Windows

Agility with Security Mitigations in Windows 10

Swamy Shivaganga Nagaraju,
Vulnerabilities and Mitigations Team,
Microsoft Security Response Center (MSRC)

Nullcon - March, 2017

Microsoft

Agenda

- Windows release cadence.
  - ➢ Pre & post Windows 10 cadence.

- Security mitigations.
  - ➢ What & impact of security mitigations.
  - ➢ Overview of some mitigations.

Acknowledgements

To all

Security folks & Developers

@Microsoft committed to

SECURE

Microsoft products & platform for the

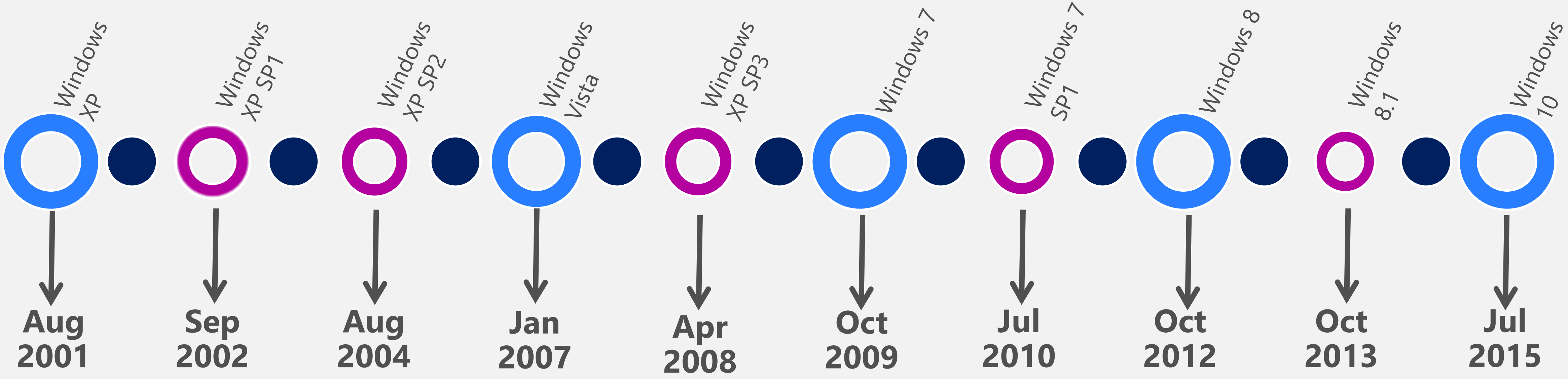
World

Windows release cadence

Microsoft

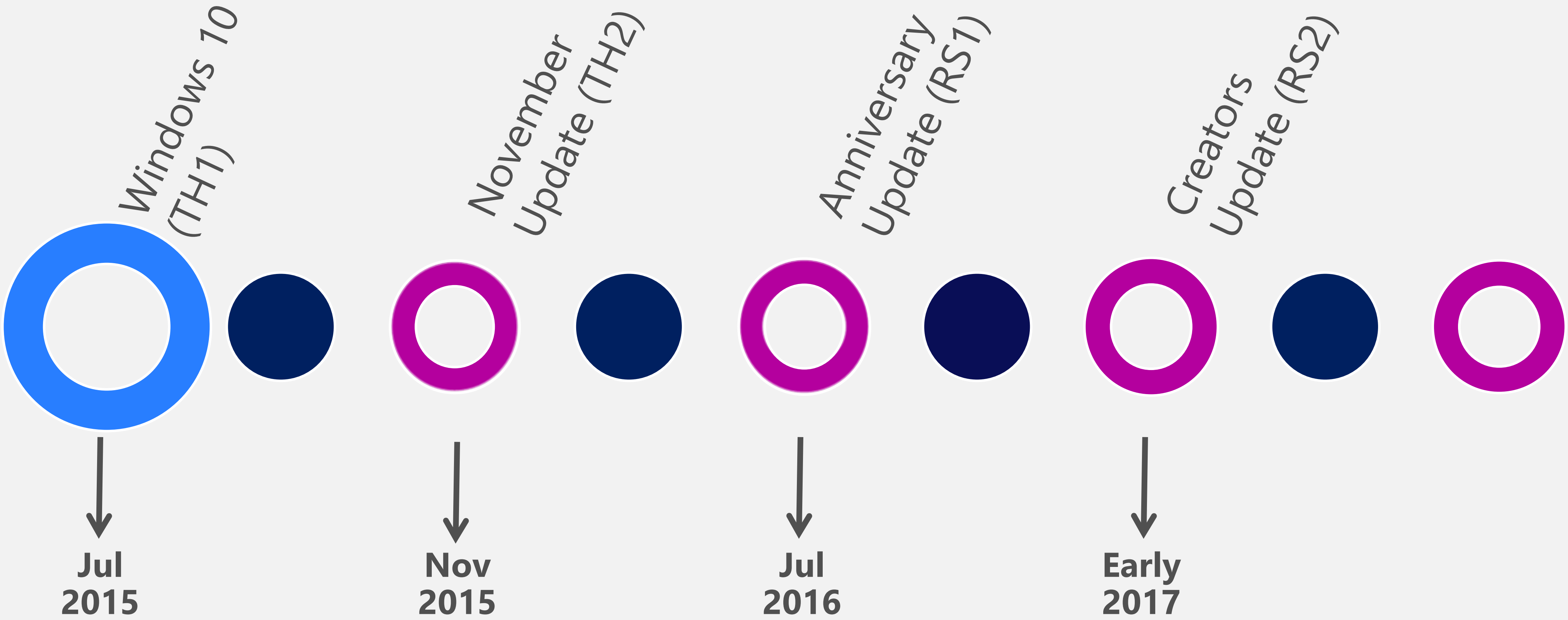
Before Windows 10

- Major releases were less frequent, one every few years.
- Mitigations are shipped mostly in major releases, unlike vulnerabilities which gets patched regularly.



Windows 10

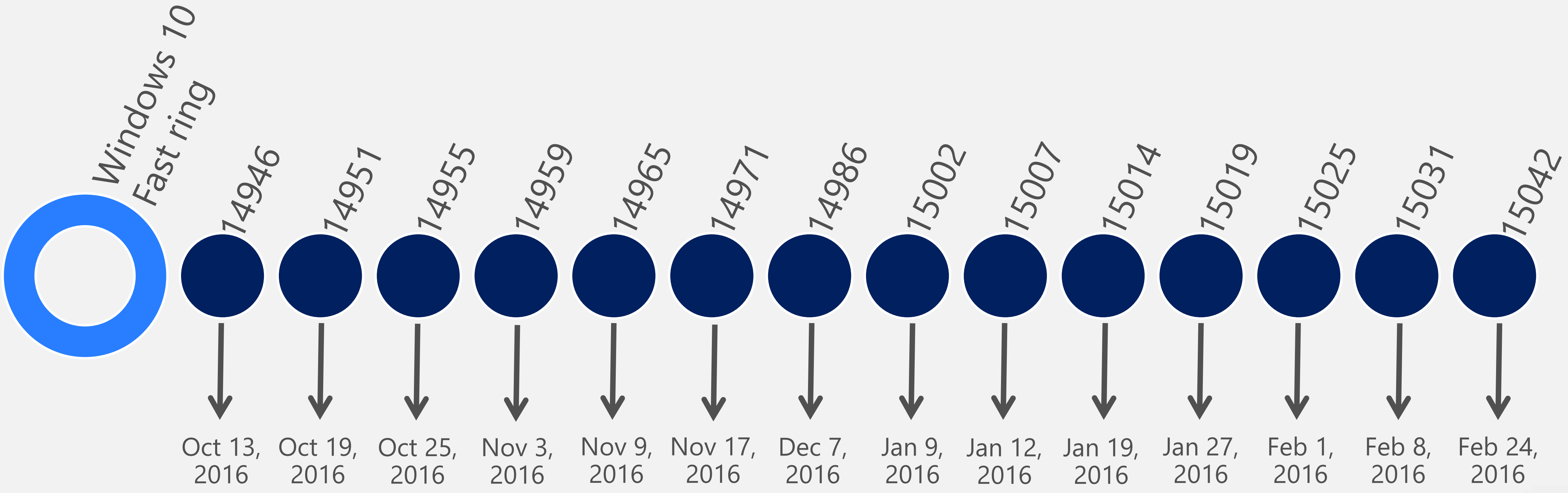
- Increase frequency in major updates.
  - ➢ More chances for adding/improving features.



Microsoft

Windows Insider Program

- Multiple rings with different frequency.
  - ➢ Fast, Slow & Release preview.
- Insiders will get features as and when it is available.



Windows 10 Fast ring

14946	Oct 13, 2016
14951	Oct 19, 2016
14955	Oct 25, 2016
14959	Nov 3, 2016
14965	Nov 9, 2016
14971	Nov 17, 2016
14986	Dec 7, 2016
15002	Jan 9, 2016
15007	Jan 12, 2016
15014	Jan 19, 2016
15019	Jan 27, 2016
15025	Feb 1, 2016
15031	Feb 8, 2016
15042	Feb 24, 2016

Security Mitigations

What is a Security Mitigation?

- A feature to disrupt exploitation.

- Mitigations make certain exploitation techniques and vulnerability classes harder or impossible to use.

- Different class of mitigations:
  - ➢ **Hard mitigations:** Harder or impossible to bypass. Typically disrupts an entire vulnerability class.
  - ➢ **Soft mitigations:** Makes exploitation harder but can be bypassed with stronger primitives.
  - ➢ **Tactical mitigations:** Aimed at disrupting specific exploit techniques.

Scope of Security Mitigations

- System mitigations.
  - ➢ These are to harden the entire platform.
  - ➢ They are mostly available by default for all the applications/processes.

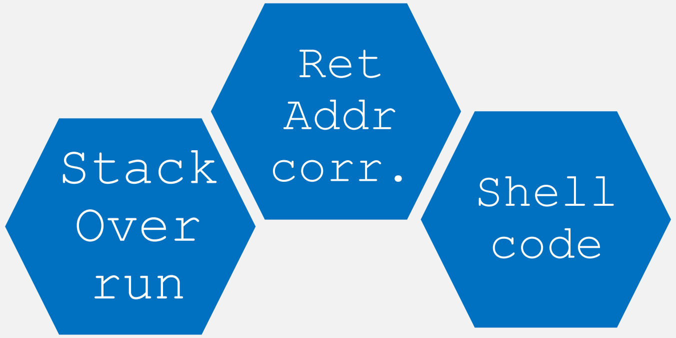
- Process mitigations.
  - ➢ Optional mitigations that a process or application can opt-in to.
  - ➢ It may be enabled by default for few applications (browsers, worker process etc..).

- Application specific mitigations.
  - ➢ Specific to a given application.
  - ➢ Application itself is modified to add more safeguards against exploitation.

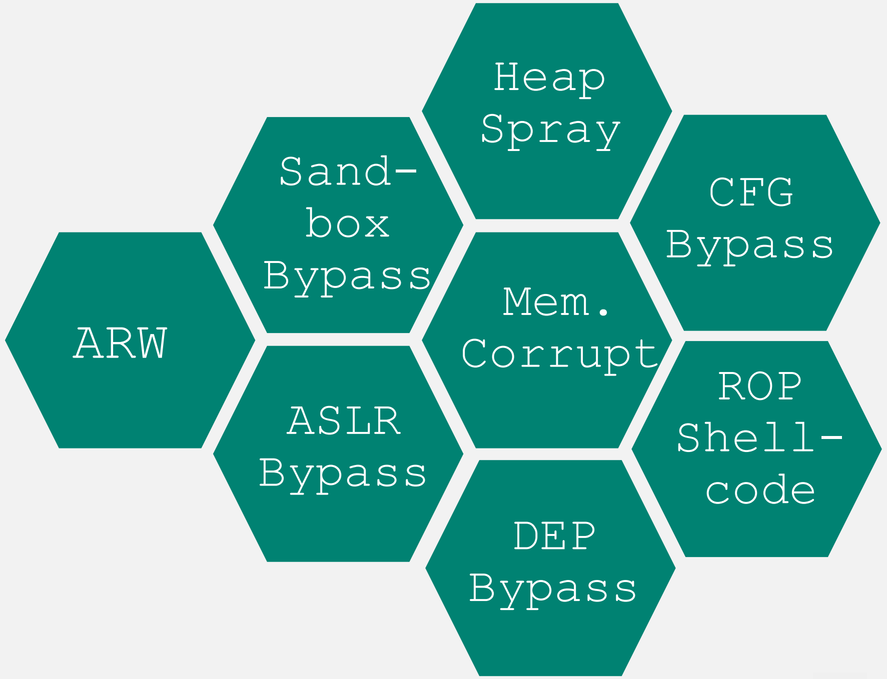
Effects of Security Mitigations

- Attack surface reduction.
  - ➢ Reduced target for attackers.
- Bug class elimination.
  - ➢ Takes soft targets out of the picture.
- Eliminates exploitation techniques.
  - ➢ New techniques need to be found.
- Reduces impact of vulns by isolation/ containment.
  - ➢ Sandboxing the target.
- Overall makes exploitation harder.
  - ➢ A decade back it used to be 3 steps to pwn, on average it now takes more than 8 steps.

Exploitation a decade back …



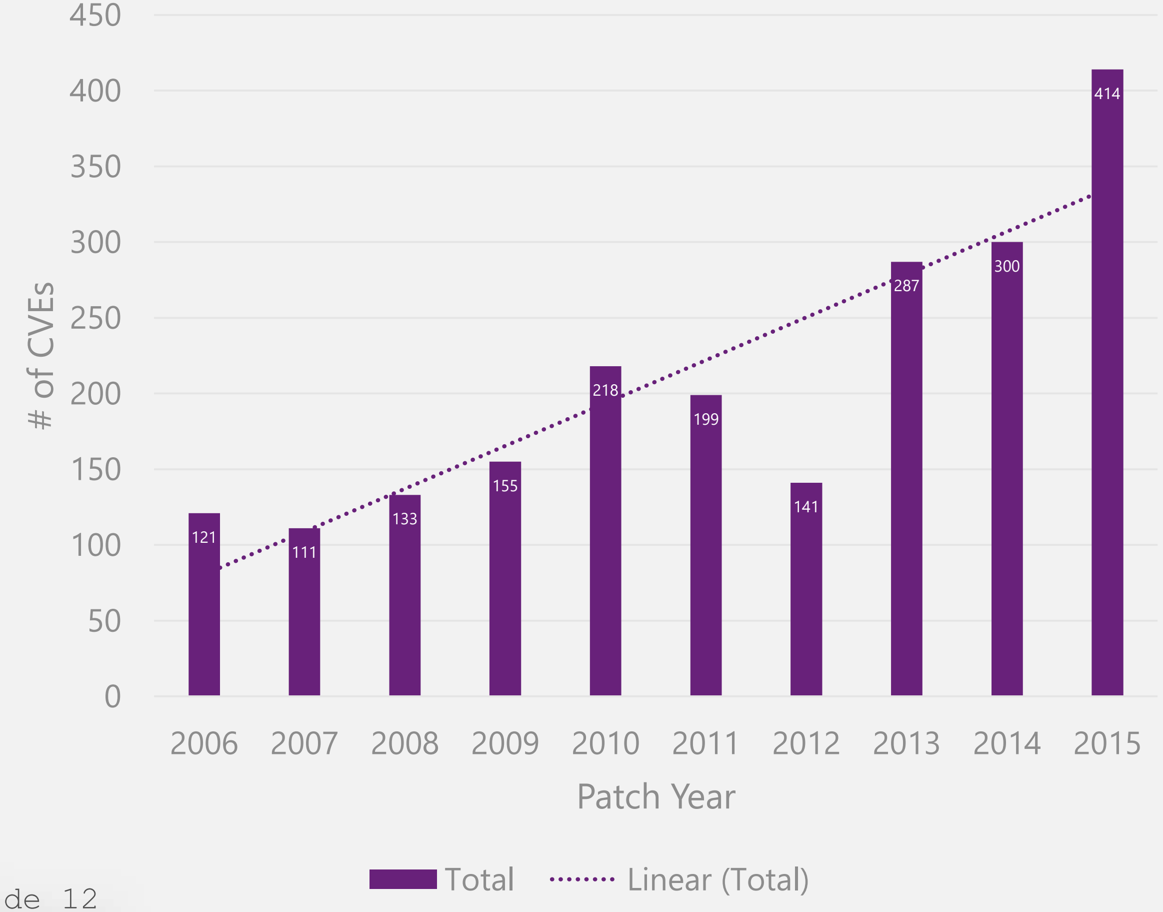
Exploitation today …



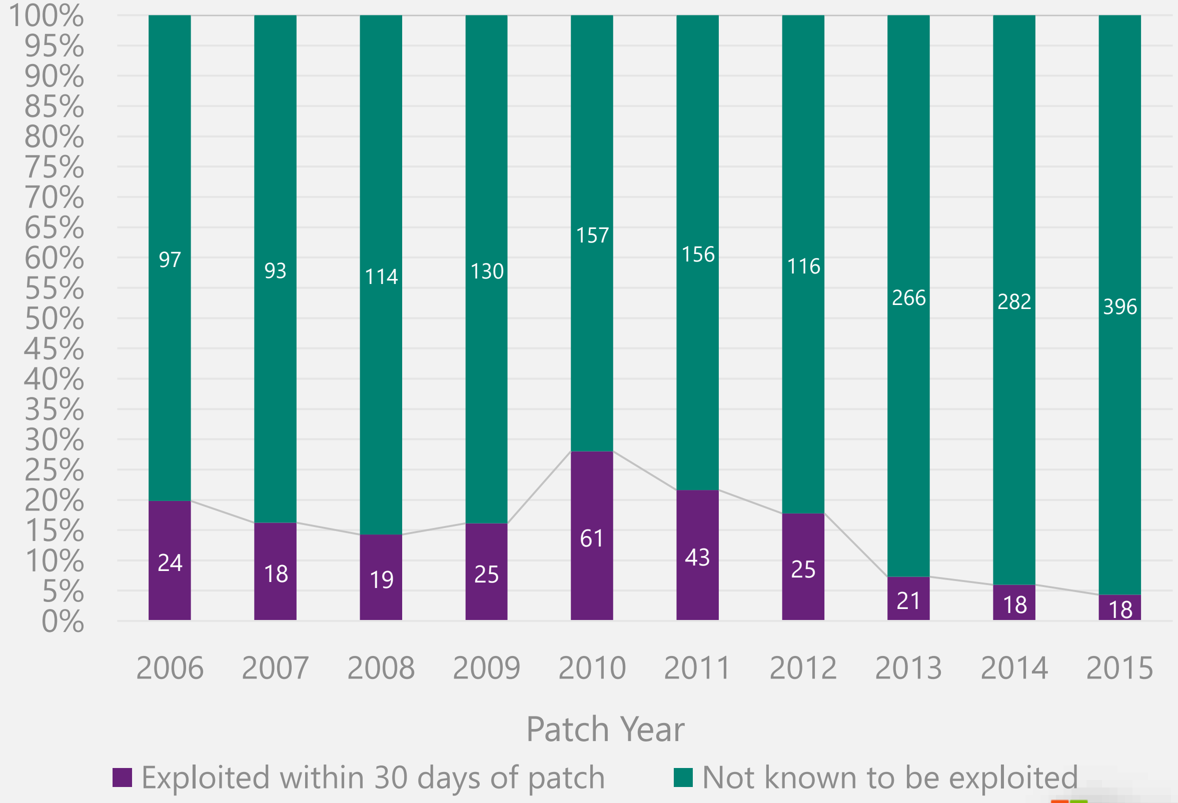
Impact of Security Mitigations

- Vulnerabilities are increasing while evidence of actual exploits is decreasing due to mitigation investments

# of Microsoft RCE/EOP CVEs by patch year

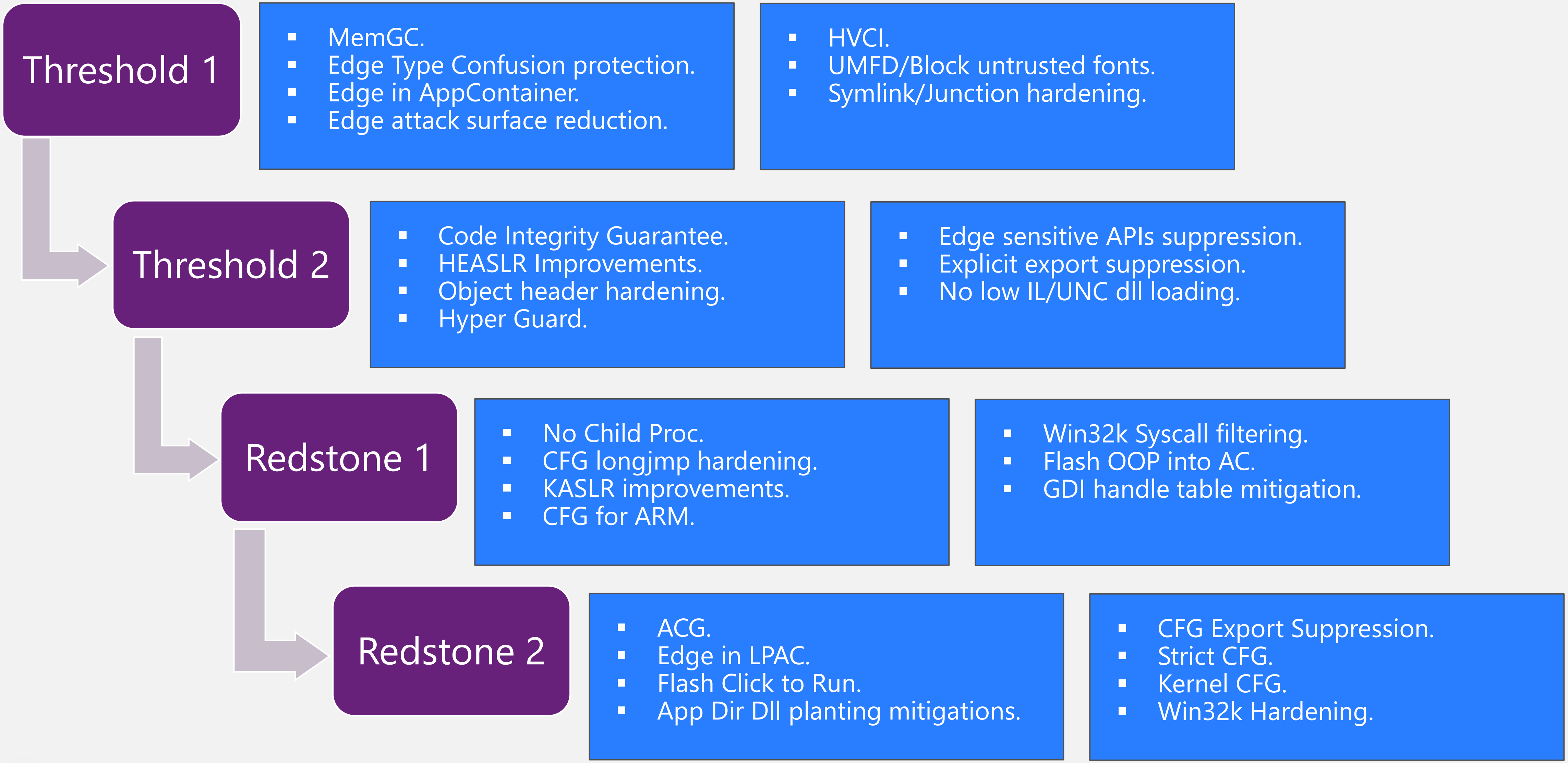


% of Microsoft RCE & EOP CVEs exploited within 30 days of patch



Overview on few Mitigations

Mitigations in Windows 10



Threshold 1

- MemGC.
- Edge Type Confusion protection.
- Edge in AppContainer.
- Edge attack surface reduction.

- HVCI.
- UMFD/Block untrusted fonts.
- Symlink/Junction hardening.

Threshold 2

- Code Integrity Guarantee.
- HEASLR Improvements.
- Object header hardening.
- Hyper Guard.

- Edge sensitive APIs suppression.
- Explicit export suppression.
- No low IL/UNC dll loading.

Redstone 1

- No Child Proc.
- CFG longjmp hardening.
- KASLR improvements.
- CFG for ARM.

- Win32k Syscall filtering.
- Flash OOP into AC.
- GDI handle table mitigation.

Redstone 2

- ACG.
- Edge in LPAC.
- Flash Click to Run.
- App Dir Dll planting mitigations.

- CFG Export Suppression.
- Strict CFG.
- Kernel CFG.
- Win32k Hardening.

Mitigation – User Mode Font Driver (UMFD)

Font processed in kernel mode: Memory corruption in font processing could be hit remotely via untrusted fonts.

- With UMFD it is moved to user mode.
  - ➢ Runs under an App Container (AC).
- All font vulns through un-trusted fonts are now contained within AC.
  - ➢ No more local EOPs/sandbox escapes using untrusted fonts.
- Un-trusted fonts can also be blocked for a process via ProcessFontDisablePolicy.
  - ➢ DisableNonSystemFonts

Font processing moved to user mode app container

System Mitigation

TH1

Enabled for all Process

Achieves Isolation & Containment

Mitigation – Win32k Syscall Filtering

Win32k exposes huge number of syscalls that can be targeted for EOP/Sandbox escapes: ~1200 Win32k syscalls are available for an application.

- Syscall filtering is done via hard coded whitelist of Win32K syscalls for Edge.
  - ➢ Blocks unnecessary Win32K APIs thus reducing the attack surface.

Removes unwanted Win32k APIs

Application Mitigation

RS1

Enabled for Edge

Reduces Attack Surface

Mitigation – Less Privileged App Container (LPAC)

App Container has access to resources protected with ALL APPLICATION PACKAGES SID: This SID has read permission on all folders by default.

- LPAC is a more restricted version of the App Container.
  - ➢ Denied access by default for everything.
  - ➢ Can access only the secured objects that are granted explicitly to LPAC.

Tightens the AC with default deny everything

Application Mitigation

RS2

Enabled for Edge

Achieves Isolation & Containment

Mitigation – App Dir Dll Planting

App dir is the first location searched while loading the DLL: This can be vulnerable in case of Low IL, Downloads folder.

- A new prefer system32 process mitigation toggles app dir and system32 in the dll search order.
  - ➢ Automatically enabled for the process when the image located under a Low IL or Downloads folder is executed.
- Enabled via the process mitigation option ProcessImageLoadPolicy.

App Dir can be switched with System32

System Mitigation

RS2

Enabled for all Process

Removes Bug Class

Context switch…

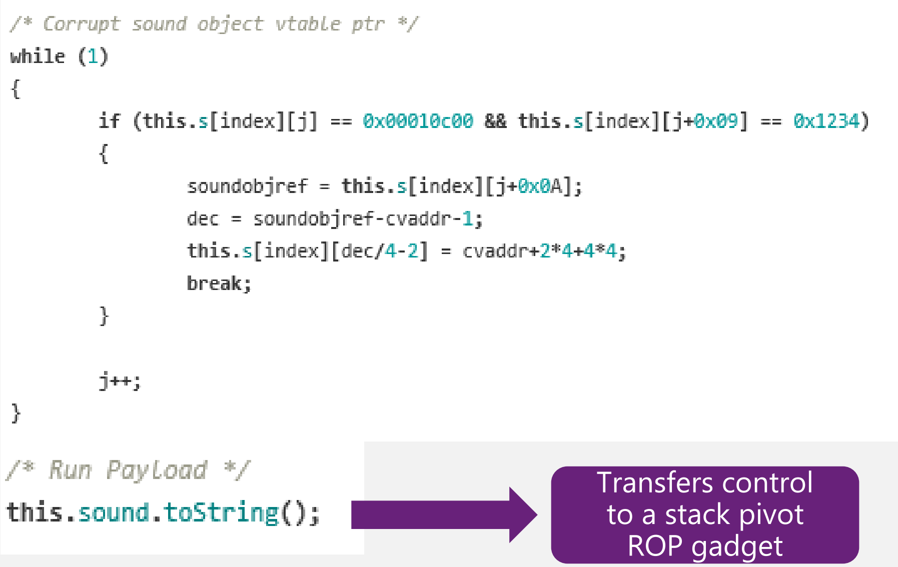
- Modern exploits typically rely on hijacking control-flow to exploit the memory corruption.

- Hijacked control flow will typically leveraged for an arbitrary native code execution.
  - ROP to execute shellcode.
  - Arbitrary dll loading.
  - Process Creation.

Context switch…

- ## Control Flow Guard (CFG)
  - ➢ CFG is our strategy for tackling memory corruption (RCE exploitation).
  - ➢ Indirect calls are validated against a bitmap before transferring the control.

Example control-flow hijack via indirect call to a ROP gadget[1]



```
/* Corrupt sound object vtable ptr */
while (1)
{
        if (this.s[index][j] == 0x00010c00 && this.s[index][j+0x09] == 0x1234)
        {
                soundobjref = this.s[index][j+0x0A];
                dec = soundobjref-cvaddr-1;
                this.s[index][dec/4-2] = cvaddr+2*4+4*4;
                break;
        }

        j++;
}
/* Run Payload */
this.sound.toString();
```

Transfers control to a stack pivot ROP gadget

Compile time → Runtime

```
void Foo(...) {
   // SomeFunc is address-taken
   // and may be called indirectly
   Object->FuncPtr = SomeFunc;
}
```

Metadata is automatically added to the image which identifies functions that may be called indirectly

```
void Bar(...) {
   // Compiler-inserted check to
   // verify call target is valid
   _guard_check_icall(Object->FuncPtr);
   Object->FuncPtr(xyz);
}
```

A lightweight check is inserted prior to indirect calls which will verify that the call target is valid at runtime

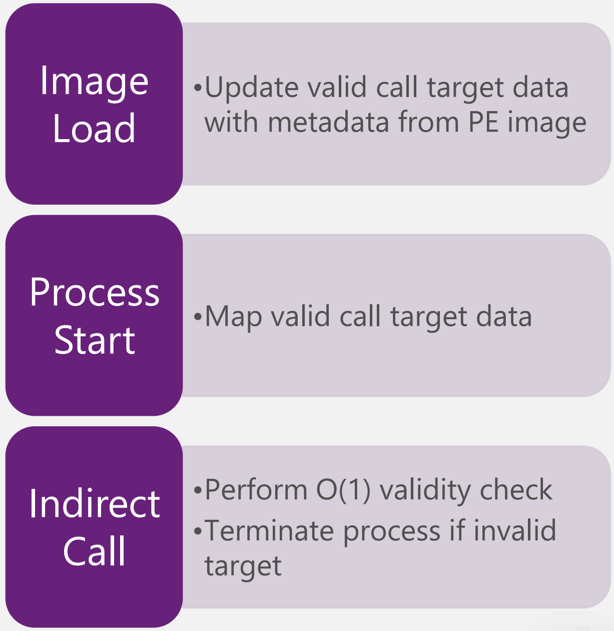


Image Load
- •Update valid call target data with metadata from PE image

Process Start
- •Map valid call target data

Indirect Call
- •Perform O(1) validity check
- •Terminate process if invalid target

Mitigation – CFG Longjmp Hardening

Jump buffer used for longjmp can be corrupted: setjmp stores the state into jmp buffer that is restored with longjmp.

- longjmp transfers are now verified for the valid setjmp targets.
  - ➢ Compile time records the locations of all the setjmp that is used to verify during longjmp.
  - ➢ Setjmp/longjmp is no longer valid indirect icalls.

- Enabled for all Windows binaries.
  - ➢ Enabled by default with /guard:cf in Visual Studio 2015 Update 3.

Longjmp is protected against jmp buffer corruption

Process Mitigation

RS1

Enabled with CFG

Makes Exploitation Harder

Mitigation – CFG Export Suppression

Module exports are valid indirect calls (icall) by default with CFG: Many of the useful function are still available to be used during memory corruption.

- Exports are now marked as invalid indirect call targets for CFG.
  - ➢ Removes wide set of useful valid icalls.
- All windows binaries are built with export suppression info.
  - ➢ A process needs to be explicitly opt in via the process mitigation option to enable export suppression.

Module exports are made invalid icall

Process Mitigation

RS2

Enabled for Edge

Makes Exploitation Harder

Mitigation – No Low IL/UNC Dll Loading

Low IL & UNC paths are controlled by attackers: These paths can be leveraged to feed in malicious dlls during other memory corruption.

- Blocks loading of binaries with Low Mandatory Label or from remote shares (UNC/Web).
- Enabled via the process mitigation option ProcessImageLoadPolicy.

Unsafe dirs are avoided for DLL load

Process Mitigation

TH2

Edge enables UNC blocking

Removes Exploitation Techniques

Mitigation – No Child Proc

Creation of child process to bypass mitigations: Attackers can trick a sandbox to create a child process.

- Blocks the spawning of a child process from a process.
  - ➢ No longer possible to bypass CFG by icall'ing WinExec & related APIs.
  - ➢ Prevents code execution via launching a child process.
- Property of the process token and thus inherited during impersonation.
- Enabled via UpdateProcThreadAttribute at process creation time.

Child process can't be created

Process Mitigation

RS1

Enabled for Edge/VMWP

Removes Exploitation Techniques

Mitigation – Code Integrity Guard (CIG)

Loading unsigned code into address space: Attackers can trick loading of unsigned binaries into the target.

- Prevents loading non-Microsoft signed binaries.
  - Microsoft, WHQL, Store or DRM signed.
- Prevents unwanted dll injection.
- Enabled via the process mitigation ProcessSignaturePolicy.

Non-signed code is not allowed

Process Mitigation

TH2

Enabled for Edge

Removes Exploitation Techniques

Mitigation – Arbitrary Code Guard (ACG)

Converting arbitrary memory into executable: Final step of exploits is to convert the pages as executable to execute shellcode.

- Blocks dynamically generation or modification of code in a process.
  - New executable code pages can’t be created.
  - No more injection of code into the process.
- Enabled via the ProcessDynamicCodePolicy.
  - Dynamic code restriction was supported sine Windows 8.1.
- Edge enables ACG in RS2.
  - All JIT is now done OOP.

Pages can’t be converted to exec

Process Mitigation

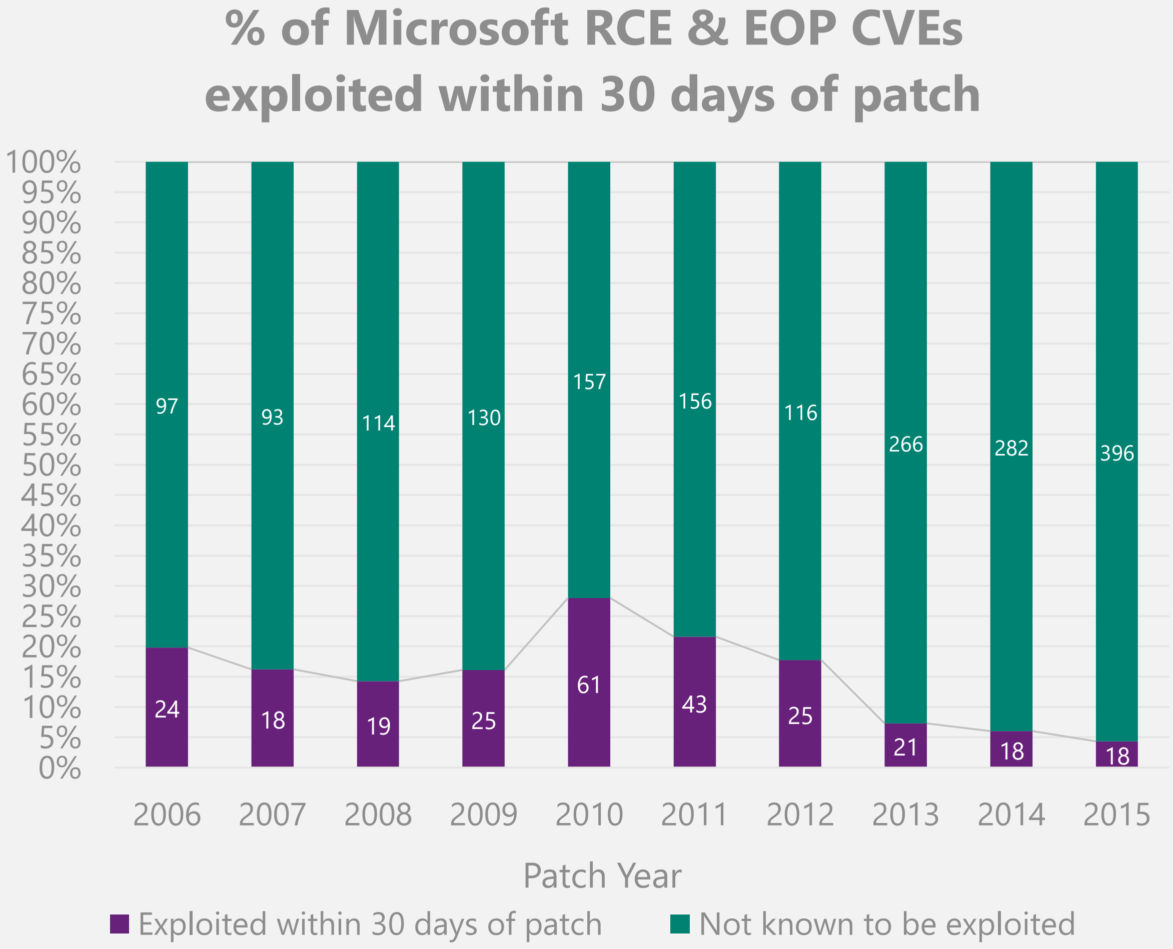
RS2

Enabled for Edge

Removes Exploitation Techniques

Closing note

- We are constantly researching and committed to add more safeguards to our platform/products and drive-up the exploitation cost.



Microsoft

Windows

Thank You!

@swamysng

Microsoft