
        +0x000 AuditBlockNonMicrosoftBinariesAllowStore : Pos 25, 1 Bit
```

EMET Replacement?

```
0: kd> dt _EPROCESS MitigationFlags2Values.
    nt!_EPROCESS
    +0x82c MitigationFlags2Values  :
        +0x000 EnableExportAddressFilter : Pos 0, 1 Bit
        +0x000 AuditExportAddressFilter : Pos 1, 1 Bit
        +0x000 EnableExportAddressFilterPlus : Pos 2, 1 Bit
        +0x000 AuditExportAddressFilterPlus : Pos 3, 1 Bit
        +0x000 EnableRopStackPivot     : Pos 4, 1 Bit
        +0x000 AuditRopStackPivot      : Pos 5, 1 Bit
        +0x000 EnableRopCallerCheck    : Pos 6, 1 Bit
        +0x000 AuditRopCallerCheck     : Pos 7, 1 Bit
        +0x000 EnableRopSimExec        : Pos 8, 1 Bit
        +0x000 AuditRopSimExec         : Pos 9, 1 Bit
        +0x000 EnableImportAddressFilter : Pos 10, 1 Bit
        +0x000 AuditImportAddressFilter : Pos 11, 1 Bit
```

CIG/ACG Implementation Details

```
1: kd> ba w 4 /p @$proc @$proc+828
1: kd> p
Breakpoint 0 hit
nt!RtlInterlockedSetClearBits+0x1c:
fffff801‘a68222f8 448bc8          mov     r9d,eax
1: kd> kc
# Call Site
00 nt!RtlInterlockedSetClearBits
01 nt!NtSetInformationProcess
02 nt!KiSystemServiceCopyEnd
03 ntdll!NtSetInformationProcess
04 KERNELBASE!SetProcessMitigationPolicy
05 host!ProhibitDynamicCode
...
2d host!main
1: kd> dd @$proc+828 l1
ffffb48f‘aaa51a68  00000120
```

ACG from Usermode

ACG (kernel enforced $W \oplus X$) + CIG

- Code is immutable
- Data cannot become code

Related memory manager APIs should now fail with `ERROR_DYNAMIC_CODE_BLOCKED`

```
VirtualProctect(CodePage, ..., PAGE_READWRITE)
VirtualProtect(DataPage, ..., PAGE_EXECUTE*)
VirtualAlloc(CodePage, ..., PAGE_EXECUTE*)
MapViewOfFile(hMapping, FILE_MAP_EXECUTE, ...)
...
```

ACG Disable

Why don't we just turn it off?

```
SetProcessMitigationPolicy(
    ProcessDynamicCodePolicy, ...)
```

ACG Disable

Why don't we just turn it off?

```
SetProcessMitigationPolicy(
    ProcessDynamicCodePolicy, ...)
```

ERROR_ACCESS_DENIED

ACG VirtualProtect

VirtualProtect a code page to RWX?

```
VirtualProtect(ThisFunction, PAGE_EXECUTE_READWRITE)
```

ACG VirtualProtect

VirtualProtect a code page to RWX?

```
VirtualProtect(ThisFunction, PAGE_EXECUTE_READWRITE)
```

ERROR_DYNAMIC_CODE_BLOCKED

ACG VirtualProtect

VirtualProtect a code page to RWX?

```
VirtualProtect(ThisFunction, PAGE_EXECUTE_READWRITE)
```

ERROR_DYNAMIC_CODE_BLOCKED

Alloc RW, then protect to RX?

```
addr = VirtualAlloc(PAGE_SIZE, PAGE_READWRITE)
VirtualProtect(addr, PAGE_READ_EXECUTE)
```

ACG VirtualProtect

VirtualProtect a code page to RWX?

```
VirtualProtect(ThisFunction, PAGE_EXECUTE_READWRITE)
```

ERROR_DYNAMIC_CODE_BLOCKED

Alloc RW, then protect to RX?

```
addr = VirtualAlloc(PAGE_SIZE, PAGE_READWRITE)
VirtualProtect(addr, PAGE_READ_EXECUTE)
```

ERROR_DYNAMIC_CODE_BLOCKED

ACG VirtualAlloc

```
VirtualAlloc(PAGE_SIZE, PAGE_EXECUTE_READWRITE)
```

ACG VirtualAlloc

```
VirtualAlloc(PAGE_SIZE, PAGE_EXECUTE_READWRITE)
```

ERROR_DYNAMIC_CODE_BLOCKED

ACG MapViewOfFile

Single section with separate mappings (RW + RX)?

```
hMapping = CreateFileMapping(PAGEFILE, PAGE_EXECUTE_READWRITE)
addr1 = MapViewOfFile(hMapping, FILE_MAP_WRITE)
addr2 = MapViewOfFile(hMapping, FILE_MAP_EXECUTE)
```

ACG MapViewOfFile

Single section with separate mappings (RW + RX)?

```
hMapping = CreateFileMapping(PAGEFILE, PAGE_EXECUTE_READWRITE)
addr1 = MapViewOfFile(hMapping, FILE_MAP_WRITE)
addr2 = MapViewOfFile(hMapping, FILE_MAP_EXECUTE)
```

ERROR_DYNAMIC_CODE_BLOCKED

ACG MapViewOfFile

Single section with separate mappings (RW + RX)?

```
hMapping = CreateFileMapping(PAGEFILE, PAGE_EXECUTE_READWRITE)
addr1 = MapViewOfFile(hMapping, FILE_MAP_WRITE)
addr2 = MapViewOfFile(hMapping, FILE_MAP_EXECUTE)
```

ERROR_DYNAMIC_CODE_BLOCKED

What if the contents are backed by a file?

```
WriteFile("testfile", "\xeb\xfe")
hMapping = CreateFileMapping("testfile", PAGE_EXECUTE_READ)
MapViewOfFile(hMapping, FILE_MAP_EXECUTE)
```

ACG MapViewOfFile

Single section with separate mappings (RW + RX)?

```
hMapping = CreateFileMapping(PAGEFILE, PAGE_EXECUTE_READWRITE)
addr1 = MapViewOfFile(hMapping, FILE_MAP_WRITE)
addr2 = MapViewOfFile(hMapping, FILE_MAP_EXECUTE)
```

ERROR_DYNAMIC_CODE_BLOCKED

What if the contents are backed by a file?

```
WriteFile("testfile", "\xeb\xfe")
hMapping = CreateFileMapping("testfile", PAGE_EXECUTE_READ)
MapViewOfFile(hMapping, FILE_MAP_EXECUTE)
```

ERROR_DYNAMIC_CODE_BLOCKED

ACG from Kernelmode

- Attacking ACG should be much easier from kernelmode
- Need to disable ACG or allocate executable memory on behalf of a userspace loader
- How can we do this?

ACG Disable

Can we toggle EPROCESS.DisableDynamicCode to turn off ACG?

```
PULONG MitigationFlagsValues = (PULONG)(((ULONG_PTR)Process) +
    EPROCESS_MITIGATION_FLAGS_VALUES_OFFSET);
ClearFlag(*MitigationFlagsValues, EPROCESS_DISABLE_DYNAMIC_CODE_MASK);
```

ACG Disable

Can we toggle EPROCESS.DisableDynamicCode to turn off ACG?

```
PULONG MitigationFlagsValues = (PULONG)(((ULONG_PTR)Process) +
    EPROCESS_MITIGATION_FLAGS_VALUES_OFFSET);
ClearFlag(*MitigationFlagsValues, EPROCESS_DISABLE_DYNAMIC_CODE_MASK);
```

SUCCESS

ACG Disable

Can we toggle EPROCESS.DisableDynamicCode to turn off ACG?

```
PULONG MitigationFlagsValues = (PULONG)(((ULONG_PTR)Process) +
    EPROCESS_MITIGATION_FLAGS_VALUES_OFFSET);
ClearFlag(*MitigationFlagsValues, EPROCESS_DISABLE_DYNAMIC_CODE_MASK);
```

SUCCESS

Will thread opt-out work even if we didn't explicitly allow it?

```
PULONG CrossThreadFlags = (PULONG)(((ULONG_PTR)Thread) +
    ETHREAD_CROSS_THREAD_FLAGS_OFFSET);
SetFlag(*CrossThreadFlags, ETHREAD_DISABLE_DYNAMIC_CODE_OPT_OUT_MASK);
```

ACG Disable

Can we toggle EPROCESS.DisableDynamicCode to turn off ACG?

```
PULONG MitigationFlagsValues = (PULONG)(((ULONG_PTR)Process) +
    EPROCESS_MITIGATION_FLAGS_VALUES_OFFSET);
ClearFlag(*MitigationFlagsValues, EPROCESS_DISABLE_DYNAMIC_CODE_MASK);
```

SUCCESS

Will thread opt-out work even if we didn't explicitly allow it?

```
PULONG CrossThreadFlags = (PULONG)(((ULONG_PTR)Thread) +
    ETHREAD_CROSS_THREAD_FLAGS_OFFSET);
SetFlag(*CrossThreadFlags, ETHREAD_DISABLE_DYNAMIC_CODE_OPT_OUT_MASK);
```

SUCCESS

ACG NtProtectVirtualMemory or ZwAllocateVirtualMemory

- No ZwProtectVirtualMemory, NtProtectVirtualMemory and MmProtectVirtualMemory aren't exported.
- `ZwAllocateVirtualMemory(PAGE_EXECUTE_READWRITE)` fails with STATUS_DYNAMIC_CODE_BLOCKED

ACG Memory Descriptor List (MDL)

Use an MDL to map RX memory into System space and write shellcode

```
mdl = MmProbeAndLockPages(UserAddress, Size)
SystemAddr = MmGetSystemAddressForMdlSafe(mdl)
memcpy(SystemAddr, shellcode)
```

ACG Memory Descriptor List (MDL)

Use an MDL to map RX memory into System space and write shellcode

```
mdl = MmProbeAndLockPages(UserAddress, Size)
SystemAddr = MmGetSystemAddressForMdlSafe(mdl)
memcpy(SystemAddr, shellcode)
```

SUCCESS

ACG Toggle Write Protect

- System-wide write protection can be enabled/disabled through cr0 bit 16.
- Toggle this to allow us to write shellcode to userspace RX memory?

```
cr0 = __readcr0();
ClearFlag(cr0, CR0_WRITE_PROTECT_MASK);
__writecr0(cr0);
memcpy(Address, Buffer, Size);
```

ACG Toggle Write Protect

- System-wide write protection can be enabled/disabled through cr0 bit 16.
- Toggle this to allow us to write shellcode to userspace RX memory?

```
cr0 = __readcr0();
ClearFlag(cr0, CR0_WRITE_PROTECT_MASK);
__writecr0(cr0);
memcpy(Address, Buffer, Size);
```

SUCCESS

ACG PTE Manipulation

- What if we start manipulating paging structures directly?
- Read page table base from cr3, and traverse physical pages (beats KASLR) to get to a PTE.
- Disable NoExecute bit to get a RWX page.

ACG PTE Manipulation

- What if we start manipulating paging structures directly?
- Read page table base from cr3, and traverse physical pages (beats KASLR) to get to a PTE.
- Disable NoExecute bit to get a RWX page.

SUCCESS

Summary

Microsoft has done a lot to make initial exploitation harder. Even if you have bugs they may not all be exploitable in a modular way.

From Usermode:

- Out of luck

From Kernelmode:

- Disable process ACG
- Disable thread ACG
- Write to existing RX memory (MDLs + WriteProtect)
- Convert RW pages to RWX

References


David Weston and Matt Miller.
Microsoft's strategy and technology improvements toward mitigating arbitrary native code execution.


Swamy Shivaganga Nagaraju.
Agility with security mitigations in windows 10.