Security Research over Windows

[kernel]

Draft v.1.1

$whoami

- @zer0mem ~ Peter Hlavaty
- Senior Security Researcher at KeenLab, Tencent

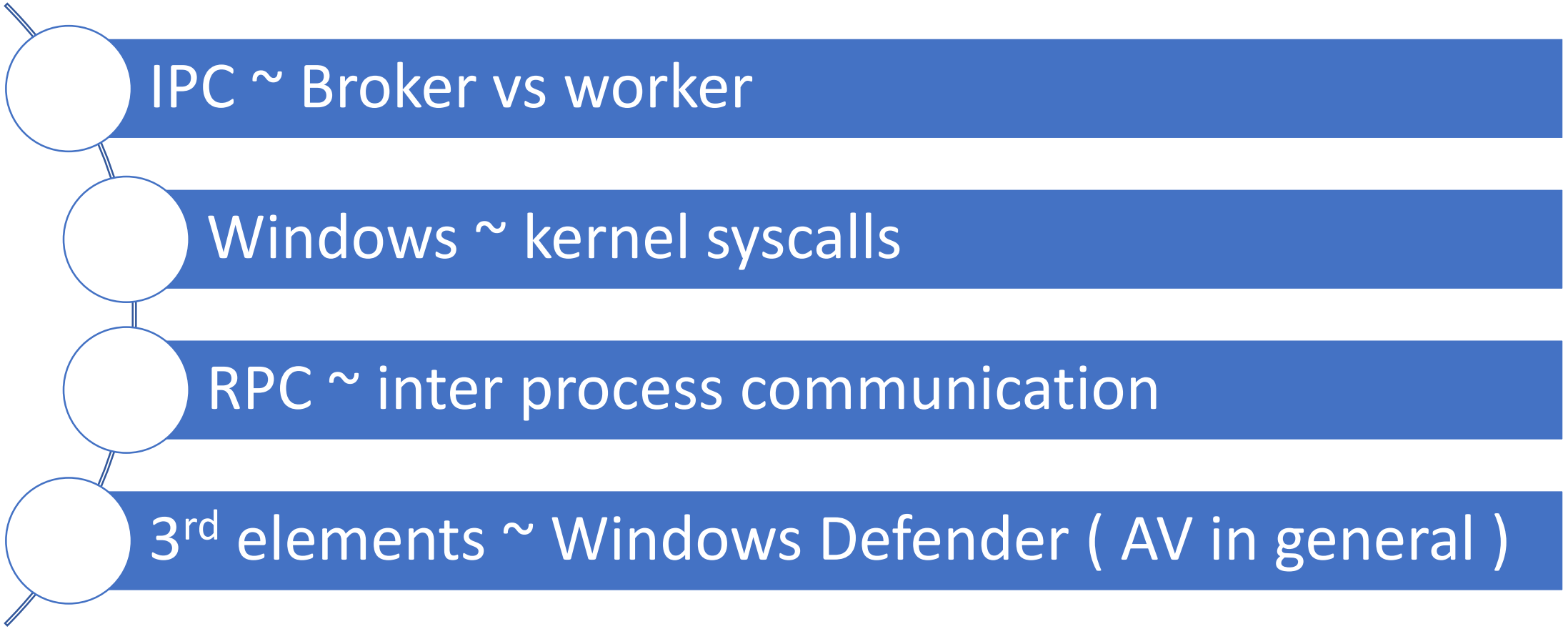
- MSRC100, pwn2own
- Focus : kernel / hyperv / mitigations

- sometimes talk somewhere ..
- wushu player +- ☺

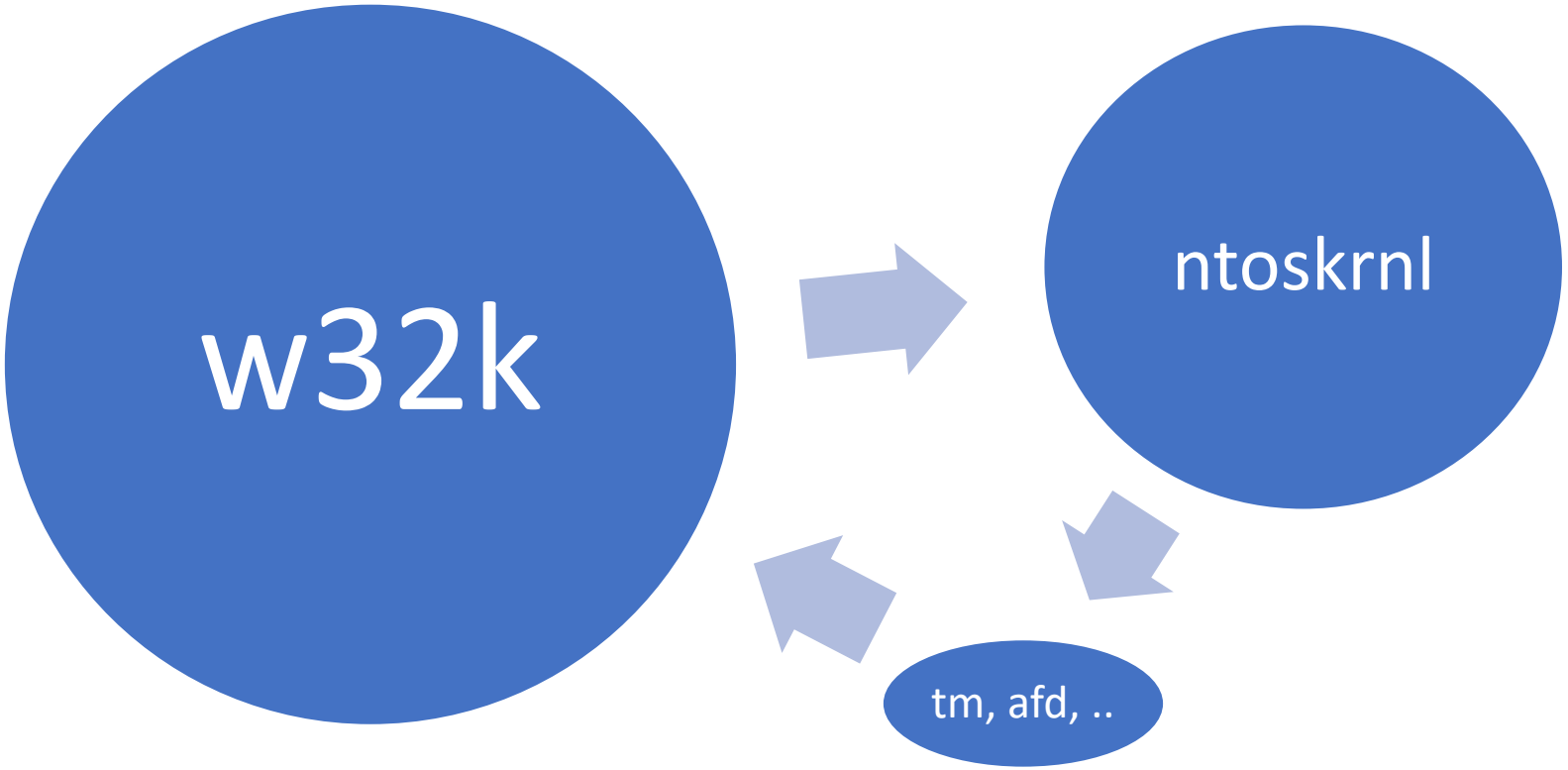
Sandbox

- Restrict resources of target (process)
  - #syscalls
  - file system
  - registry
  - inter-process interaction
- Different integrity levels
  - Untrusted
  - App Container
  - Low
  - Medium, ..

sandbox attack surface +-



Windows kernel ~ attack surface



w32k attack surface hardening

4 years ago

- **Fonts**
  - TTF **emulation** in kernel
  - Loading custom fonts
- **GDI**
  - 6+ different kernel objects
    - ***huge*** source of UAF, overflows, …
  - **EMF – *remote***
- User
  - **User mode callbacks machinery**

now

- ~~**Fonts**~~
  - TTF emulation in ~~kernel~~ user mode
  - **Sandboxes low priv proc** for custom
- ~~GDI~~ -> restricted/**no** mode
  - 6+ different kernel objects
    - *huge* source of UAF, overflows, …
  - ~~EMF – *remote*~~ -> disabled by def
- User -> restricted/**no** mode
  - **User mode callbacks machinery**

w32k exploitation hardening

- Restricted resources for exploitation
  - No resources if DisableW32kSystemCalls flag on ☺
- Type isolation
- Tactical mitigations, f.e. tagWnd
- bugs--
  - Refactored w32k (win32k -> win32kfull + win32kbase)
    - this also left/brings lot of bugs, but showing importance of cleaning up mess
  - Security researchers community support (msrc100, insider bounties, ..)
  - Internal fuzzing++ ?

w32k still alive

- DirectX
- w32k – user callbacks
- Small parts of GDI + DComposition
- New syscalls keep added in new builds

- ~ no w32k in your target ?
  - w32k is somehow essential of GUI app
  - Bridge from your target to part of app which have access
  - Perhaps you can attack another part of app with w32k on ?

ntos attack surface

- TM + CLFS
  - ‘hidden syscalls’
  - CLFS : Lockdown for sandboxed processes!
    - Well finally, heavy parsing in kernel mode..
  - Without CLFS backup it is very simple logic
    - However nice connections ~ Manager + Transaction + Enlistment + Resource
- (A)LPC, Pipe, Sockets, Registry hives
  - Good amount of logic there
  - In SDL quite some time, crucial part of windows kernel!
- Memory management, Sync, ..
  - + : lots of syscalls!
  - - : logic you can alter is way too simplistic

RPC – user processes

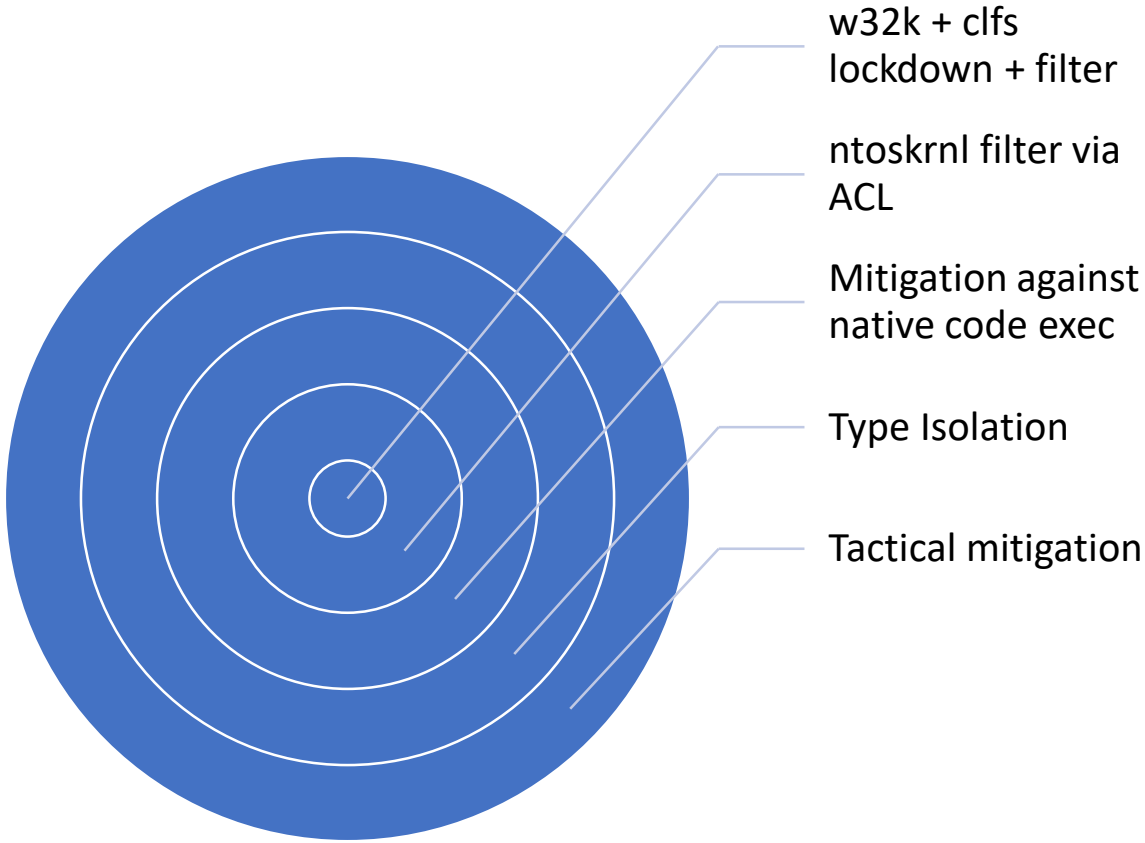
- Any process has opened ALPC port
  - Everybody needs to have opened port at least to csrss.exe !
- Mostly ‘unknown’ area ~ previously little researched
  - https://hakril.net/slides/A_view_into_ALPC_RPC_pacsec_2017.pdf
  - NtAlpc* ~ undocumented
- COM using ALPC at the background
  - C++ inter-process interface

Native code exec

But, OK .. you got a bug, what’s next ?

Mitigations on the rise

- Past years Windows invest heavily into breaking attack surface and techniques !
- Guards :
  - (k)CFG
  - HVCI
  - VBS
  - ACG
  - CIG
  - Jit OoP
  - ..

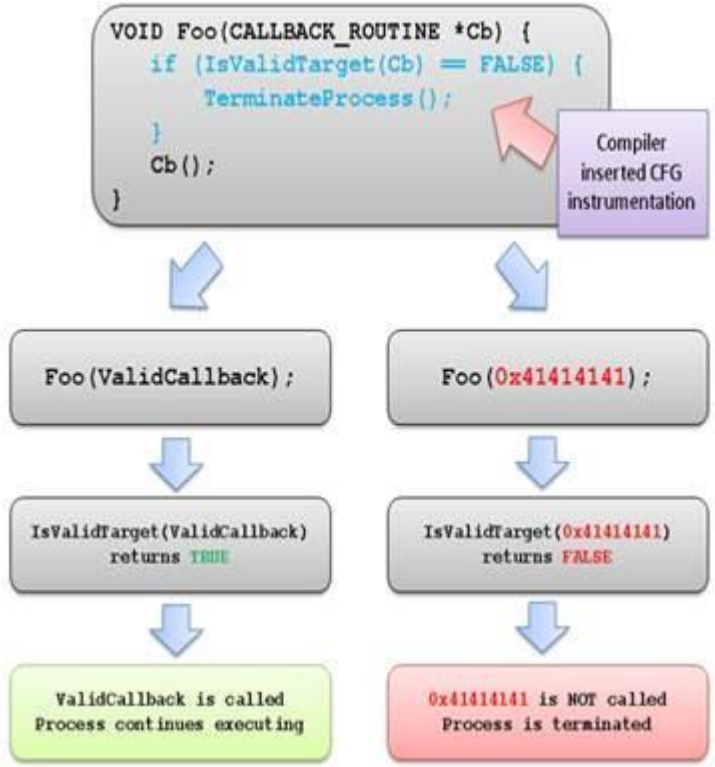


..G, ..G, ..G .. wut ?

- Lots of guards in windows ;)
- Must read :
  - https://cansecwest.com/slides/2017/CSW2017_Weston-Miller_Mitigating_Native_Remote_Code_Execution.pdf
  - https://github.com/Microsoft/MSRC-Security-Research/blob/master/presentations/2018_02_OffensiveCon/The%20Evolution%20of%20CFI%20Attacks%20and%20Defenses.pdf
- How those are enabled for sandboxing :
  - SetProcessMitigationPolicy
  - PROC_THREAD_ATTRIBUTE_MITIGATION_POLICY of UpdateProcThreadAttribute

CFG

- _guard_check_icall



- More about CFG :
  - https://blog.trendmicro.com/trendlabs-security-intelligence/exploring-control-flow-guard-in-windows-10/
- This is nice article :
  - https://www.endgame.com/blog/technical-blog/disarming-control-flow-guard-using-advanced-code-reuse-attacks
  - Covering also mini COOP ;) – check that out

RFG ~ pulled down, but CET (check CET!)

- 2 stacks : Control + Data
- Control stack no pointer in user mode
  - It is OK to be write-able ~ therefore with write primitive you can write there
  - But problem : how to find it ? => no leaks == no way ?
- At each function prolog store return address also to Control stack
- At each function epilogue check if ControlStack[rsp]==DataStack[rsp]
  - Aka return address match

- BRILIANT IDEA + DESIGN = no compatibility issues, can plug it right now!
  - Only 5 instruction per function!

- Key Problems :
  - Race condition -> could be done in stable way
  - Secret based ~ what if is possible to reveal address of control stack without pointer leak ?

RFG ~ explain in two slides with graphs

CIG + ACG + Jit OoP : In short

- Code Integrity (CIG) ~ only signed images can be loaded
  - Ok but we can do RWX + shellcode ☺
- Arbitrary code guard (ACG) -> no you can not ..
  - No RWX page same time!
  - X pages -> in fact you can not VirtualProtect to Exec* anymore
- JIT : but I need it!
  - Nope … nope .. nope
  - Process can not have RWX pages nor from Data page make Code page
  - Therefore only different process can do it for you
  - Browser : Jit Process -> Worker process

Type Isolation

- Important exploit primitives consists :
  - Structure with control and data parts
    - Control : pointers, sizes
    - Data : controlled data by user
- Outcome :
  - Data or size overflow lead to full compromise of domain
- Mitigation :
  - Separate Control & Data part of structure to two different places
  - Crucial : data should not reach control part ~ page guards / different pools

Tactical mitigation

- prevalent methodology of misusing object for arbitrary read / write
  - Start with limited read/write
  - Boost it to full read/write to domain
  - Usually pivot-worker schema

- Tactical mitigation == Break particular techniques, one by one!
- How : Introduce safe – checks
  - Buffer ranges
  - Pool limitation
- Outcome : need to chain *limited* read/write primitives

- Crucial :
  - safe boundaries must not be reachable by our limited write
  - broken for tagWnd ~ check this nice references :

https://github.com/MortenSchenk/tagWnd-Hardening-Bypass/blob/master/tagWnd/tagWnd/tagWnd.cpp
https://improsec.com/blog/hardening-windows-10-with-zero-day-exploit-mitigations-under-the-microscope

Therefore..

- *Theory*
  - No W^X memory anymore
  - No Arbitrary modules
  - No @rip hijack
  - No return address hijack
  - No Overflows (buffer or size/counters) exploitable
  - No/Limited Read/Write primitive
- **Practice**
  - Not there yet, most of those bypassable by design limitations
  - However showing interesting shift towards security, does not it ?
    - especially memory corruptions

Sandbox++

When kernel is not a boundary

virtualization

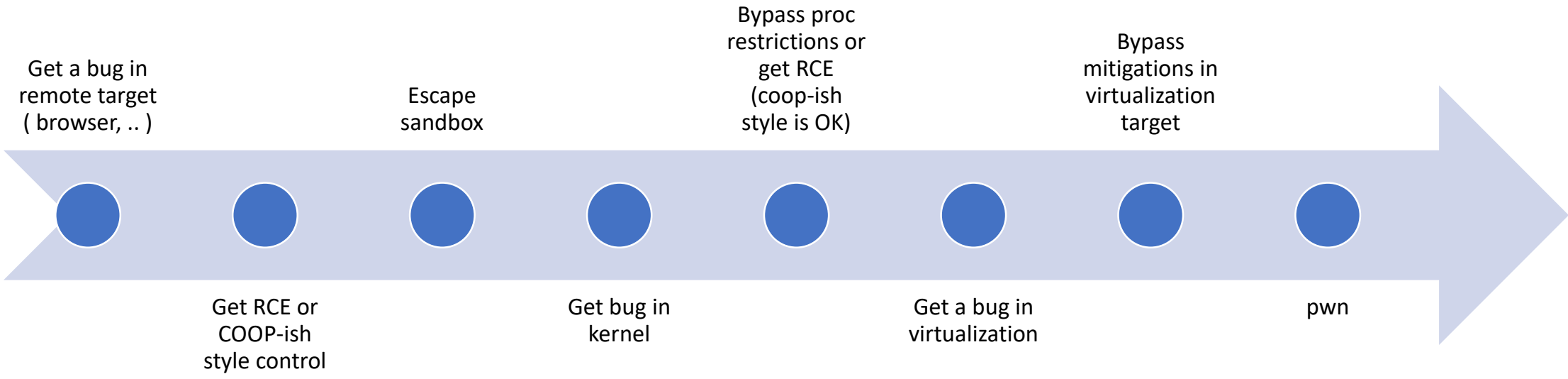
- HyperV technology
  - VM machine
  - Well Security designed!
    - Legacy striped
    - (relatively) small (+ heavily audited) attack surface
    - Mitigations applied
- WDAG applying HyperV technologies
  - Another layer of sandbox introduced for edge
  - And not only for edge!

https://cloudblogs.microsoft.com/microsoftsecure/2018/04/19/introducing-windows-defender-system-guard-runtime-attestation/

vmwp overview

- What ?
  - User mode process on host side responsible for running guest-partition
- Minimum legacy
  - IO devices
  - No complex structures (in IO)
  - Minimal interaction (no Drag&Drop, basic session by default, ..)
  - Generation2 way to go, however Generation1 still default
- Clean design
- All mitigations
- Sandboxed!
- pwn vmwp complexity ~ remote pwn

Successful attack in the future (?)



Bug is just the start line

But even though .. how to ?

How to approach

- Understanding of attack surface
  - Windows landscape
- Understanding of target
  - Reverse engineering & internals
- Make use of technologies :
  - IntelPt (+ QemuPt)
  - windbg + TTD
  - Qemu + KVM
  - Hypervisors (tooling + automatization)
  - BochsPwn reloaded / DigTool alike approaches
- Make use (and proper understanding) of “state of the art” tools
  - syzkaller
  - (k)AFL
- .. **then** make your own patches / tools / plugins

Fuzzing vs Eye-balling

- Fuzzing :
  - Easy to make dummy fuzzer
  - Easy to overengineer fuzzer and kill its randomness
- Eyes :
  - You can easily miss trivial bugs
  - Hard to comprehend complex logic

- Why not combine both ?
  - Make random-enough fuzzing
  - Inject (to fuzzer) knowledge from auditing-code
  - Use fuzzer to check some complex logic for you + automate it!

RCE

- RCE is not all about browsers!
- Microsoft Office
- SMB
  - SMB v1 non default ~ big attack surface
  - non auth attack vector seems finally heavily audited ?

- Most modern apps connect over internet
- Skype, Slack, games, .. ?

Other windows cool targets ~ kernel

- Sockets
- UoW (ubuntu on windows ~ WSL)
- SMB (v1, v2, v3)
- HyperV (user, kernel, hypervisor)
- VhdParser
- RDP
- .. .sys ? UEFI ?