Windows

The State of Kernel-Mode RCE Defense

Joe Bialek (@JosephBialek) – MSRC Vulnerabilities & Mitigations Team

Microsoft

Abstract

While user-mode is often the primary target for remote-code-execute exploits, the Eternal* exploits have demonstrated that the kernel should not be forgotten. In this talk, we'll discuss the strategy Microsoft has been using to mitigate software vulnerabilities and the unique challenges posed by protecting the kernel. We'll look at public exploits for SMB vulnerabilities to help illustrate fundamental limitations of protecting a privileged domain but also to highlight that absolute security isn't necessary to obtain some level of protection. Finally, we'll discuss the path forward for kernel security.

Agenda

Overview of Microsoft's exploit mitigation strategy

Eternal* exploits case study

Takeaways

How we move forward protecting our kernel

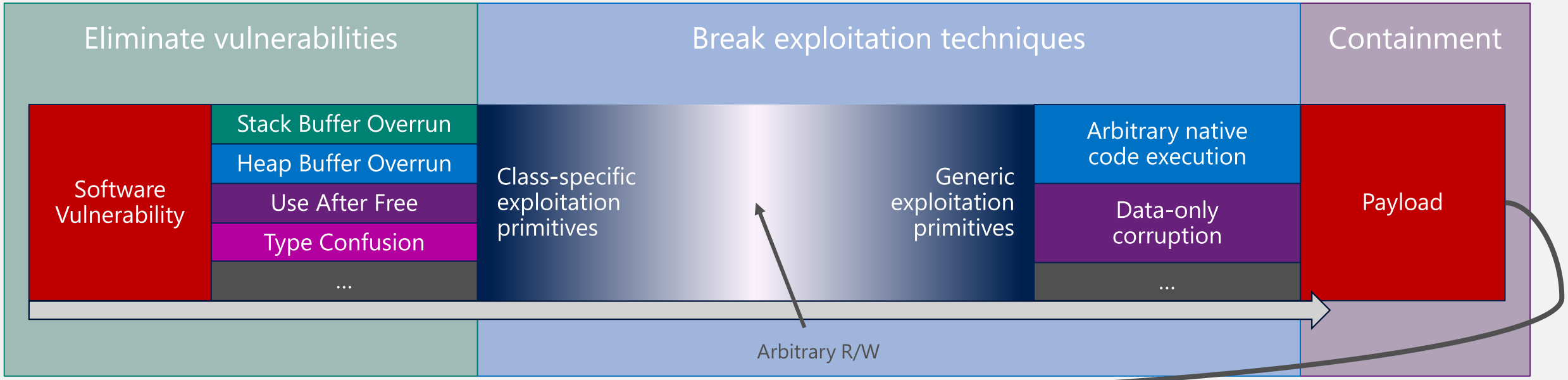
Microsoft's strategy for exploit mitigations

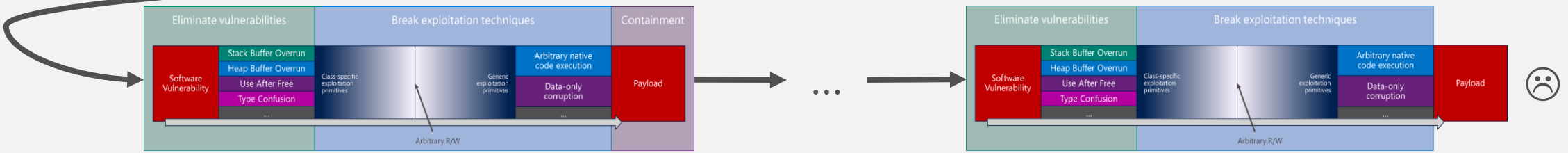
Layered, data-driven software defense in Windows 10

Our Strategy	Make it difficult & costly to find, exploit, and leverage software vulnerabilities
Our Tactics	Eliminate entire classes of vulnerabilities
	Break exploitation techniques
	Contain damage & prevent persistence
	Limit the window of opportunity to exploit

How we think about mitigating user-mode software vulnerabilities

Attackers transform software vulnerabilities into tools for delivering a payload to a target device





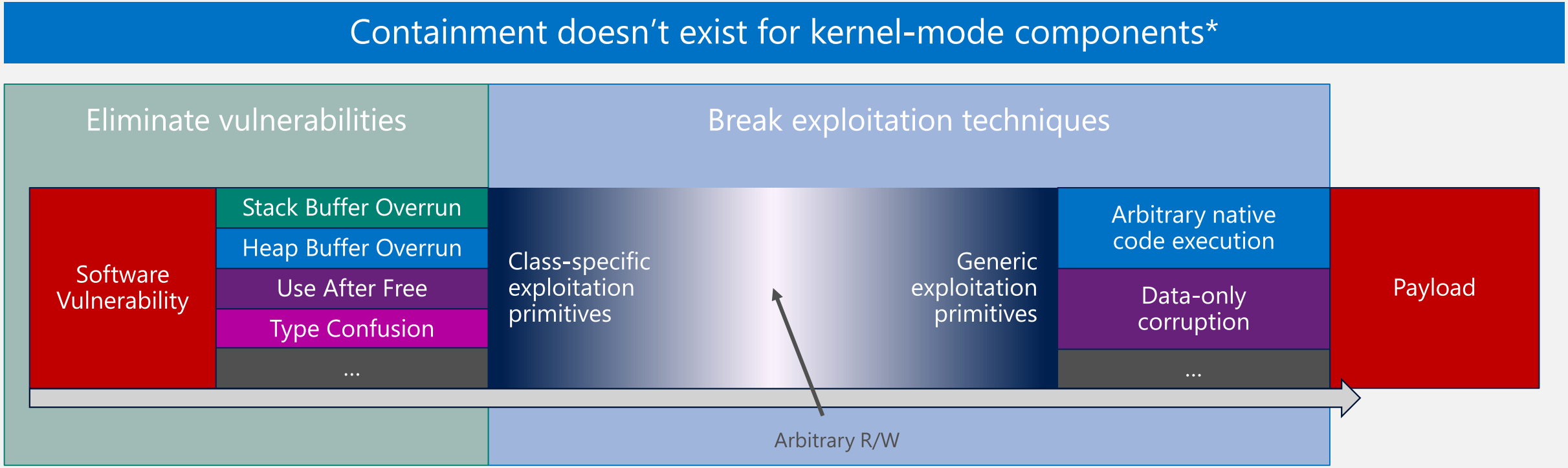
Attackers typically need to elevate privileges

At some point, we lose containment as a defense

This means applying the same defenses to privileged attack surfaces

This leaves eliminating vulnerabilities & breaking techniques

How we think about mitigating kernel software vulnerabilities



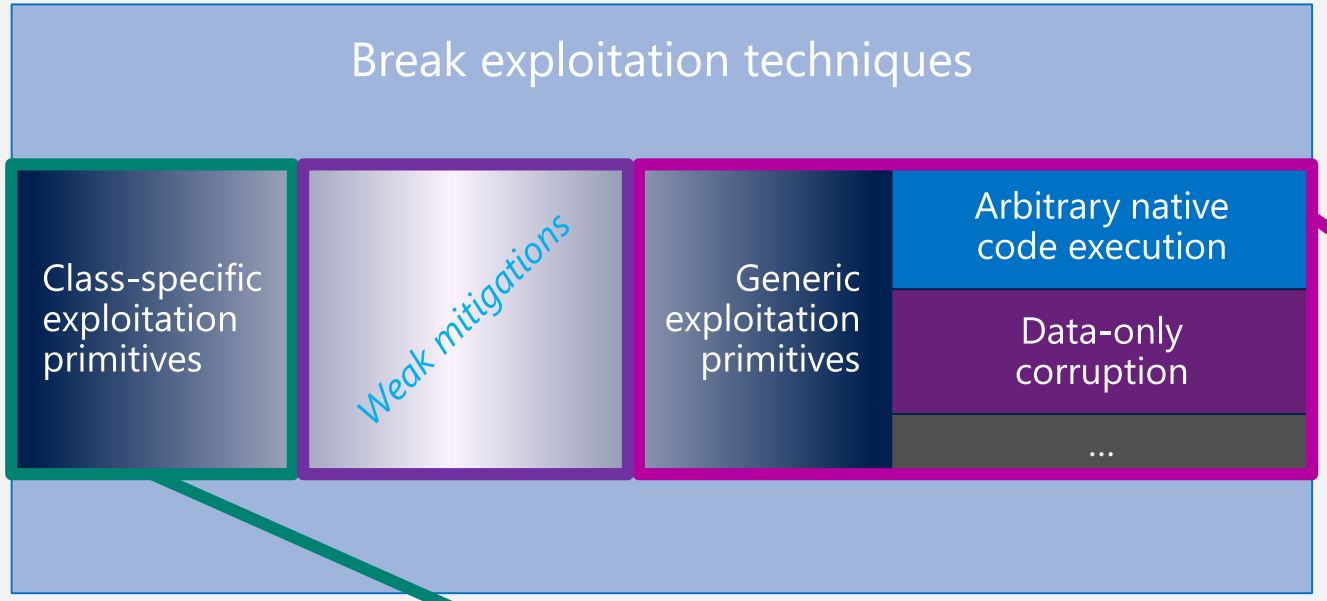
Two options for securing kernel code

1. Eliminate vulnerabilities
    1. Finding/fixing bugs
    2. Moving code to user-mode
    3. Deleting code
2. Break exploitation techniques

* WDAG allows apps to be run in a Hyper-V isolated container. This isolates the kernel used by those apps.

Chokepoints for breaking exploitation techniques

All exploits rely on combining various primitives to enable the delivery of a payload



Chokepoint #1

Break class-specific techniques for transforming a vulnerability into generic primitives.

Goal: make it difficult or impossible to exploit certain types of vulnerabilities.

Chokepoint #2

Break generic primitives with the assumption that an attacker has arbitrary R/W.

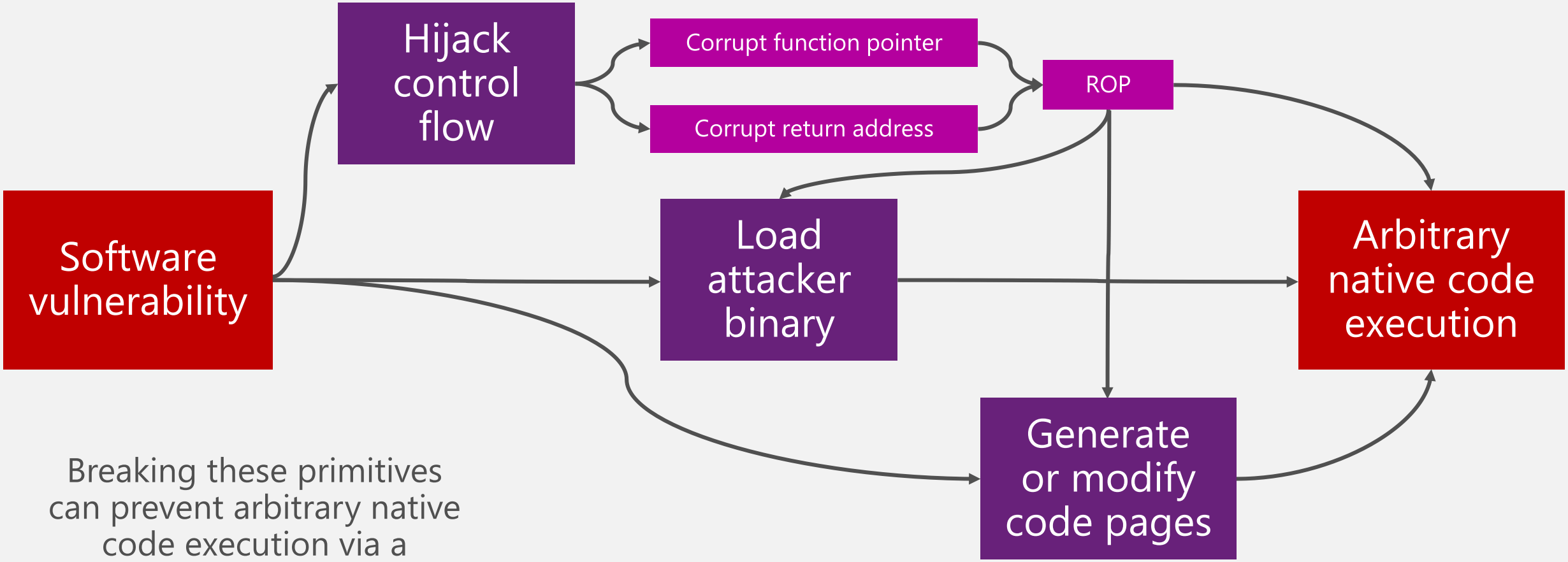
Goal: make it difficult or impossible to deliver a payload independent of the type of vulnerability.

Class specific exploitation primitives are easier to robustly mitigate but less comprehensive

Generic primitives are extremely difficult to robustly mitigate but are in-theory completely comprehensive

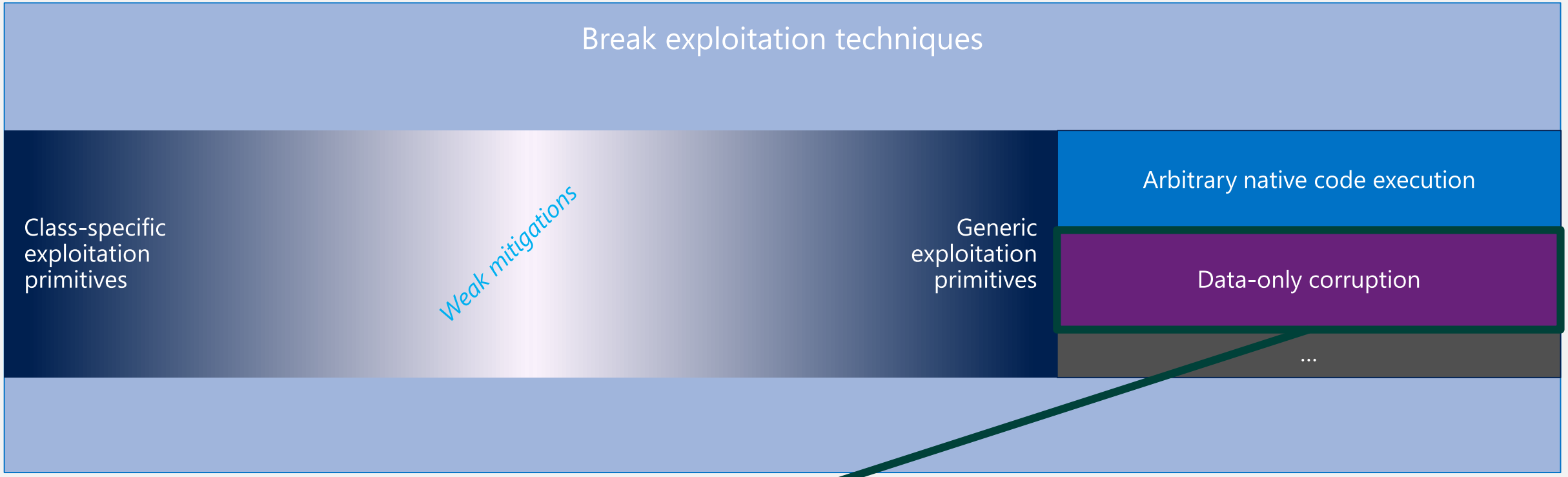
The paths to arbitrary native code execution

There are a finite number of ways to transform a vulnerability into arbitrary native code execution



Breaking these primitives can prevent arbitrary native code execution via a vulnerability

What about data corruption



Data-only corruption derails the expected state machine of a program

Must assume an attacker can perform any actions the programs permissions allow for, without obtaining native code execution. E.g., token swapping.

Kernel mitigations

Kernel mitigations

Class specific	"In between region"	Arbitrary native code
Stack canaries	ASLR	Kernel CFG
Win32k isolated pool	Misc. header hardening	HVCI
Win32k UAF protections	SMEP	NX Kernel Regions
NULL deref protection	Patch protection	
SystemBuffer init	Isolated stack region	

Not all investments are made specifically to mitigate exploits, e.g., PatchGuard, but may still help

Challenges with kernel-mode engineering

- 3rd party code “in-proc” that cannot be removed or special cased
  - Breaking compat == system may not boot, fail to upgrade, etc.
  - Widely used drivers take dependencies on undocumented behavior
- Code with wildly different requirements in the same address space
  - Some code extremely performance sensitive, different bottlenecks (I/O, CPU, avoid context switches, etc.)
  - Some code in kernel just to keep secrets
  - Some code has no reason to be there but is too expensive to move
  - The requirements of all code my be catered to
- Kernel specific considerations like IRQL, non pageable resources, etc.

Mitigations - KASLR

All kernel regions except UserSharedData have ASLR applied as of RS2

Top level page table has 512 entries: 256 user, 256 kernel

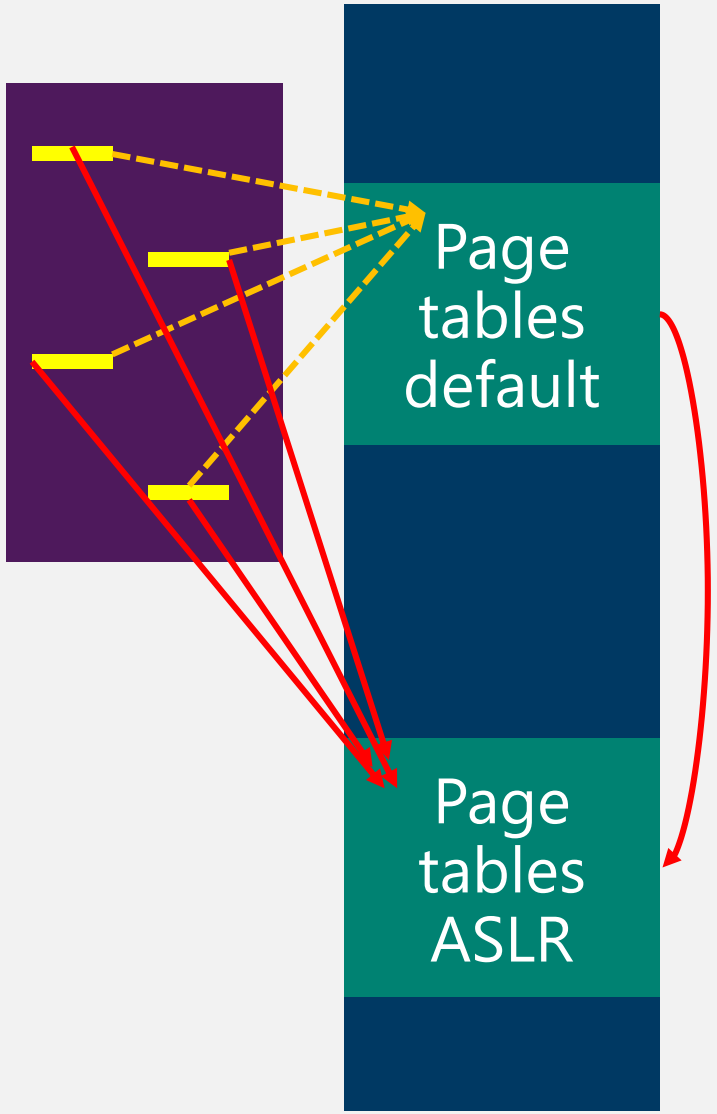
Kernel regions allocated in 512 GB chunks (1 PXE) during early boot

Each region receives 8 bits of entropy

Some regions sub-randomize for additional entropy

“Dynamic fixup” mechanism used to preserve performance

Mitigations – KASLR Dynamic Fixups



- Region address defined as a symbolic constant
- Compiler metadata stores where symbol used
- NT memory manager chooses random location for region
- NT performs relocations on symbolic constants
- Regions can be accessed without global indirection

Eternal* exploit case study

Background

- A handful of exploits targeting fixed vulnerabilities
- 4 targeting recently patched SMB vulnerabilities
  - EternalBlue, EternalChampion, EternalRomance, EternalSynergy
  - Exploits as-written affected up to Windows 7 X64
- Leaked exploits repackaged and turned in to WannaCry, Petya
- Latest versions of Windows unaffected by public exploits for 3 months after patches released

Timeline

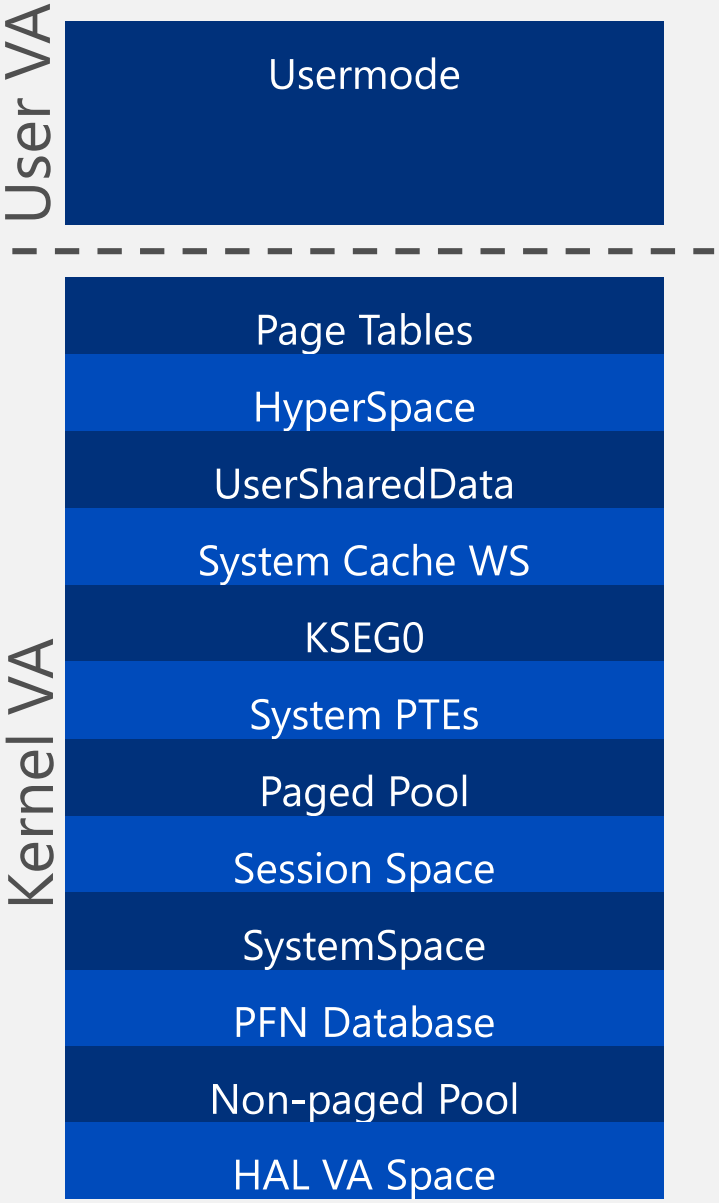
- March 14th – Microsoft releases patches for SMB vulnerabilities
- April 14th – ShadowBrokers release dump containing exploits
  - 4 SMB exploits for bugs fixed April 10th
  - 3 exploits support up Win7 X64, 1 supports up to Win8 X64
- May 12th – WannaCry worm goes live
  - Over 200,000 victims and 300,000 computers infected according to Wikipedia (must be true)
- May 14th – EternalBlue ported up to Windows 8.1/2012R2*
- June 8th – EternalBlue ported up to Windows 10 TH2**
  - Exploit wasn't published, just report
- June 19th – EternalRomance ported up to Windows 10 RS2***

* https://gist.github.com/worawit/074a27e90a3686506fc586249934a30e/revisions

** Report no longer available, was published by RiskSense: https://www.risksense.com/_api/filesystem/466/EternalBlue_RiskSense-Exploit-Analysis-and-Port-to-Microsoft-Windows-10_v1_2.pdf

***https://github.com/worawit/MS17-010/blob/master/zzz_exploit.py

System VA Space Overview (Windows 7 X64)



All kernel regions at fixed virtual addresses

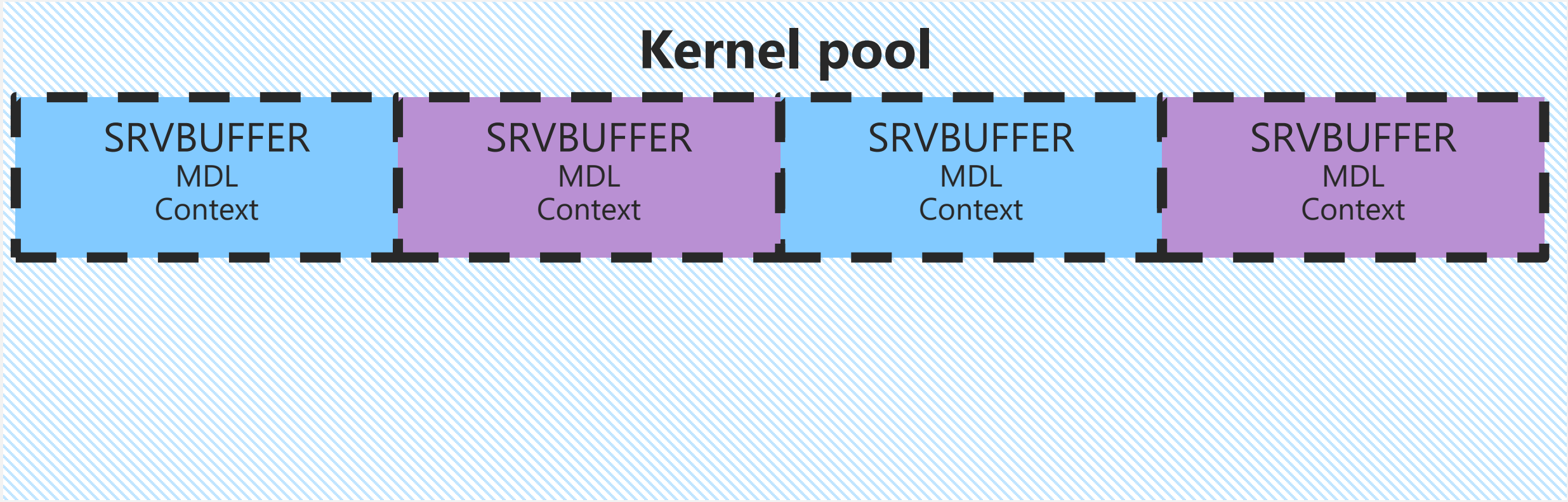
Some regions sub-allocations randomized

Most regions read/write/execute

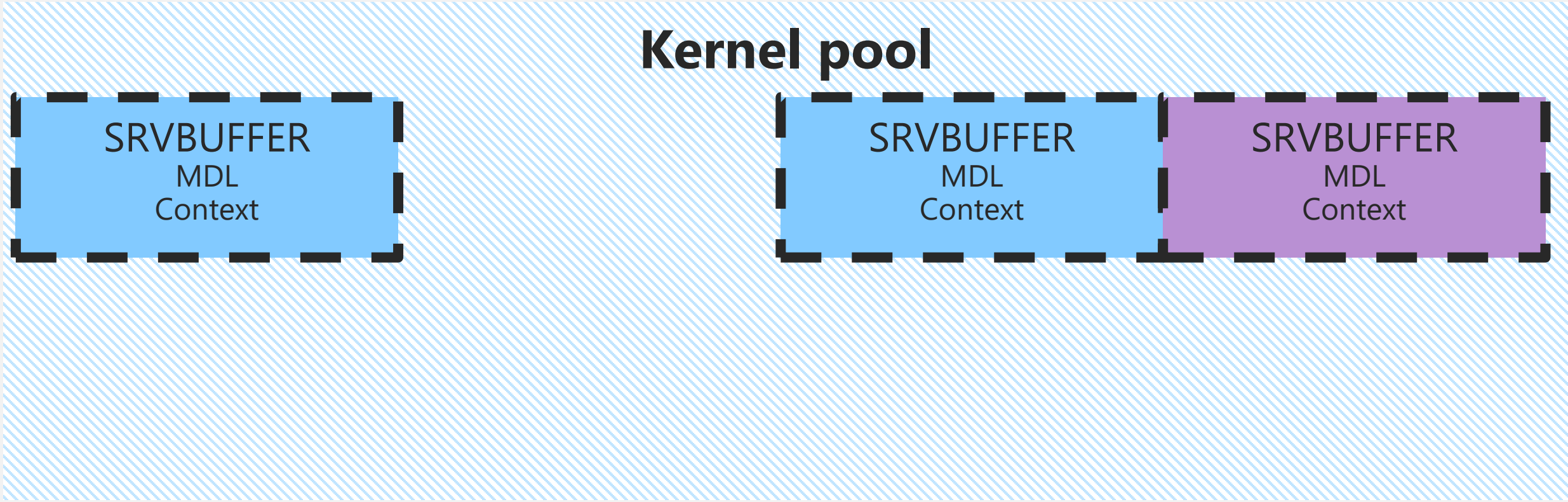
Not all pages in a region necessarily mapped

EternalBlue

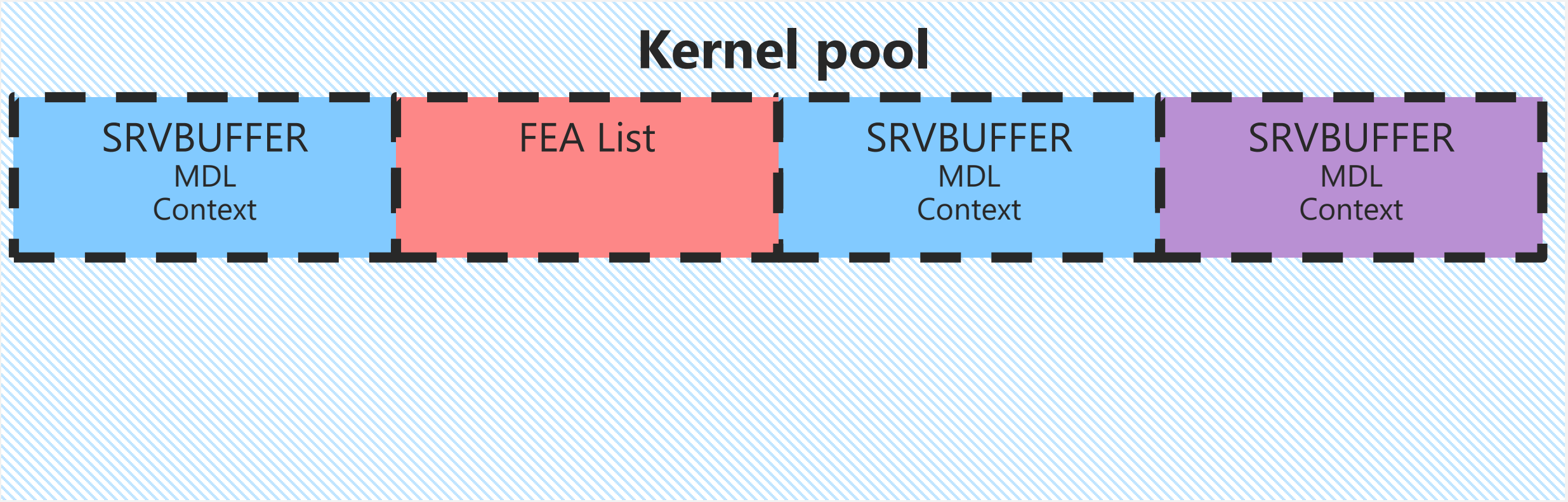
- Pre-authentication (Windows 7 and below)
- Linear pool overflow, attacker controlled size & content
- Reliability issues due to the pool massaging required



Attacker issues numerous SMB2 requests to spray the non-paged pool with SRVBUFFER structures



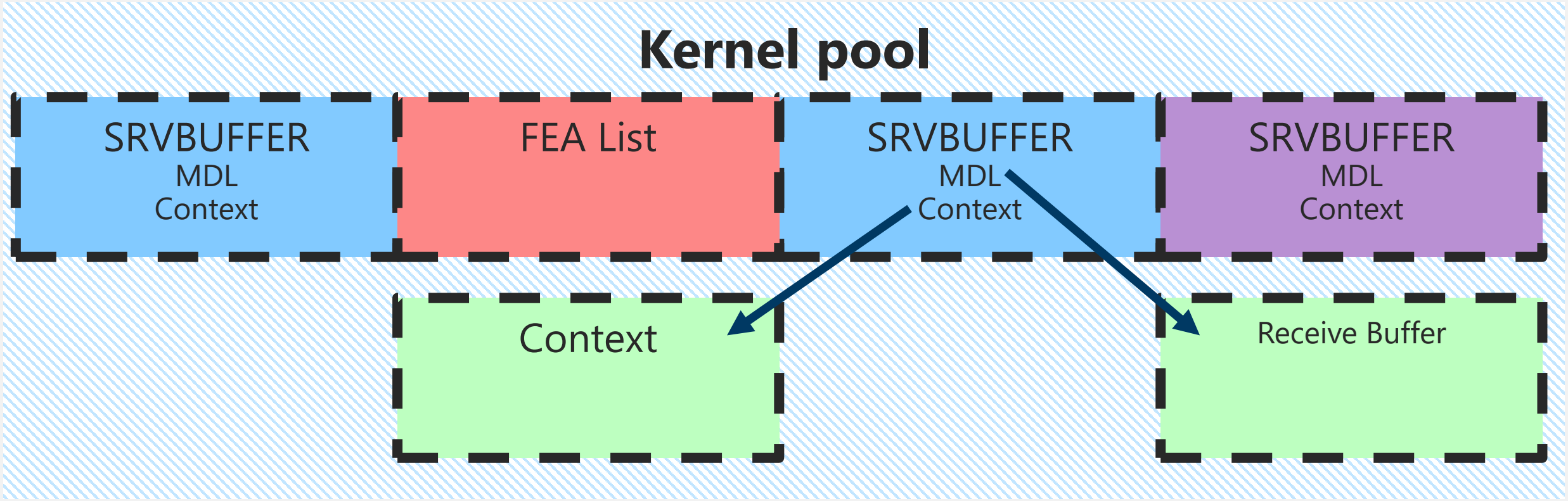
Attacker triggers a free of one of these structures (kill a pending request)



Attacker issues an SMB1 request.

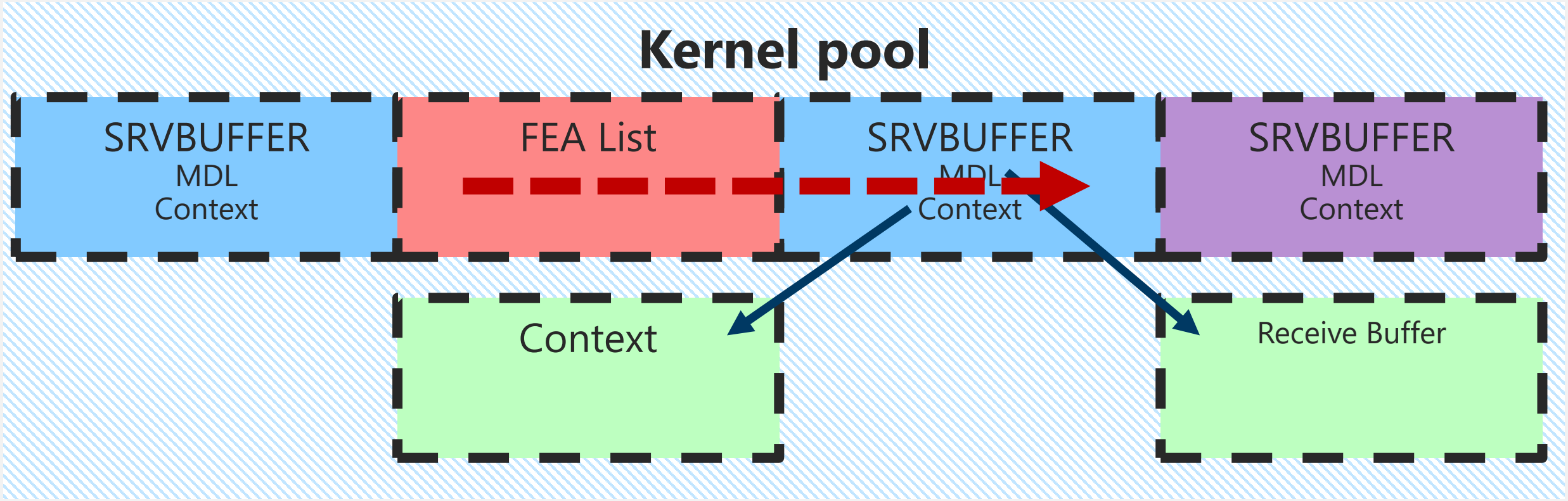
An FEA List buffer is allocated. It fills the pool bucket previous taken by an SRVBUFFER.

The FEA List buffer is allocated too small due to a integer truncation bug.



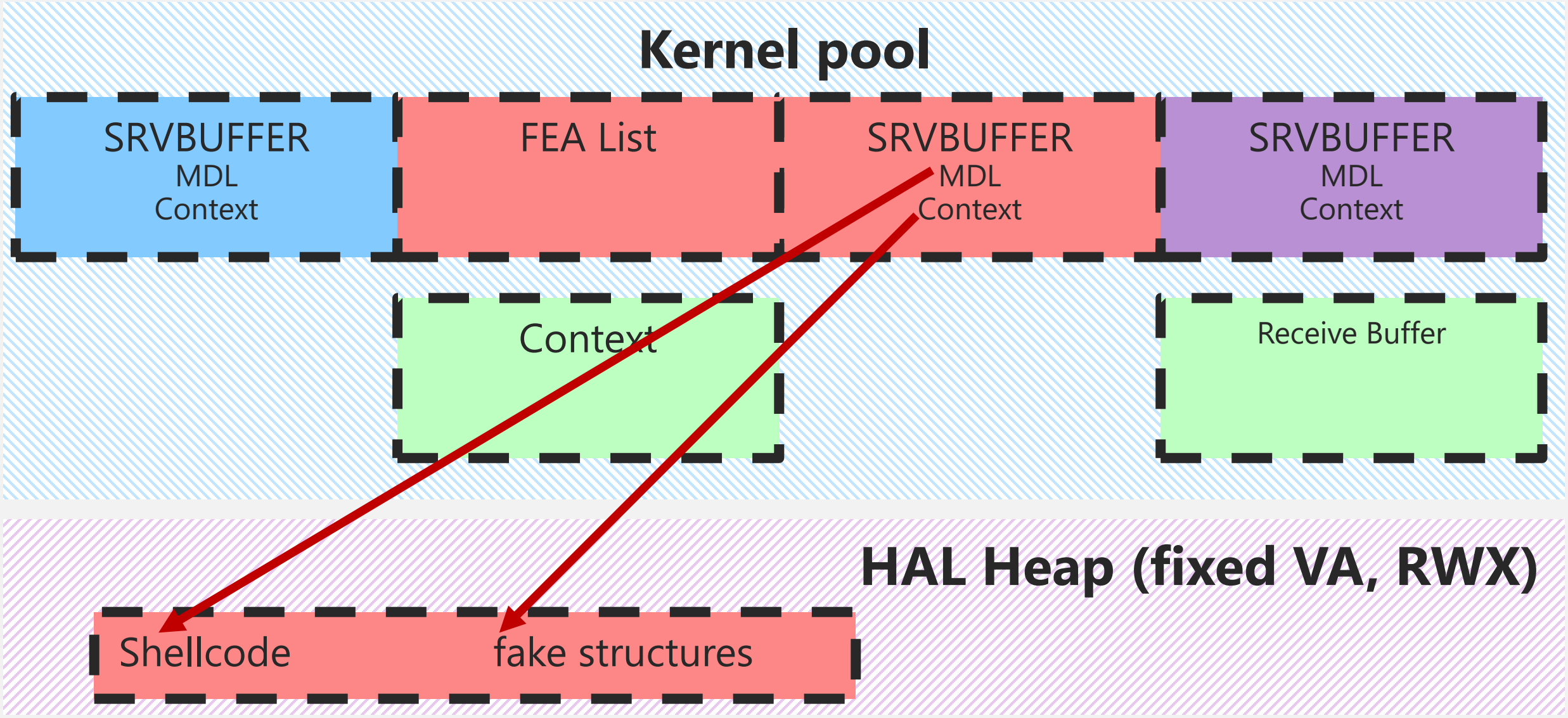
A legitimate SRVBUFFER contains (among other things):

1. An MDL mapping a “receive buffer”
2. A context pointer (contains pointers to other structs that have function pointers)

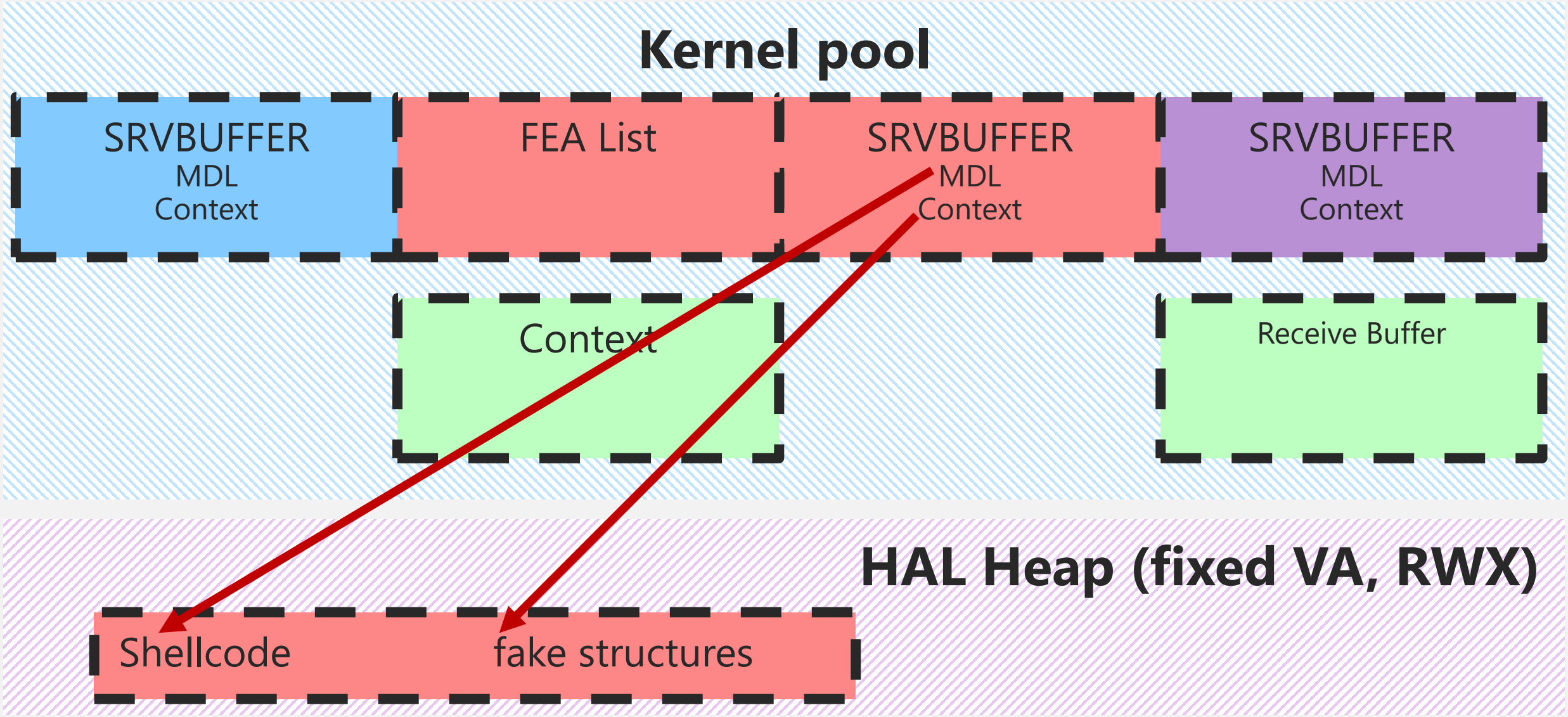


The attacker overflows the FEA List (since it is too small) and corrupts the adjacent SRVBUFFER

The attacker can set the MDL and Context pointer to arbitrary values



The attacker makes the receive buffer MDL and context point in to the HAL heap



Data the attacker sends to the SMB2 session gets written to the receive buffer (HAL Heap)

The attacker writes shellcode and a context structure whose function pointers point to the shellcode

Mitigations

- Exploit breaks on Windows 8 due to widespread NX improvements
  - Non-paged pool
  - HAL heap
  - PFN database
  - Page tables
  - System cache
  - Etc.

- Exploit breaks on Windows 10 due to kASLR

EternalBlue – public evolution

- 1 month later: Windows 8.1/Server 2012r2 support added
- New technique: Trigger the vulnerability twice
  - First write, modify the page tables
  - Second write, same technique as the original exploit

Mitigations

- Kernel ASLR (1607 / RS1)
  - 8 bits of entropy minimum for all kernel regions including the page tables
  - HAL heap received randomization in RS2
- Hypervisor Code Integrity (RS1)
  - Pages are not marked executable in the SLAT unless SecureKernel validates their page hash
  - Only images may be marked executable, no executable pools, etc.

EternalRomance

- Similar vulnerability primitive to EternalBlue, linear pool overflow
- Seemingly more stable than EternalBlue (due to allocation size?)
  - Also requires authentication, but so does EternalBlue on Windows 8 and above
  - Better candidate for exploitation on Windows 8 and above
- June 19th – Public exploit released for Windows RS2/Server 2016
  - Instead of corrupting function pointers and HAL heap, corrupt the user context of the SMB session
  - All file accesses now happen using the SYSTEM account

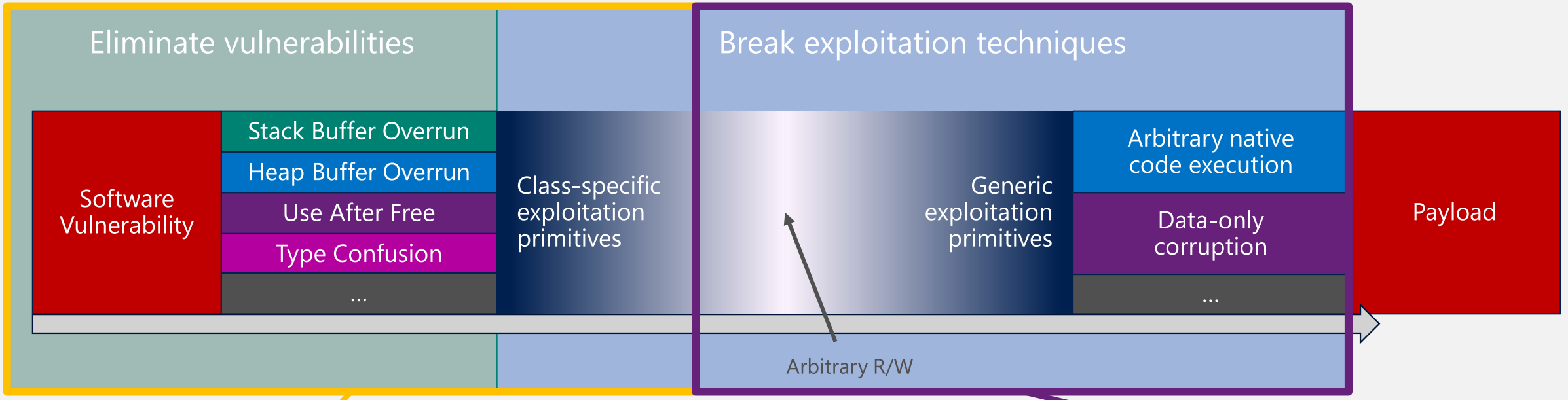
Takeaways

Positive takeaways

- Exploit mitigations bought customers additional time to patch
  - ~2 month after patches released for Windows 8.1/Server 2012r2
  - ~3 months after patches released for Windows 10 RS2 / Server 2016
- Mitigations pushed exploit into a component-specific attack
  - Technique will not work as-implemented if SMB is inaccessible
  - Exploit now involves writing a file that will be executed by something, detection opportunity
- Not all defense scenarios require perfect mitigations

Moving forward

Strategies for kernel protection



Robust mitigations possible

Kill vulnerabilities at their source

Kill class-specific techniques

Eliminate attack surface

Defense-in-depth mitigations

Not 100% robust

Increase exploit complexity & provide other benefits such as mitigating rootkits

The closer a mitigation is to the bug, the more confident we can be about the robustness

Microsoft's future kernel plans

- Continue to invest in exploit mitigations
  - Prefer mitigations that hard kill bugs or severely impact reliability
  - Push to enable security features by default, e.g. HVCI, kCFG
- Where applicable, utilize Hyper-V isolation (e.g. WDAG for Edge)
- Increase proactive bug finding efforts internally
  - E.g. Microsoft has recently found numerous SMB and Hyper-V bugs and fixed them on all supported platforms
- Attack surface reduction, e.g. disable SMB1 by default
- Disrupt attacks as much as possible with our 2x/yr release cadence

New Microsoft Bounty Programs

- Windows Bounty Program includes all critical and important bugs in
  - **Windows Insider Preview**
  - Hyper-V
  - Microsoft Edge
  - Windows Defender Application Guard

Follow us on the MSRC Blogs to get information on new bounties

https://blogs.technet.microsoft.com/msrc/

Windows Bounty Program Targets

- Submit:
  - Critical and important vulnerabilities in Windows Insider Preview slow
  - Hyper-V escapes, Information disclosure and DOS bugs in Hyper-V
- This continues our effort in finding bugs in various stages of development

Category	Targets	Windows Version	Payout range (USD)
Base	NEW Windows Insider Preview	WIP slow	$500 to $15,000
Focus area	NEW Microsoft Hyper-V	Windows 10 Windows Server 2012 Windows Server 2012 R2 Windows Server Insider Preview	$5,000 to $250,000
Focus area	NEW Windows Defender Application Guard	WIP slow	$500 to $50,000
Focus area	Microsoft Edge	WIP slow	$500 to $15,000
Focus area	Mitigation bypass and Bounty for defense	Windows 10	$500 to $200,000

Windows Insider Bounty Program

Submit high quality critical and important vulnerabilities in Windows Insider Preview slow

Vulnerability Type in Windows Insider Preview Slow	Whitepaper / Report Quality/ Proof of Concept	Pay-out Range(USD)
Remote Code Execution	High	Up to $15,000
	Low	Up to $1,500
Elevation of Privilege	High	Up to $10,000
	Low	Up to $5,000
Information Disclosure	High	Up to $5,000
	Low	Up to $2,500
Remote Denial of Service	High	Up to $5,000
	Low	Up to $2,500
Tampering / Spoofing	High	Up to $5,000
	Low	Up to $2,500

Hyper-V

- Hyper-V escapes that will receive a bounty
  - Guest-to-Host
  - Guest-to-Guest
  - Guest-to-Host DoS (non-distributed, from a single guest)
- Total payout range is: Up to $250,000 USD

- RCE $5,000 to $250,000
- Info Disclosure $5,000 to $25,000
- DOS $5,000 to $15,000

Window Defender Application Guard

An eligible submission includes a RCE vulnerability in WDAG that enables a guest in the WDAG container to compromise the hypervisor, and escape to the host. Also included are vulnerabilities inside the container that do not lead to an escape to the host.

Vulnerability Type	Proof of concept	Functioning Exploit	Report Quality	Payout range (USD)
Vulnerability resulting in escape from the WDAG container to the host	Required	Yes	High	$30,000
		No	High	$20,000
		No	Low	$10,000
Vulnerability within the Application Guard container, no container escape	Required	No	High	$10,000
		No	Low	$2,000

Take Action

1. Visit https://aka.ms/BugBounty for a current list of active bounties
2. Identify the bounty you want to go after and start hacking away at it
3. Report your findings to secure@microsoft.com
   - Describe the bug and how you exploit it
   - Provide a Proof of Concept (PoC)
   - For complicated bugs (software) provide a white paper or detailed write up
     - If it's a high quality report, you get larger bounties
     - If it has greater impact to Microsoft, you get larger bounties
4. Give us your name and a good email to reach you at
5. Encrypt with our public key (if it's a PoC or working exploit)
6. For eligible bounty cases, GET PAID!

Appendix

Mitigations – Hypervisor Code Integrity

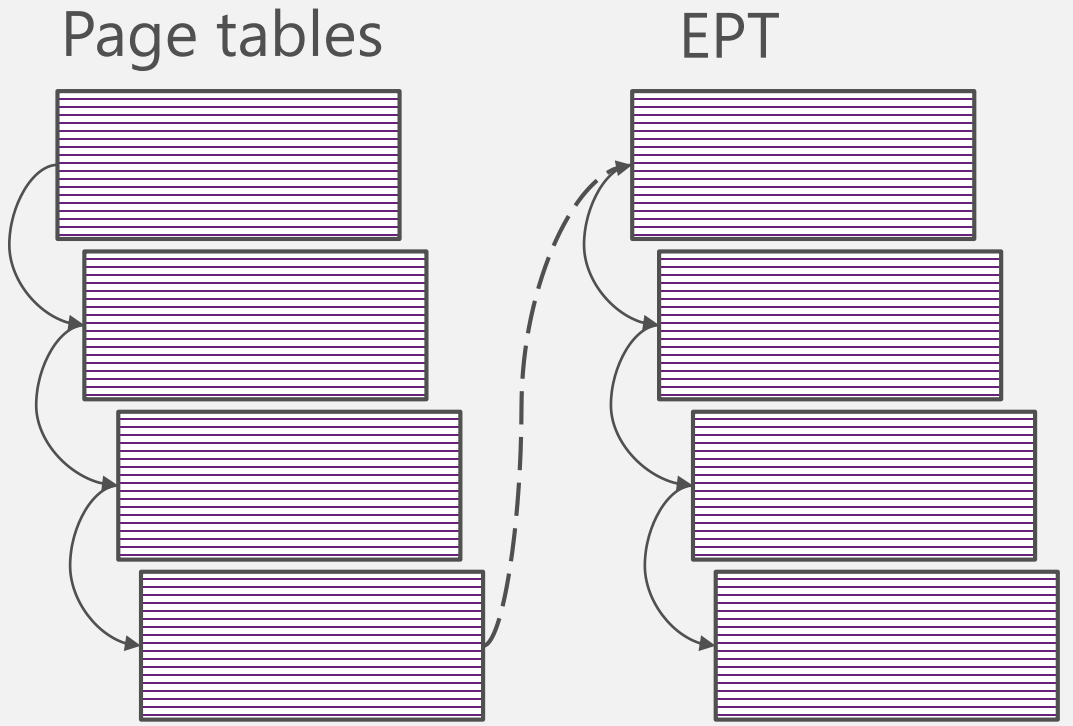
EPT – Extended Page Tables

2nd level of page tables managed by the hypervisor and SecureKernel

Map “guest physical” to “system physical” addresses

SecureKernel only marks “executable” on EPT entries if it validates the signature

Only applies to kernel mode pages, user mode can run unsigned code (e.g., JIT)



Mitigations – Hypervisor Code Integrity

Pre-KabyLake: Hyper-V maintains EPT's for user-mode and kernel-mode

Kernel-mode EPT marks validly signed pages executable

User-mode EPT allows all pages to be executable

Hyper-V catches all attempts to context switch from user->kernel & switches EPT

If EPT #PF and CPL==3, guest transitioned to UserMode, load UserMode EPT

KabyLake: EPT contains separate user/kernel execute bits, separate EPT's not needed