SELECT Bugs FROM Binary WHERE Pattern LIKE CVE-1337-DAYS

Tao Yan (@Ga1ois), Ken Hsu, Bo Qu
Palo Alto Networks

About us

- Security researchers from Palo Alto Networks
  - Tao Yan (@Ga1ois)
  - Ken Hsu
  - Bo Qu
- Vulnerability researchers
  - Multiple times for MSRC Top 10 Researchers.
  - Several hundreds CVEs for Browser, PDF, Office, Windows, etc.
- Pwn2Own Winner
  - Windows EoP category at Pwn2Own 2021
- Conference speakers
  - Black Hat, CanSecWest, Blue Hat, POC, HITCON, Recon, etc.
- Patent inventors
  - New defense and detection techniques

Agenda

- Background and challenges for static code analysis
- A new code pattern search tool for binaries
- Find vulnerability variants and exploitation primitives with code patterns
  - File hijacking pattern
  - Reparse point pattern
  - ACL overwritten pattern
  - Pool spray pattern
- Summary

Background and Challenges for Static Code Analysis

Background for Static Code Analysis

- Why we need static code analysis?
  - Human make similar mistakes repeatedly, so do engineers
    - Lack of check: stack overflow, heap overflow, etc
    - Bad API design: _snprintf, GetFileAttribute, etc
    - Too complicated to make the code right: one msi vulnerability was patched 5 times
  - Similar vulnerable codes or same vulnerable libraries are used in many places, but only be patched in one place
- Is it feasible?
  - Vulnerabilities can be categorized according to the root cause
  - Similar vulnerabilities have similar code patterns
- Code pattern search is useful for vulnerability variants discovery
- Static code analysis for code pattern search is a solution

Challenges for Static Code Analysis

- Effective code pattern extraction from huge amounts of various vulnerabilities
  - If code pattern is too complicated, it is hard to search
  - If code pattern is too simple, noises are too many
- Efficient code pattern search from huge amounts of different modules
- Existing works
  - Most code patterns are for memory corruption vulnerabilities
    - Stack overflow, heap overflow, integer overflow
  - CodeQL is awesome but only works for open-source software

A New Code Pattern Search Tool for Binaries

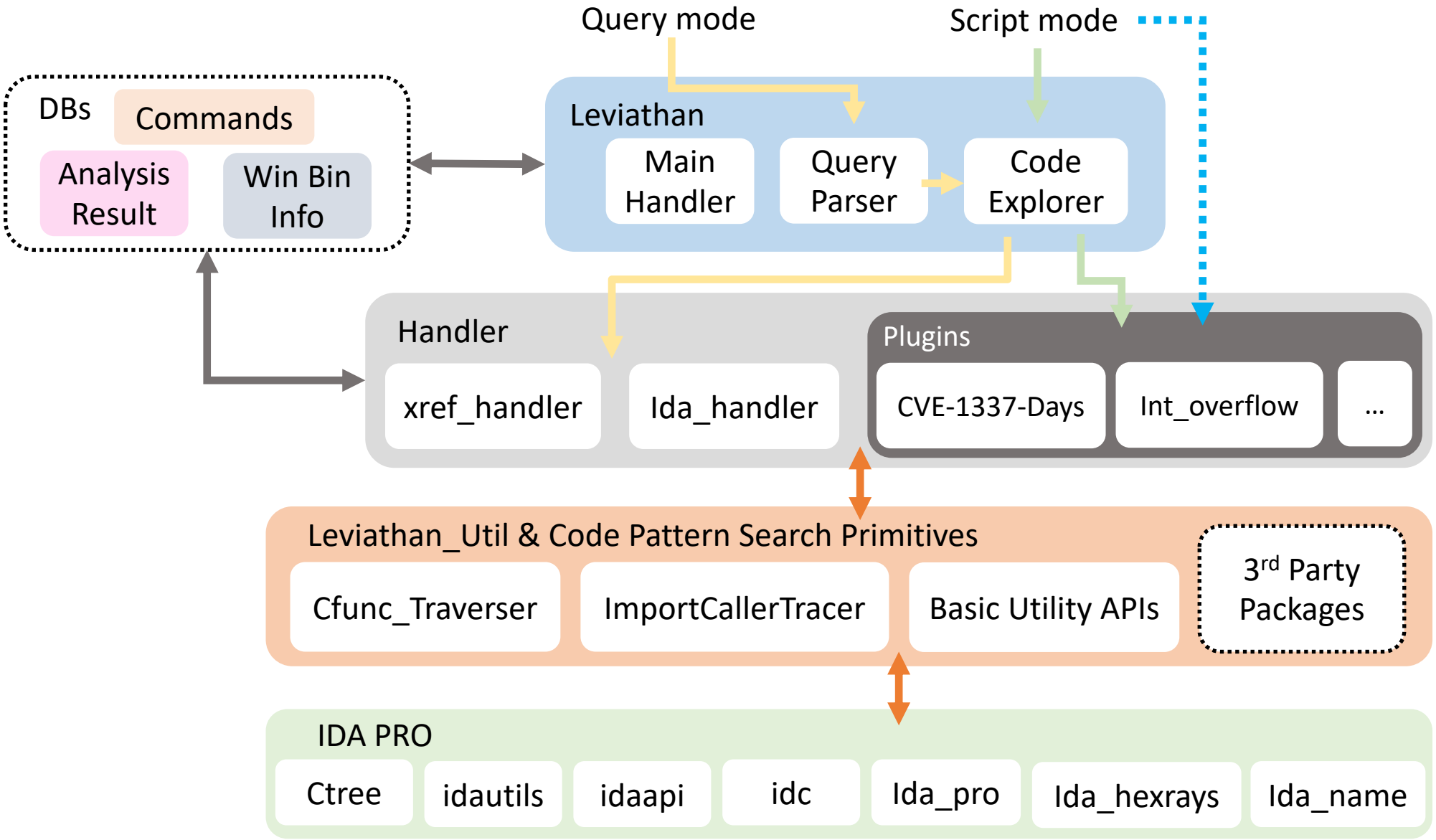
Leviathan

- Our own custom code pattern search tool for binaries
  - Because there’s no off-the-shelf solutions ;(
- Custom query language to facilitate our research
  - Automate away the repetitive and common tasks
- Custom code pattern search primitives and APIs
- Based on IDA Pro and IDAPython
  - idautil, idaapi, ida_hexrays APIs, and Ctree
- Driven by research
  - Not full-blown

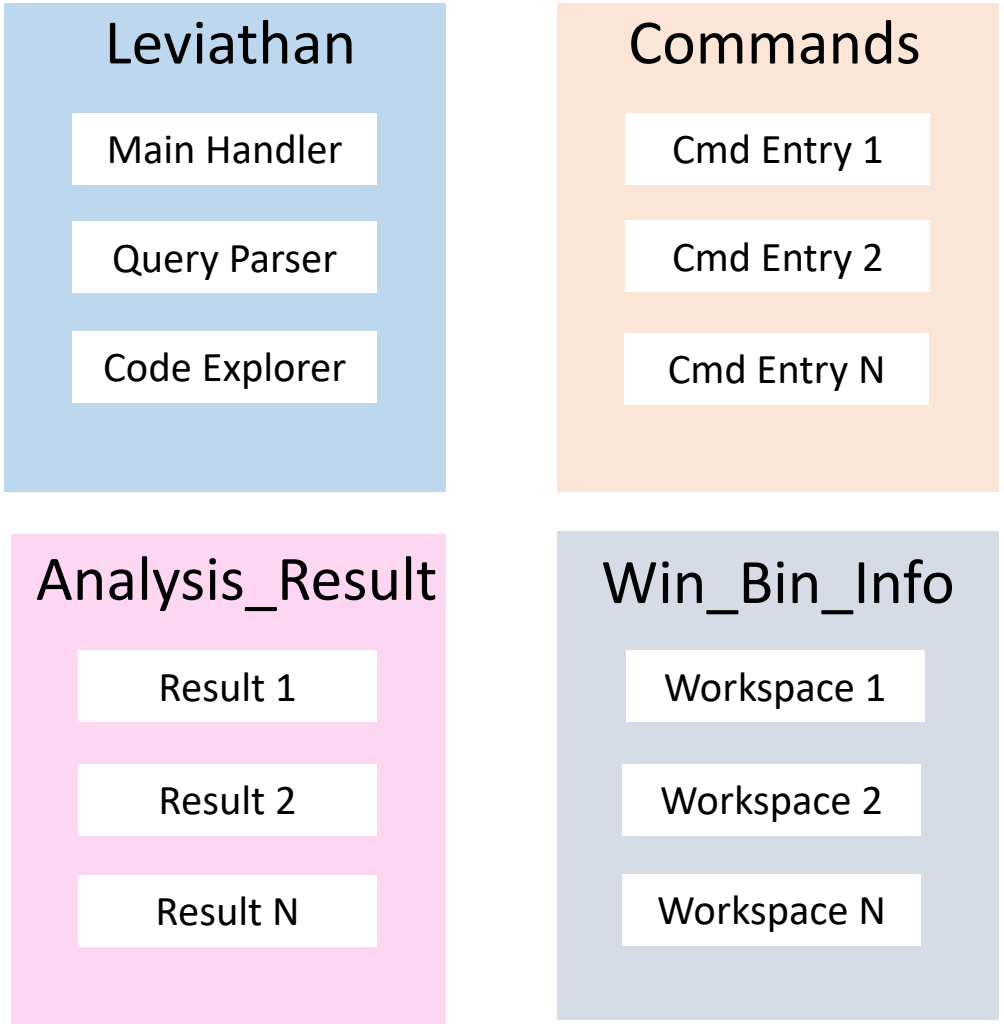
Core Features

- Variable
  - Type and value
- Function
  - Search by function name
  - Arguments (value, type)
  - Return value
  - Bitmask
- Common caller of functions (direct and indirect)
- Condition checks (e.g sanitizer pattern)
- Loop (e.g memory write in loop)
- Find specific code patterns (e.g integer overflow pattern)
- Taint and Flow analysis
  - Source and sink (within 1 function; no cross function)
- Cross Binary Function Call Reference

Design and Architecture



Databases



Commands (SQLite DB)

- Command Entry – enqueued job containing the command query and command hash

Analysis_Result (SQLite DB)

- Analysis Result – Output result

Win_Bin_Info (SQLite DB)

- Workspace – Basic info of all the binaries in the specified directory

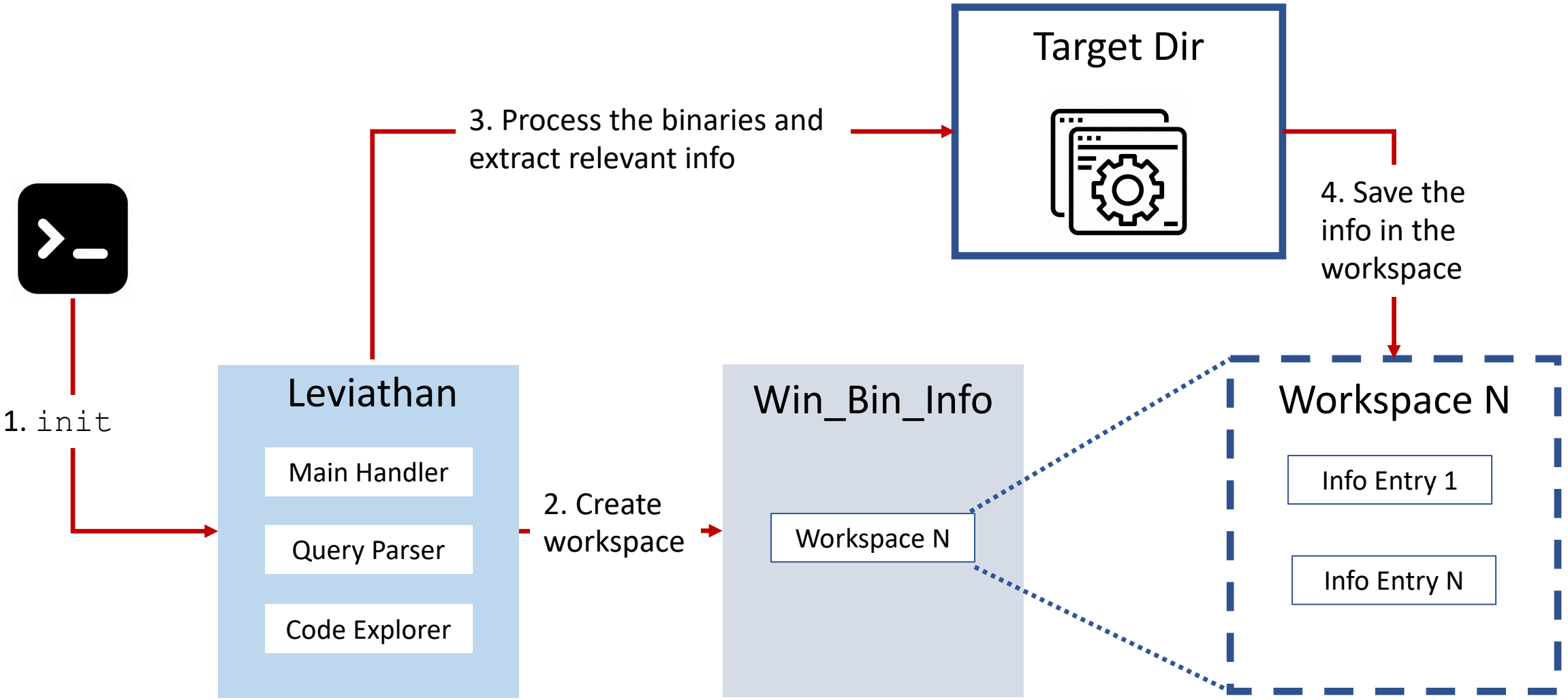
What is Workspace, Exactly?

- Essentially a DB Table
  - It contains all the basic info (like function names, imports, and exports, etc) of IDBs in the specified directory
- Created via `init` command
  - `init_info` plugin does all the heavy lifting
  - In addition to basic info extraction, it can also categorize the IDBs through simple heuristics
    - For example, Windows-system-service-related modules likely have impersonation/revert related functions in the import table
    - The modules that have RPC interfaces are also likely to be system-services. `init_info` obtains such info by incorporating findrpc
  - This can help narrow the scope and thus increases the performance
- Why so many workspaces/tables?
  - For better organization (e.g RDP-related modules in workspace A, system-service related modules in workspace B, etc)

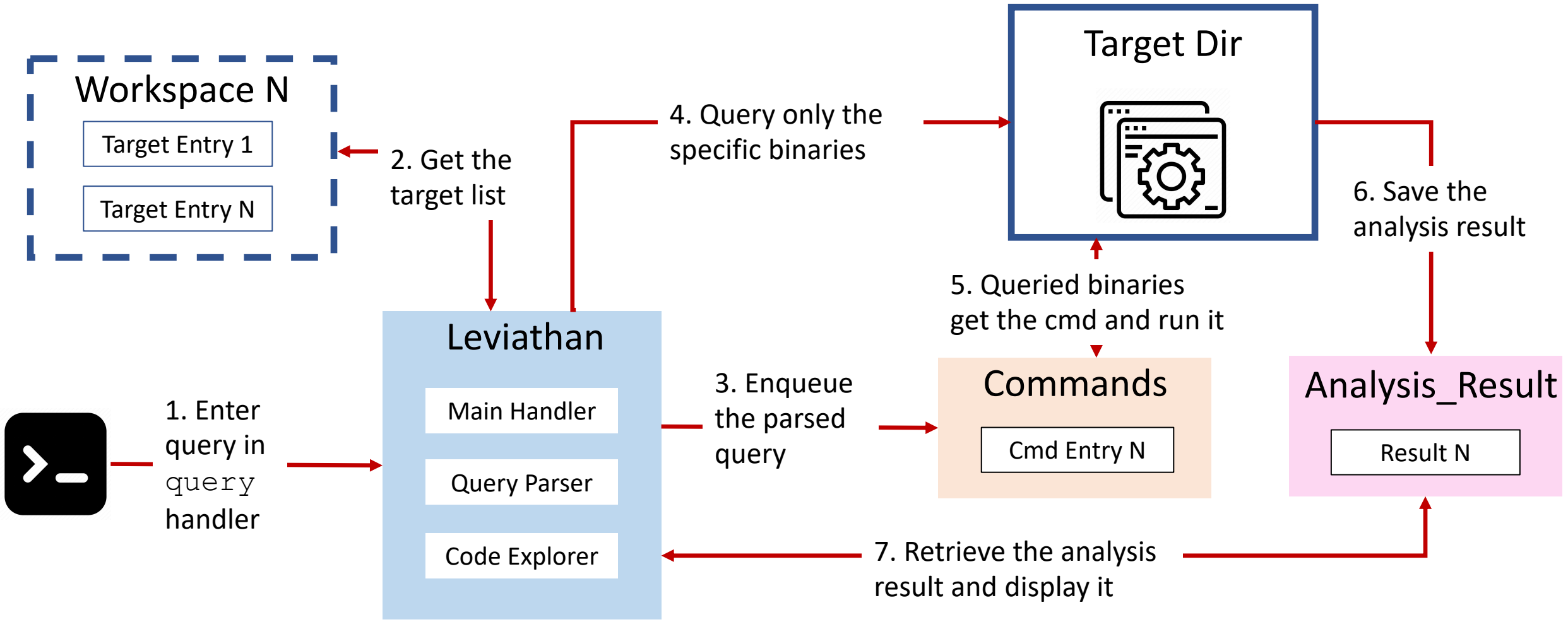
Leviathan - Commands

- Handled by Leviathan’s main handler
- List of commands (bare minimum)
  - `init`
  - `set workspace <workspace name>`
  - `show workspace`
  - `query`
  - `exit/quit`

Workflow – Init Workspace



Workflow – Run Query



black hat EUROPE 2022

Query Mode

- SQL query mode
  - Pseudo-SQL queries for generic search
  - Multiple statements can be chained to find result
- Script mode
  - Based on our own APIs and custom search primitives
  - More control on the granular level since it’s essentially python script leveraging all Leviathan's features

SQL Query Mode

- Simple SQL-query parser based on sqlparse
  - Supports pseudo-SQL syntax
  - Nowhere near any of the full-blown SQL query parsers used in SQL database
    - It's meant to facilitate our bug hunting process
  - Enough to handle common tasks like checking function argument's type and value
  - Access only to a subset of code pattern search primitives and APIs
- Why the (pseudo) SQL syntax?
  - More intuitive

SQL Query Format

```
Select <Output Filter> from workspace.<Target> where <Constraints>
```

- Output Filter
  - Can be `ea`, `func_name`, `mod_name`, or `*`
- Target
  - Target binaries in the workspace
- Constraints
  - The conditions
  - Keywords: `func`, `flow_src`, `flow_sink`, `in_sys32`, `xflow_src`, etc
  - Operators: `is`, `has`, `=`, `not`, etc
  - Special keywords: `Show`, `Join`, etc

Example SQL Queries

```
select * from workspace.elevated_service
where func has loadLibrary and
loadLibrary.arg1.type = “string” and
loadLibrary.arg1.value NOT LIKE “API-*” and
loadLibrary.arg1.value LIKE “*.dll” and
NOT in_system_dir(loadLibrary.arg1.value)
```

Example 1
Find conditional calls to `LoadLibrary` in all system services

```
Select * from workspace.db
where func has wcpreadreparsepoint
```

Example 2
Find `wcpreadreparsepoint` function in all binaries in the current workspace

```
Select * from workspace.daxexec
where func has wcpreadreparsepoint and
flow_src is wcpreadreparsepoint.arg2 and
flow_sink is memcpy.arg3
```

Example 3
For functions that have `wcpreadreparsepoint` in `daxexec` binary, do flow analysis from the arg2 of `wcpreadreparsepoint` to the arg3 of `memcpy`

Parsing and Translating SQL Query

SQL Query

```
Select * from workspace.binary

where func has loadlibraryex and

loadlibraryex.arg1.type = “str” and

loadlibraryex.arg1.value NOT LIKE “API-*” and

loadLibraryex.arg1.value LIKE “*.dll” and

(loadLibraryex.arg3.value & 0x62) = 0 and

NOT in_system_dir(loadlibraryex.arg1.value)
```

Python Dictionary

```
{
TGT_FUNC1_KEY: ["loadlibraryex"],
ANCHOR1_CONDS: [
{CHECK_FUNC_ARG: "loadlibraryex.arg1.type",
TARGET_VALUE: "str", OPERATOR: "eq"},
{CHECK_FUNC_ARG: "loadlibraryex.arg1.value",
TARGET_VALUE: "API-*", OPERATOR:
"regex_not"},
{CHECK_FUNC_ARG: "loadlibraryex.arg1.value",
TARGET_VALUE: "*.dll", OPERATOR: "regex"},
{CHECK_FUNC_ARG: "loadlibraryex.arg3.value",
TARGET_VALUE: 0, OPERATOR: "eq",
VAL_BIT_OPERATOR: "&", VAL_BIT_MASK: 98},
{CALL_FUNC: "in_sys", IN_FUNC_ARG:
"loadlibraryex.arg1.value", NEG_FLAG: True}]
}
```

Handler: How ida_handler Works with cmd_query

Ida_handler

```
cmd_query = cdb.get_cmd_query(job_id)
…
anchors1 = cmd_query[TGT_FUNC1_KEY] if TGT_FUNC1_KEY in cmd_query
else []

anchors2 = cmd_query[TGT_FUNC2_KEY] if TGT_FUNC2_KEY in cmd_query
else []

ict = ImportCallerTracer(anchors1, anchors2)
result_dict = {}
anchor1_conds = {}
anchor2_conds = {}

if ANCHOR1_CONDS in cmd_query and len(cmd_query[ANCHOR1_CONDS]) > 0:
  anchor1_conds[ANCHOR1_CONDS] = cmd_query[ANCHOR1_CONDS]
  ict.set_anchor1_conds(anchor1_conds)
…
ict.trace_direct_callers()
```

cmd_query

```
{
TGT_FUNC1_KEY: ["loadlibraryex"],

ANCHOR1_CONDS: [
{CHECK_FUNC_ARG: "loadlibraryex.arg1.type",
TARGET_VALUE: "str", OPERATOR: "eq"},
{CHECK_FUNC_ARG: "loadlibraryex.arg1.value",
TARGET_VALUE: "API-*", OPERATOR:
"regex_not"},
{CHECK_FUNC_ARG: "loadlibraryex.arg1.value",
TARGET_VALUE: "*.dll", OPERATOR: "regex"},
{CHECK_FUNC_ARG: "loadlibraryex.arg3.value",
TARGET_VALUE: 0, OPERATOR: "eq",
VAL_BIT_OPERATOR: "&", VAL_BIT_MASK: 98},
{CALL_FUNC: "in_sys", IN_FUNC_ARG:
"loadlibraryex.arg1.value", NEG_FLAG: True}]
}
```

Why the Conversion?

- Design decision
  - Need a data type that meets the following requirements
    1. Can easily represent the parsed user input
    2. Can work well with the handler and the internal structs (e.g Ctree)
- More generic and flexible
  - The format of the user's input can change in the future (e.g file, JSON, etc)
    - As long as these input can be translated to this python dictionary format, the underlying primitives and APIs still work
  - In case if we decide to switch to different decompilation framework, the handler and pattern matching mechanism remain intact
    - Just need to implement the Ctree traversal counterpart for that decompiler
- This is not written in the stone

Query Mode Limitation

- Query mode has its limitation
  - Not all patterns can be cleanly and nicely abstracted and/or expressed in SQL query format
  - For example,
```
unsigned int16 v8 = *((WORD *)buf + 0x10) + 0x16;
```
  - To match this pattern, would need something like

```
where … and expr like (this.op = asg and this.rhs.type = "int" and this.lhs.type =
"unsigned int" and this.rhs.rhs.op = "num" and this.rhs.lhs.op = "ptr" and …) and …
```

  - Technically doable but not very usable
  - Too convoluted
- Script mode to the rescue!

Script Mode

- Essentially a specialized version of our generic query handler
  - The command query (in dictionary format) is already embedded, so no need to fetch the parsed query from `Commands` database
- More control on the granular level, hence better accuracy
  - Access to all code pattern search primitives and APIs
  - Incorporated more expression matching that are otherwise convoluted for our pseudo-SQL query parser

Script Mode Format

```
Select <Output Filter> from workspace.<Target> where PATTERN is/like <PLUGIN>
```

- Output Filter
  - Same as query mode
- Target
  - Same as query mode
- Keyword
  - `PATTERN` (case insensitive) is the keyword to invoke script mode
- Plugin
  - The target script to be executed; can be a custom script

Example SQL Query Running Script Mode

```
Select * from workspace.*
where PATTERN is int_overflow
```

Example 1
Search for integer overflow pattern in binaries in the current workspace

```
#int_overflow.py, plugin for script mode
result = OrderedDict()
sanitizer = []
…
…
ct = Cfunc_Traverser(ea)
# ar_list: list of Analysis_Result objects
ar_list = ct.set_constraints(constraints, m_expr, sanitizer)

for ar in ar_list:
  if ar.is_sink_tainted() and len(ar.get_expr_matched()) > 0 and \
     len(ar.get_sanitizer_matched()) == 0:
    result[ea] = ea_name_map[ea]
    # get/save more verbose info from ar
    # …
…
```

```
{
  FLOW_SRC:”blah.ret",
  FLOW_SINK:”memcpy.arg2”
}
```

Custom expression

Custom expression

Leviathan Utility & Pattern Search Primitives

- Basic utility APIs
  - `is_likely_sys_srvc()`
  - `get_ref_to_callers()`
  - `get_caller_chain()`
  - `get_unique_caller_chain()`
  - …
- ImportCallerTracer
  - Function matching
  - Interfaces:
    - `trace_indirect_common_callers()`
    - `trace_direct_callers()`
    - `set_anchor_conds()`
    - …
- Cfunc_Traverser
  - Generic Expression Matching
  - Flow constraint
  - Interfaces:
    - `check_pattern_from_constraint()`
    - `set_constraints()`
    - `check_target_ea_pattern_from_constraint()`
    - `check_flow_between_src_and_sinks()`
    - …

Function Matching: Format and Example

General Format

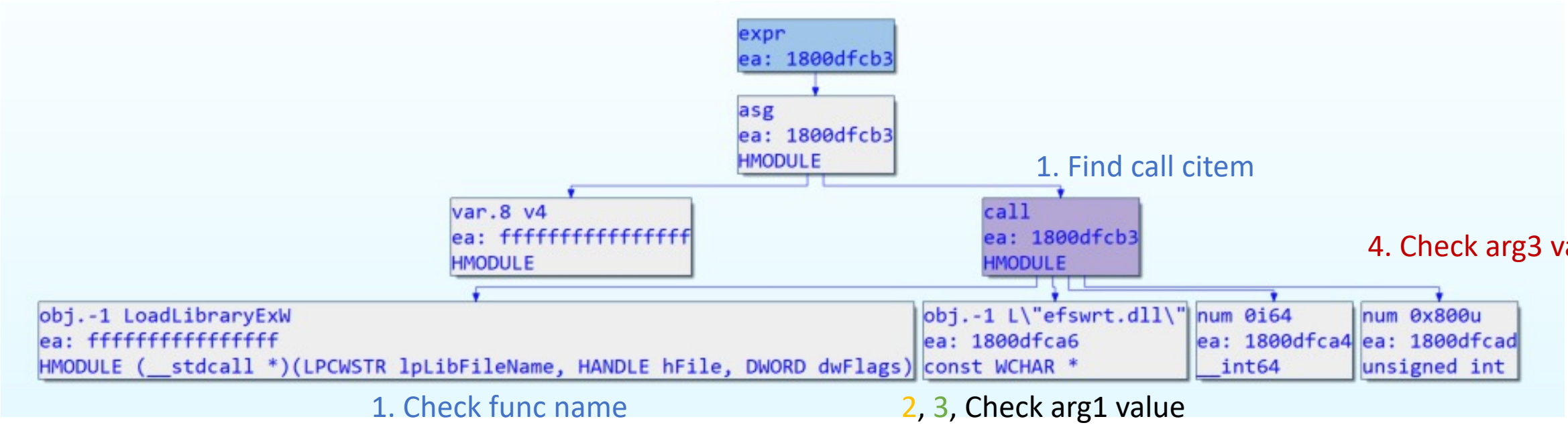
```
{
  TGT_FUNC1_KEY: [<Function Name>],
  ANCHOR1_CONDS: [{CHECK_FUNC_ARG: <arg_expr1>, TARGET_VALUE: <value1>, OPERATOR: <op1>},
{CHECK_FUNC_ARG: <arg_expr2>, TARGET_VALUE: <value2>, OPERATOR: <op2>}, {CHECK_FUNC_ARG:
<arg_expr3>, TARGET_VALUE: <value3>, OPERATOR: <op3>,
{CALL_FUNC: <utility function>, IN_FUNC_ARG: <arg_exp4>, NEG_FLAG: <bool>}] Optional
}
```

Example

```
{
  TGT_FUNC1_KEY: ["loadlibraryex"],
  ANCHOR1_CONDS: [{CHECK_FUNC_ARG: "loadlibraryex.arg1.type", TARGET_VALUE: "str", OPERATOR:
"eq"}, {CHECK_FUNC_ARG: "loadlibraryex.arg1.value", TARGET_VALUE: "API-*", OPERATOR:
"regex_not"}, {CHECK_FUNC_ARG: "loadlibraryex.arg3.value", TARGET_VALUE: 0, OPERATOR: "eq",
VAL_BIT_OPERATOR: "&", VAL_BIT_MASK: 98}, Optional
{CALL_FUNC: "in_sys32", IN_FUNC_ARG: "loadlibraryex.arg1.value", NEG_FLAG: True}]
}
```

How Function Matching Works Underneath

```
1  TGT_FUNC1_KEY: ["loadlibraryex"]
2  {CHECK_FUNC_ARG: "loadlibraryex.arg1.type", TARGET_VALUE: "str", OPERATOR: "eq"}
3  {CHECK_FUNC_ARG: "loadlibraryex.arg1.value", TARGET_VALUE: "API-*", OPERATOR: "regex_not"}
4  {CHECK_FUNC_ARG: "loadlibraryex.arg3.value", TARGET_VALUE: 0, OPERATOR: "eq",
   VAL_BIT_OPERATOR: "&", VAL_BIT_MASK: 98}
```



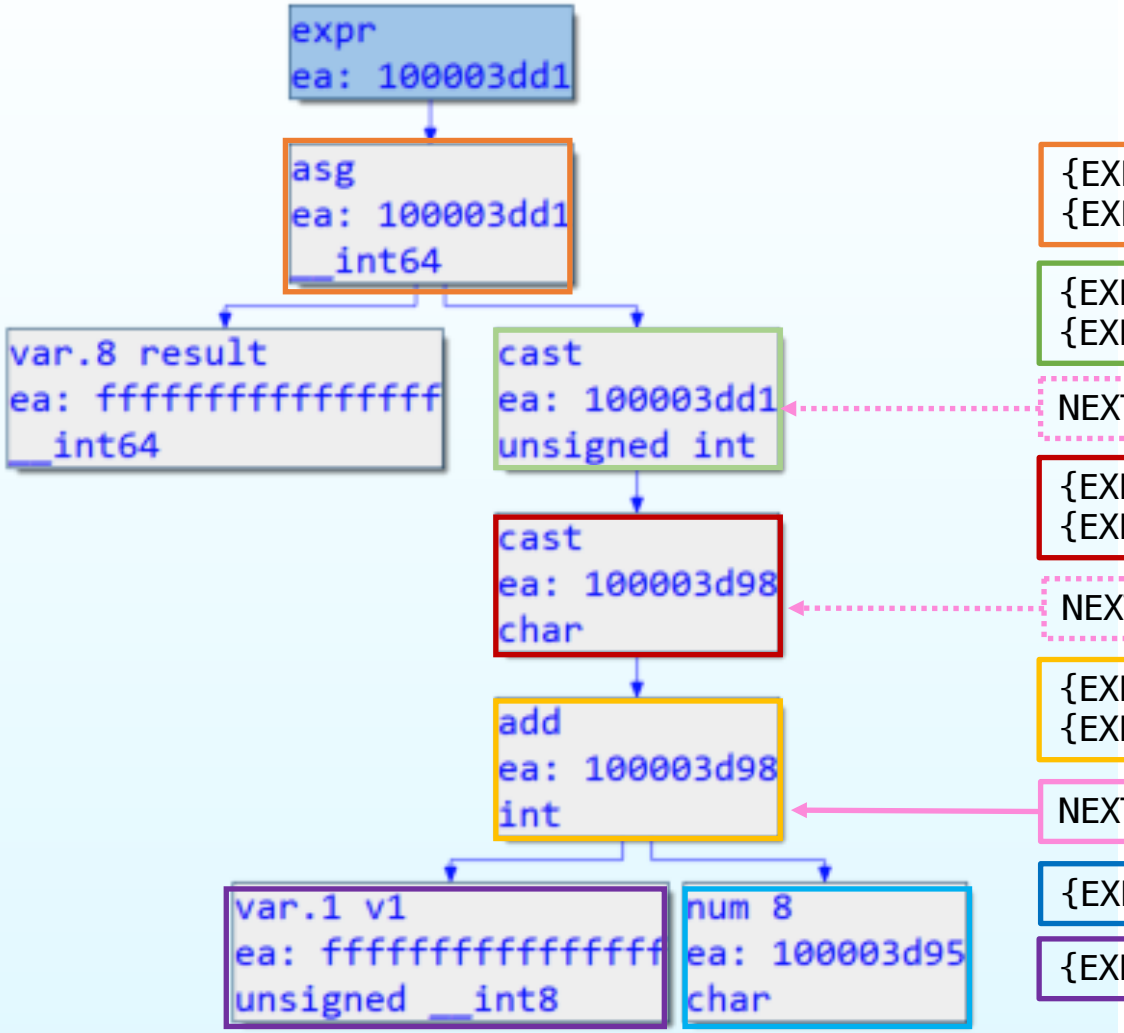
Generic Expression Matching: Format

```
CONDITIONS:[{EXPR:<expr1>, TARGET_VALUE:<value1>, OPERATOR:<op1>}],
NESTED:{
  CONDITIONS:[{EXPR:<expr2>, TARGET_VALUE:<value2>, OPERATOR:<op2>}],
  NEXT:<Node>,
  NESTED:{
    CONDITIONS:[{EXPR:<expr5>, TARGET_VALUE:<value5>, OPERATOR:<op5>}],
  }
}
```

Generic Expression Matching: Example

```
CONDITIONS:[{EXPR:"expr.op", TARGET_VALUE:"asg", OPERATOR:"eq"}, {EXPR:"expr.op.type", TARGET_VALUE:"int",
OPERATOR:"eq"}],
NESTED:{
  CONDITIONS:[{EXPR:"this.rhs.type", TARGET_VALUE:"unsigned int", OPERATOR:"eq"}, {EXPR:"this.rhs.op",
TARGET_VALUE:"cast", OPERATOR:"eq"}],
  NEXT:"this.rhs",
  NESTED:{
    CONDITIONS:[{EXPR:"this.lhs.op", TARGET_VALUE:"cast", OPERATOR:"eq"}, {EXPR:"this.lhs.type",
TARGET_VALUE:"char", OPERATOR:"eq"}],
    NEXT:"this.lhs",
    NESTED:{
      CONDITIONS:[{EXPR:"this.lhs.type", TARGET_VALUE:"int", OPERATOR:"eq"}, {EXPR:"this.lhs.op",
TARGET_VALUE:"arithm", OPERATOR:"eq"}],
      NEXT:"this.lhs",
      NESTED:{
        CONDITIONS:[{EXPR:"this.rhs.op", TARGET_VALUE:"num", OPERATOR:"eq"}, {EXPR:"this.lhs.type",
TARGET_VALUE:"unsigned int", OPERATOR:"eq"}],
      },
    },
  }
}
```

How Expression Matching Works Underneath




```
{EXPR:"expr.op", TARGET_VALUE:"asg", OPERATOR:"eq"}
{EXPR:"expr.op.type", TARGET_VALUE:"int", OPERATOR:"eq"}
```

```
{EXPR:"this.rhs.type", TARGET_VALUE:"unsigned int", OPERATOR:"eq"}
{EXPR:"this.rhs.op", TARGET_VALUE:"cast", OPERATOR:"eq"}
```

```
NEXT:"this.rhs"
```

```
{EXPR:"this.lhs.op", TARGET_VALUE:"cast", OPERATOR:"eq"}
{EXPR:"this.lhs.type", TARGET_VALUE:"char", OPERATOR:"eq"}
```

```
NEXT:"this.lhs"
```

```
{EXPR:"this.lhs.type", TARGET_VALUE:"int", OPERATOR:"eq"},
{EXPR:"this.lhs.op", TARGET_VALUE:"arithm", OPERATOR:"eq"}
```

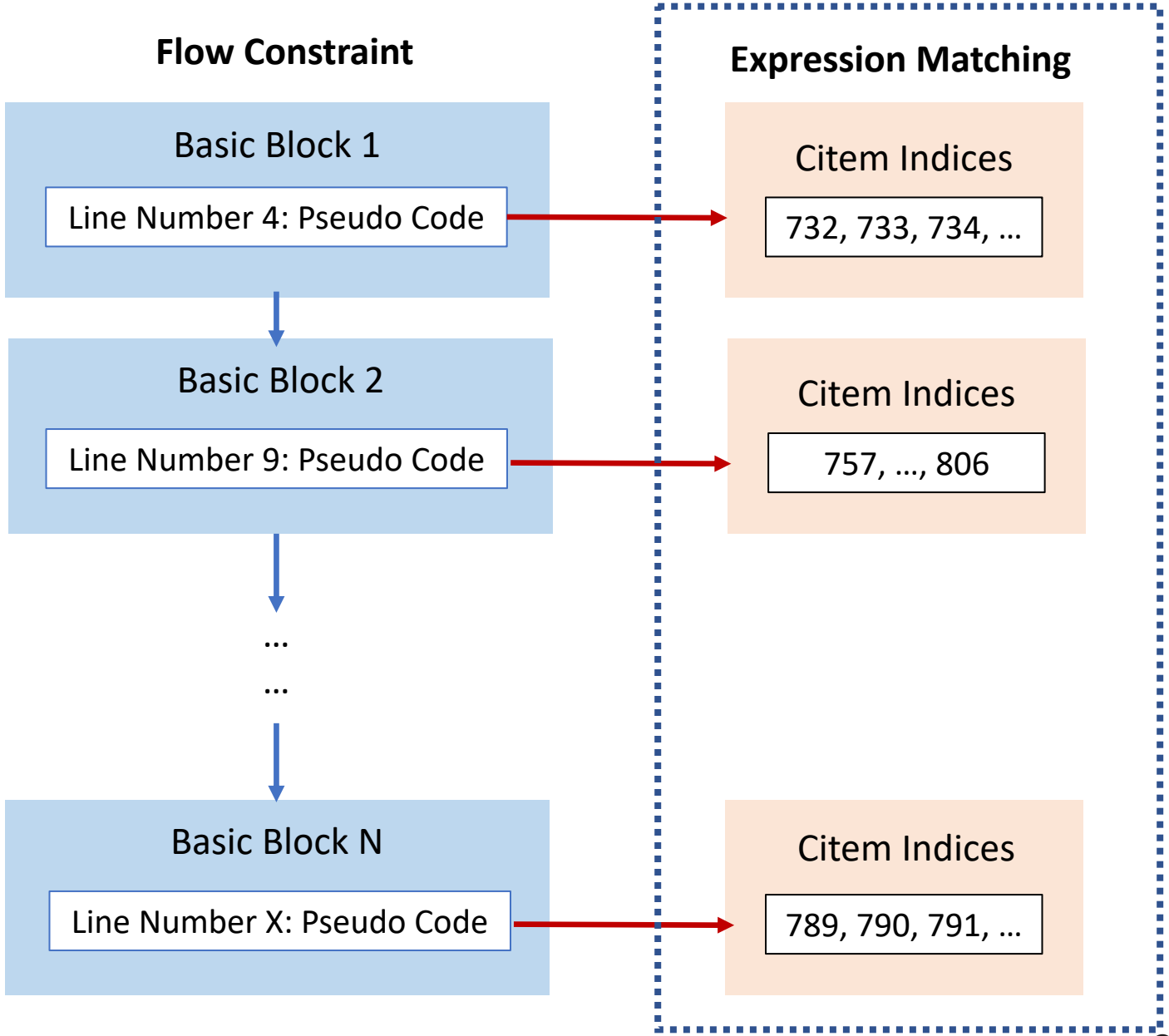
```
NEXT:"this.lhs"
```

```
{EXPR:"this.lhs.type", TARGET_VALUE:"unsigned int", OPERATOR:"eq"}
```

```
{EXPR:"this.rhs.op", TARGET_VALUE:"num", OPERATOR:"eq"}
```

Flow Constraint

- Used for flow/taint analysis
- Essentially, a python dictionary of dictionaries
- First, find the path(s) between the `src` and `sink`
  - `Networkx` graph is pretty sweet
- Each basic block in a path contains a list of citem indices, each of which are ordered by pseudocode line numbers
  - `Lighthouse` plugin has a cool utility that can find relevant citem indices hidden within the associated line of code
- Expression matching when doing flow analysis is optional



Taint Implementation

- Python dynamic attribute for the win!
  - Each citem has a new attribute called `tainted`
- Each variable is also tracked
  - lvars, args, etc
- Taint propagation
  - Start from the `src` (e.g `flow_src` and `taint_src`)
  - Check if the citem's type can propagate taint
    - Op == `*asg`, `ref`, etc
  - Scope is defined by the flow constraint

Cross-Binary Xrefs

xref_handler, processing binary A

```
matched = get_matched_analysis_result_from_joined_job(xflow_sink, joined_jid)
…

for expo_name in matched.keys():
  anchors1 = [expo_name]
  anchors2 = []
  ict = ImportCallerTracer(anchors1, anchors2)
  callers_ref_map = get_func_ea_map(ict.trace_direct_callers())
…

callers_ref_map_all[ea] = caller_ref_map[ea]
…

tgt_ea_callchain_map = OrderedDict()
for ea_key in callers_ref_map_all.keys():
  call_chain = get_caller_chain(ea_key)
  uniq_call_chain = get_unique_caller_chain(call_chain)
  uniq_tgt_call_chain = get_selected_caller_chain(uniq_call_chain, [xflow_src])
  if len(uniq_tgt_call_chain) > 0:
    tgt_ea_callchain_map[ea_key] = uniq_tgt_call_chain
```

cmd_query

```
{
  XFLOW_SRC: <Some func in binary A>,

  XFLOW_SINK: <Some func in binary B>,

  JOIN: <job_id>,
}
```

1. From the job `jid`, find the callpath to export from `xflow_sink`, if any

2. Find the callers that calls these export function from binary B, if any

3. Enumerate call paths of all ref callers from step2 and search for `xflow_src`

Find Vulnerability Variants and Exploitation Primitives With Code Patterns

File Hijacking Pattern: Seed Vulnerabilities

```
void __fastcall Service::RegisterCallbackThunk(struct tagComCallData *a1)
{
  void *v1; // rsi
  HMODULE v2; // rax
  __int64 v3; // rdx
  HMODULE v4; // r14
  HMODULE v5; // rbx
  DWORD v6; // edi

  v1 = a1->pUserDefined;
  if ( (int)SharingPlatform::SharingPlatformStatics::InitializeInternal() >= 0 )
  {
    v2 = LoadLibraryW(L"cdpsgshims.dll");
    v4 = (HMODULE)*((_QWORD *)v1 + 28);
    v5 = v2;
```

NOT FOUND!

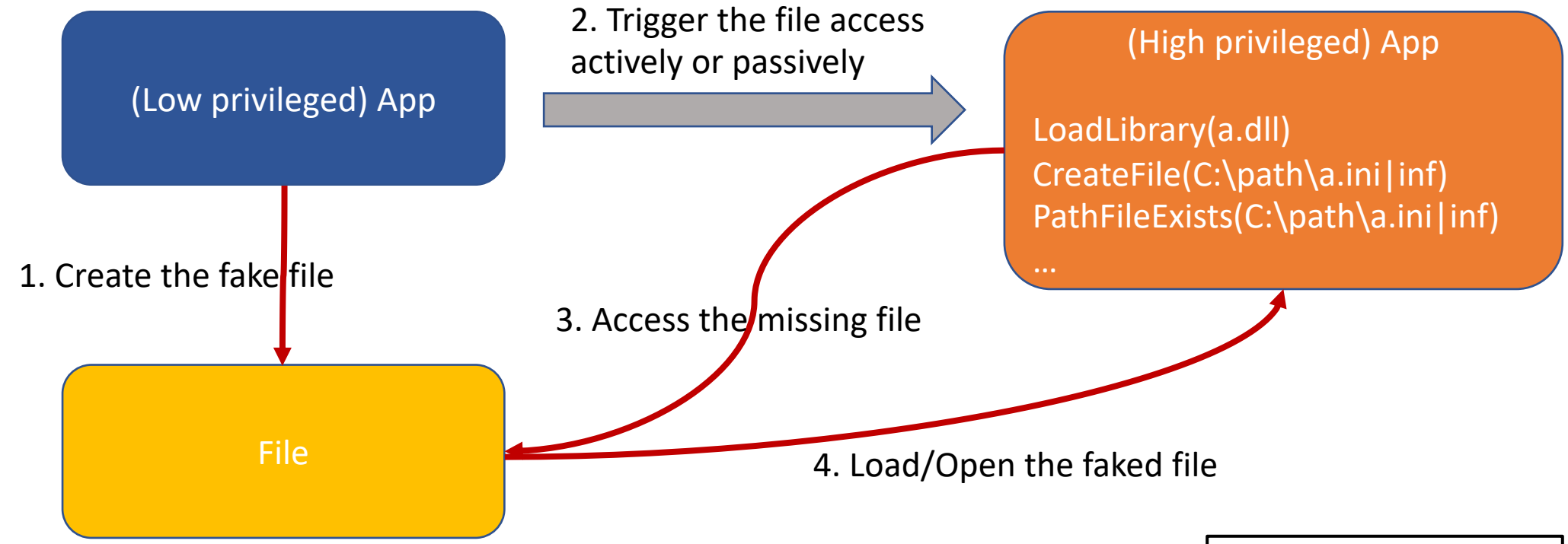
#1 DLL hijacking
Cdpsgshims.dll can be faked in a directory from the PATH environmental variable.

```
  v3 = L"WinBioPlugIns\\FaceDriver\\HelloFace.inf";
  v5 = L"WinBioPlugIns\\FaceDriver\\HelloFaceMigration.inf";
  memset(Buffer, 0, 0x208ui64);
  if ( PathFileExistsW(L"c:\\Analog\\FaceDriver\\HelloFace.inf") )
  {
    if ( (unsigned int)dword_180084000 > 4 )
      fn_EventWriteTransfer((__int64)&dword_180084000, (unsigned __
    v3 = L"c:\\Analog\\FaceDriver\\HelloFace.inf";
    v5 = L"c:\\Analog\\FaceDriver\\HelloFaceMigration.inf";
  }
```

NOT FOUND!

#2 inf hijacking
HelloFace.inf and HelloFaceMigration.inf can be faked by a standard user.

File Hijacking Pattern: Vulnerability Modeling



DLL is not existing in system directories;
Inf|ini file is not existing ~~or controllable by low privileged app/user.~~

Privilege escalation

Whitelist bypass
Malware persistency

File Hijacking Pattern Extraction

- DLL hijacking vulnerability pattern extraction
  - ~~Workspace: elevated service (for privilege escalation vulnerability)~~
  - Sensitive API call
    - LoadLibrary
    - LoadLibraryEx
    - QueryOptionalDelayLoadedAPI
  - DLL file is a constant string
  - DLL file ends with “*.dll”
  - DLL file does not start with “API-*”
  - DLL file is not loaded as data or resource
  - DLL file is not existing in the system directory

File Hijacking Pattern: Code Query Construction

- Queries for LoadLibrary, LoadLibraryExW and QueryOptionalDelayLoadedAPI – query mode

```
Select * from all_binaries.db
where func has LoadLibrary and
LoadLibrary.arg1.type = “string” and
LoadLibrary.arg1.value NOT LIKE “API-*” and
LoadLibrary.arg1.value LIKE “*.dll” and
NOT in_system_dir(LoadLibrary.arg1.value)
```

```
Select * from all_binaries.db
where func has LoadLibraryEx and
LoadLibraryEx.arg1.type = “string” and
LoadLibraryEx.arg1.value NOT LIKE “API-*” and
LoadLibraryEx.arg1.value LIKE “*.dll” and
LoadLibraryEx.arg3.value & 0x62 = 0 and
NOT in_system_dir(LoadLibraryEx.arg1.value)
```

```
Select * from all_binaries.db
where func has QueryOptionalDelayLoadedAPI and
QueryOptionalDelayLoadedAPI.arg2.type = “string” and
QueryOptionalDelayLoadedAPI.arg2.value NOT LIKE “API-*” and
QueryOptionalDelayLoadedAPI.arg2.value LIKE “*.dll” and
NOT in_system_dir(QueryOptionalDelayLoadedAPI.arg2.value)
```

File Hijacking Pattern Result

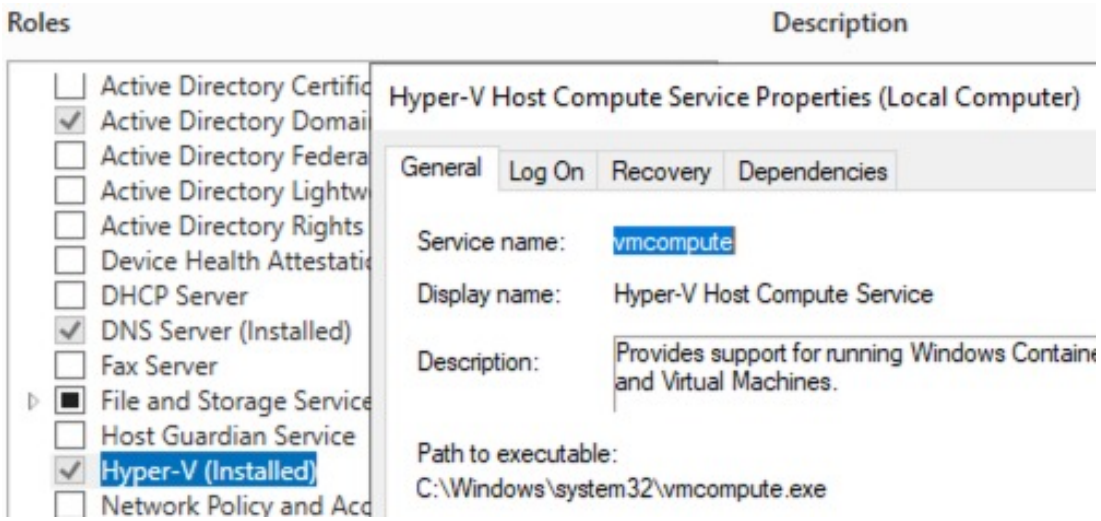
Module	Function	Missing DLL
Certutil.exe	GetFunctionAddressFromCertOcm	Certocm.dll
fveapi.dll	CFveApi::GetFveVolumeHandleForPartition	Fveenabletask.dll
Storsvc.dll	ResetPhoneWorkerCallback	SprintCSP.dll
…	…	…
acmigration.dll	DeviceCensusMatchingPlugin	Centel.dll
appsruprov.dll	InitializeMemoryCounter	EmSvcs.dll
Storsvc.dll	ResetPhoneWorkerCallback	ShellChromeAPI.dll
…	…	…
AboutSettingsHandlers.dll	IsResetDevicePresent	UpdateAPI.dll
ngccredprov.dll	IsResetDevicePresent	UpdateAPI.dll
NgcCtnr.dll	IsResetDevicePresent	UpdateAPI.dll
NgcCtnrSvc.dll	IsResetDevicePresent	UpdateAPI.dll
Vmcompute.exe on Win Server2019	ComputeServiceModule::PrepareToRunSelf	Ccglaunchpad.dll

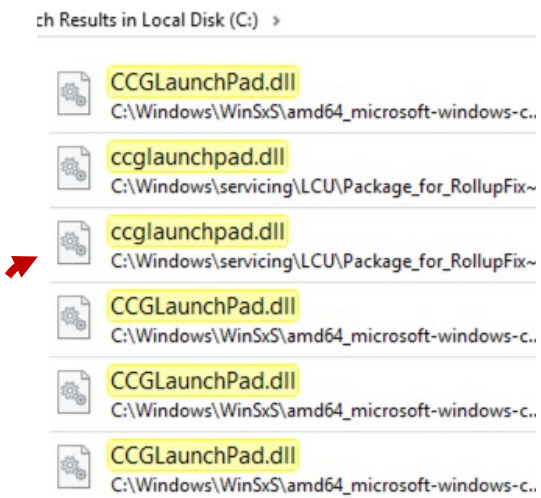
LoadLibrary: 64/3601

LoadLibraryEx: 521/3601

QueryOptionalDelay LoadedAPI: 5/3601

File Hijacking Pattern: Vulnerability Confirmation





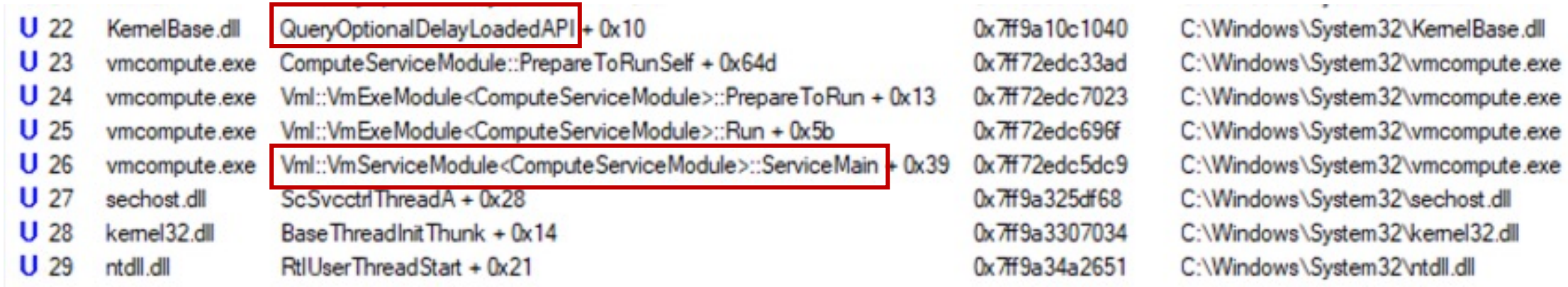
No result in system directories!

```
v29 = LoadLibraryExW(L"computestorage.dll", 0i64, 0x800u);
v30 = GetModuleHandleW(0i64);
if ( !(unsigned int)QueryOptionalDelayLoadedAPI(v30, "ccglaunchpad.dll", "CCGLaunch", 0i64) ||
  v31 = 0;
if ( v29 )
  FreeLibrary(v29);
v32 = GetModuleHandleW(0i64);
v33 = v31 | ((unsigned int)QueryOptionalDelayLoadedAPI(v32, "vmprox.dll", "GetVmErrInfo", 0i64)
EnterCriticalSection(&stru_1402FDD60);
v2 = 0i64;
```

NOT FOUND!

File hijacking pattern: PoC - Triage

- Find connection between the vulnerable function and the trigger function by call stack

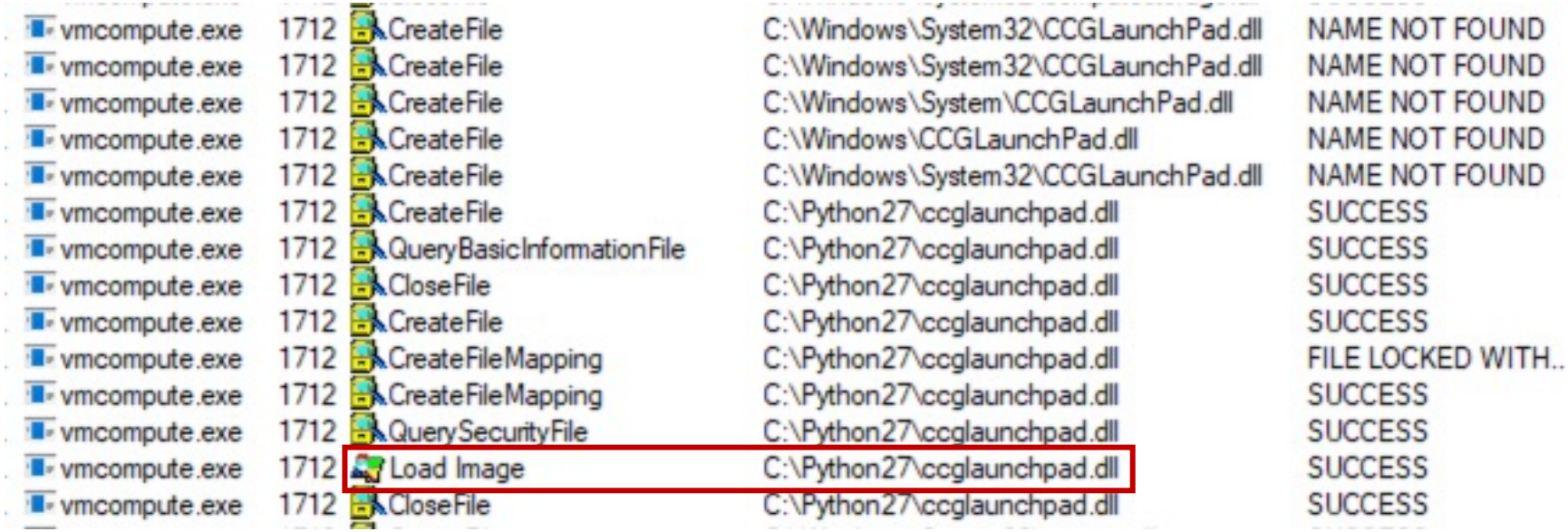


U	22	KernelBase.dll	QueryOptionalDelayLoadedAPI + 0x10	0x7ff9a10c1040	C:\Windows\System32\KernelBase.dll
U	23	vmcompute.exe	ComputeServiceModule::PrepareToRunSelf + 0x64d	0x7ff72edc33ad	C:\Windows\System32\vmcompute.exe
U	24	vmcompute.exe	Vml::VmExeModule<ComputeServiceModule>::PrepareToRun + 0x13	0x7ff72edc7023	C:\Windows\System32\vmcompute.exe
U	25	vmcompute.exe	Vml::VmExeModule<ComputeServiceModule>::Run + 0x5b	0x7ff72edc696f	C:\Windows\System32\vmcompute.exe
U	26	vmcompute.exe	Vml::VmServiceModule<ComputeServiceModule>::ServiceMain + 0x39	0x7ff72edc5dc9	C:\Windows\System32\vmcompute.exe
U	27	sechost.dll	ScSvcctrlThreadA + 0x28	0x7ff9a325df68	C:\Windows\System32\sechost.dll
U	28	kernel32.dll	BaseThreadInitThunk + 0x14	0x7ff9a3307034	C:\Windows\System32\kernel32.dll
U	29	ntdll.dll	RtlUserThreadStart + 0x21	0x7ff9a34a2651	C:\Windows\System32\ntdll.dll

```
C:\Users\administrator>net stop vmcompute
The Hyper-V Host Compute Service service is stopping.
The Hyper-V Host Compute Service service was stopped successfully.


C:\Users\administrator>net start vmcompute
The Hyper-V Host Compute Service service is starting.
The Hyper-V Host Compute Service service was started successfully.
```

File hijacking pattern: PoC - Demo



vmcompute.exe	1712	CreateFile	C:\Windows\System32\CCGLaunchPad.dll	NAME NOT FOUND
vmcompute.exe	1712	CreateFile	C:\Windows\System32\CCGLaunchPad.dll	NAME NOT FOUND
vmcompute.exe	1712	CreateFile	C:\Windows\System\CCGLaunchPad.dll	NAME NOT FOUND
vmcompute.exe	1712	CreateFile	C:\Windows\CCGLaunchPad.dll	NAME NOT FOUND
vmcompute.exe	1712	CreateFile	C:\Windows\System32\CCGLaunchPad.dll	NAME NOT FOUND
vmcompute.exe	1712	CreateFile	C:\Python27\ccglaunchpad.dll	SUCCESS
vmcompute.exe	1712	QueryBasicInformationFile	C:\Python27\ccglaunchpad.dll	SUCCESS
vmcompute.exe	1712	CloseFile	C:\Python27\ccglaunchpad.dll	SUCCESS
vmcompute.exe	1712	CreateFile	C:\Python27\ccglaunchpad.dll	SUCCESS
vmcompute.exe	1712	CreateFileMapping	C:\Python27\ccglaunchpad.dll	FILE LOCKED WITH..
vmcompute.exe	1712	CreateFileMapping	C:\Python27\ccglaunchpad.dll	SUCCESS
vmcompute.exe	1712	QuerySecurityFile	C:\Python27\ccglaunchpad.dll	SUCCESS
vmcompute.exe	1712	Load Image	C:\Python27\ccglaunchpad.dll	SUCCESS
vmcompute.exe	1712	CloseFile	C:\Python27\ccglaunchpad.dll	SUCCESS

vmcompute.exe	< 0.01	1,932 K	8,192 K	1712 Hyper-V Host Compute Servi...	Microso
Lsalso.exe		1,492 K	3,588 K	688	

Name	Description	Company Name	Path
advapi32.dll	Advanced Windows 32 Base API	Microsoft Corporation	C:\Windows\System32\advapi32.dll
bcrypt.dll	Windows Cryptographic Primitives ...	Microsoft Corporation	C:\Windows\System32\bcrypt.dll
bcryptprimitives.dll	Windows Cryptographic Primitives ...	Microsoft Corporation	C:\Windows\System32\bcryptprimitives.dll
ccglaunchpad.dll	Microsoft Debug Device	Microsoft Corporation	C:\Python27\ccglaunchpad.dll

Reparse Point Pattern: Seed Vulnerability

Read reparse point data

```
hFileHandle = CreateFileW(lpFileName, 0x80u, 3u, 0i64, 3u, 0x2200000u, 0i64);
v2 = (__int64)hFileHandle;
if ( hFileHandle == (HANDLE)-1i64
  || !DeviceIoControl(hFileHandle, 0x900A8u, 0i64, 0, lpOutBuffer, 0x4000u, 0i64, 0i64) )//
                                        // 0x900A8: FSCTL_GET_REPARSE_POINT
{
  v8 = GetLastError();
  v5 = (unsigned __int16)v8 | 0x80070000;
  if ( v8 <= 0 )
    v5 = v8;
  goto LABEL_9;
}
if ( *lpOutBuffer == 0xA000001F )       // 0xA000001F: IO_REPARSE_TAG_WCI_TOMBSTONE
{
  v5 = 0x800703FA;
  goto LABEL_9;
}
if ( *lpOutBuffer != 0x80000018         // 0x80000018: IO_REPARSE_TAG_WCI
  && *lpOutBuffer != 0x90001018         // 0x90001018: IO_REPARSE_TAG_WCI_1
  && *lpOutBuffer != 0xA0000027         // 0xA0000027: IO_REPARSE_TAG_WCI_LINK
  && *lpOutBuffer != 0xA0001027 )       // 0xA0001027: IO_REPARSE_TAG_WCI_LINK
{
  v5 = 0x800700A1;
  goto LABEL_9;
}
*lpOut = lpOutBuffer;
000898D1 WcpReadReparsePoint:19 (1800898D1)
```

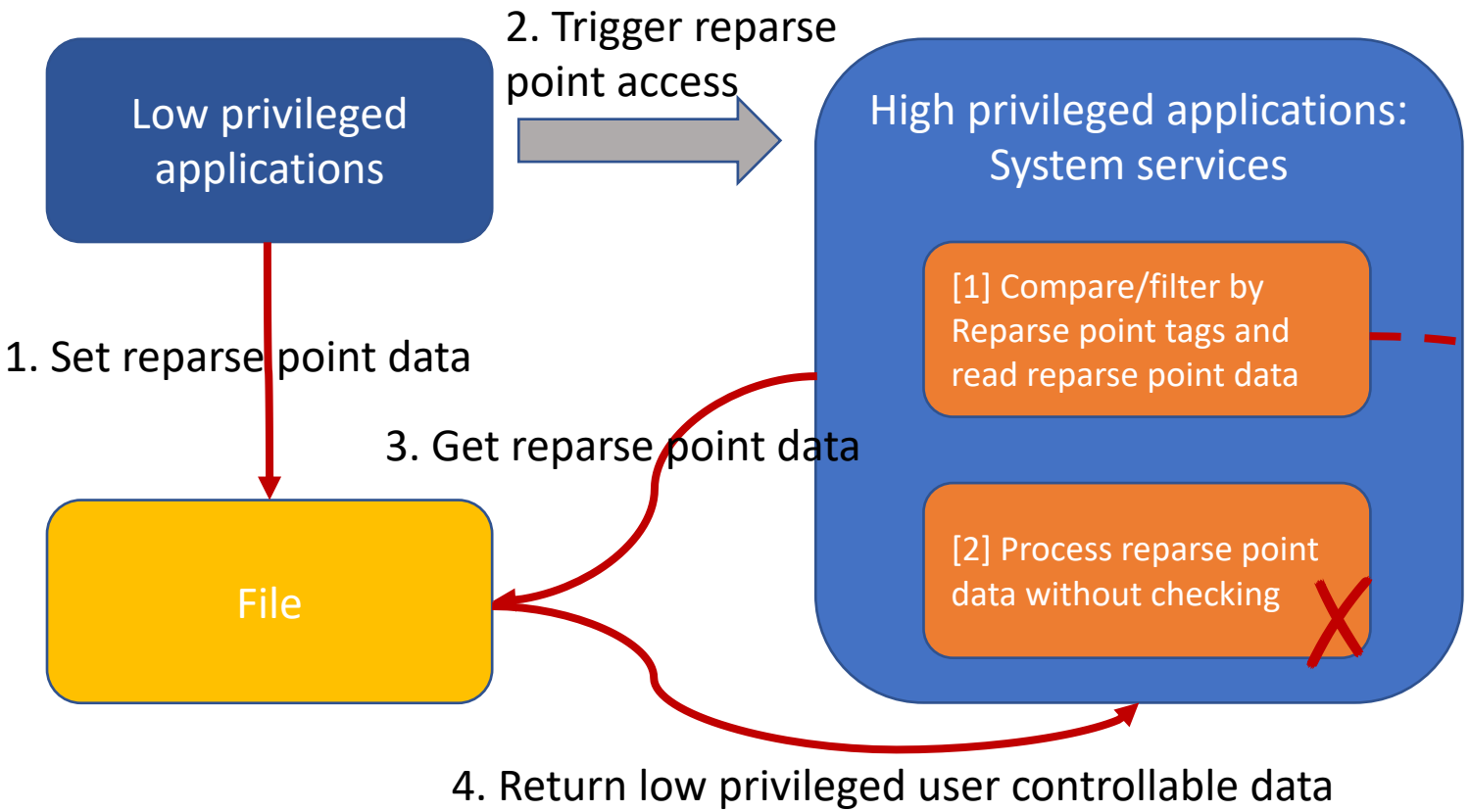
Integer/heap overflow

```
__int64 __fastcall WciReadReparsePointData(__int64 a1, __int64 DstBuffer, unsigned __int16 *pBufferSize)
{
  unsigned __int16 *Block_1; // rbx
  int v6; // eax
  unsigned int v7; // edi
  unsigned __int16 v8; // bp
  char *v9; // rdx
  void *Block; // [rsp+50h] [rbp+18h] BYREF

  Block_1 = 0i64;
  Block = 0i64;
  if ( pBufferSize && (!*pBufferSize || DstBuffer) )
  {
    v6 = WcpReadReparsePoint((const WCHAR *)a1, &Block);
    Block_1 = (unsigned __int16 *)Block;
    v7 = v6;
    if ( v6 >= 0 )
    {
      v8 = *((_WORD *)Block + 0x10) + 0x16;     // integer overflow here
                                                // v8 can larger than *((_WORD *)Block + 0x10) + 0x16
                                                // if *((_WORD *)Block + 0x10) is over 0xffea
                                                // pass the check
      if ( *pBufferSize >= v8 )
      {
        v9 = (char *)Block + 0x22;
        *(_DWORD *)DstBuffer = *((_DWORD *)Block + 3);
        *(_OWORD *)(DstBuffer + 4) = *((_OWORD *)Block_1 + 1);
        *(_WORD *)(DstBuffer + 0x14) = Block_1[0x10] >> 1;
        memcpy_0((void *)(DstBuffer + 0x16), v9, Block_1[0x10]);// heap overflow happens
        *pBufferSize = v8;
        v7 = 0;
      }
      else
      {
        v7 = 0x8007007A;
        *pBufferSize = v8;
      }
    }
  }
  else
  {
00064D95 WciReadReparsePointData:28 (180065995)
```

black hat EUROPE 2022

Reparse Point Pattern: Vulnerability Modeling



REPARSE_POINT_TAG (50+)
IO_REPARSE_TAG_DRIVE_EXTENDER 0x80000005
IO_REPARSE_TAG_HSM2 0x80000006
IO_REPARSE_TAG_SIS 0x80000007
IO_REPARSE_TAG_WCI 0x80000018
IO_REPARSE_TAG_WCI_1 0x90001018
IO_REPARSE_TAG_WCI_LINK 0xA0000027
IO_REPARSE_TAG_WCI_LINK_1 0xA0001027
…

Reparse Point Pattern

- Extract the vulnerability pattern
  - Find reparse point read
    - DeviceIoControl
    - dwIoControlCode (2nd arg) is FSCTL_GET_REPARSE_POINT (0x900A8)
    - Compare *lpOutBuffer (5th arg) with reparse point tags
  - Find integer overflow – optional
    - Taint from lpOutBuffer to source address of memcpy
    - Add a WORD (unsigned int16) value of the controllable (WORD *) pointer with a constant and assign the sum of them to a WORD variable
    - Sanitizer: no length check

Reparse Point Pattern: Code Query Construction

```
# Find Reparse point read – query mode

Select * from workspace.elevated_service
where func has deviceiocontrol and
deviceiocontrol.arg2.value = 0x900A8 and
flow_src is deviceiocontrol.arg5 and
flow_sink is expr.eq.var|expr.neq.var and
flow_sink_type is num and
flow_sink_val = 0x80000018,0xA0001027,0xA0000027,0x90001018,0xA000001F
```

Result: Module Name: daxexec.dll  Func Name: WcpReadReparsePoint

```
# Find reparse point data process – query mode [optional]

Select * from workspace.daxexec
where func has wcpreadreparsepoint and
flow_src is wcpreadreparsepoint.arg2 and
Flow_sink is memcpy.arg3
```

black hat EUROPE 2022

Reparse Point Pattern: Code Query Construction

- Find integer overflow – script mode

```
# Step 1 - find all target funcs
ea_name_dict = find_reparse_point_read() # return value is an ea-name dict

# Step 2 - Get ref to the callers of all funcs found in stage1
callers_ref_dict = find_callers(ea_name_dict) # return value is an ea-name dict

# Step 3 - Check if any result from stage 2 has int_overflow and sanitizer expr
matched
# return value is an ea-name dict
final_result_dict, vi_dict = find_integer_overflow(callers_ref_dict)

# Final Step - Save the result
prepare_entry_and_save_result(job_id, mod_name, mod_md5, final_result_dict,
paths, show_option, vi)
```

Reparse Point Pattern: Code Query Construction

- Find integer overflow: tainting and expression matching – script mode

```
# flow/taint
constraints = {FLOW_SRC:"wcpreadreparsepoint.arg2", FLOW_SINK:"memcpy.arg3"}

# int overflow expr
m_expr = {
    CONDITIONS:[{EXPR:"expr.op", TARGET_VALUE:"asg", OPERATOR:"eq"}, {EXPR:"expr.op.type", TARGET_VALUE:"unsigned int", OPERATOR:"eq"}],
    NESTED:{
        CONDITIONS:[{EXPR:"this.rhs.type", TARGET_VALUE:"int", OPERATOR:"eq"}, {EXPR:"this.rhs.op", TARGET_VALUE:"arithm", OPERATOR:"eq"}],
        NEXT:"this.rhs",
        NESTED:{
            CONDITIONS:[{EXPR:"this.rhs.op", TARGET_VALUE:"num", OPERATOR:"eq"}, {EXPR:"this.lhs.op", TARGET_VALUE:"ptr", OPERATOR:"eq"}],
        }
    }
}

# sanitizer expr
sanitizer_expr = {
    CONDITIONS:[{EXPR:"expr.op", TARGET_VALUE:"ult", OPERATOR:"eq"}],
    NESTED:{
        CONDITIONS:[{EXPR:"this.rhs.type", TARGET_VALUE:"unsigned int", OPERATOR:"eq"}, {EXPR:"this.rhs.op", TARGET_VALUE:"num", OPERATOR:"eq"},
                    {EXPR:"this.lhs.op", TARGET_VALUE:"var", OPERATOR:"eq"}]
    },
    IN_IF:True,
}
```

Unsigned int16 v8 = *((WORD *)block + 0x10) + 0x16

Ult: unsigned less than

if (v8 < 0x16u)

black hat EUROPE 2022

Reparse Point Pattern Result: 5 out of 3601

Module	Function	Vulnerability type	note
AppVEntVirtualization.dll	WcpReadReparsePoint	Integer overflow/OOB Write OOB Read Race condition	Silently patched, no CVE
AppXDeploymentServer.dll	sub_18024AC74	Integer overflow/OOB Write OOB Read Race condition	Silently patched, no CVE
Daxexec.dll	WcpReadReparsePoint	Integer overflow/OOB Write OOB Read Race condition	OOBW:CVE-2021-33759 OOBR: CVE-2021-36957
Wci.dll	WcpReadReparsePoint	Integer overflow/OOB Write OOB Read Race condition	Silently patched, no CVE
[illegible]	[illegible]	[illegible]	[illegible]

Reparse Point Pattern: vulnerability confirmation

Daxexec.dll

```
v6 = WcpReadReparsePoint(a1, &Memory);
v4 = (unsigned __int16 *)Memory;
v7 = v6;
if ( v6 >= 0 )
{
  v8 = *((_WORD *)Memory + 16) + 22;          // integer overflow
  if ( *a3 >= v8 )
  {
    v9 = (char *)Memory + 34;
    *(_DWORD *)a2 = *((_DWORD *)Memory + 3);
    *(_OWORD *)(a2 + 4) = *((_OWORD *)v4 + 1);
    *(_WORD *)(a2 + 20) = v4[16] >> 1;
    memcpy_0((void *)(a2 + 22), v9, v4[16]);// OOB Write & Read
```

AppVEntVirtualization.dll

```
v7 = WcpReadReparsePoint(a2, &Memory);
v5 = (unsigned __int16 *)Memory;
v8 = v7;
if ( v7 >= 0 )
{
  v9 = *((_WORD *)Memory + 16) + 22;          // integer overflow
  if ( *a4 >= v9 )
  {
    v10 = (char *)Memory + 34;
    *(_DWORD *)a3 = *((_DWORD *)Memory + 3);
    *(_OWORD *)((char *)a3 + 4) = *((_OWORD *)v5 + 1);
    *((_WORD *)a3 + 10) = v5[16] >> 1;
    memcpy_0((char *)a3 + 22, v10, v5[16]); // OOB write & read
```

wci.dll

```
v6 = WcpReadReparsePoint(a1, &Memory);
v4 = (unsigned __int16 *)Memory;
v7 = v6;
if ( v6 >= 0 )
{
  v8 = *((_WORD *)Memory + 16) + 22;          // integer overflow
  if ( *a3 >= v8 )
  {
    v9 = (char *)Memory + 34;
    *(_DWORD *)a2 = *((_DWORD *)Memory + 3);
    *(_OWORD *)(a2 + 4) = *((_OWORD *)v4 + 1);
    *(_WORD *)(a2 + 20) = v4[16] >> 1;
    memcpy_0((void *)(a2 + 22), v9, v4[16]);// OOB write & read
```

AppXDeploymentServer.dll

```
v6 = sub_18024AC74(a1, &Memory);
v4 = Memory;
v7 = v6;
if ( v6 >= 0 )
{
  v8 = *a3;
  v9 = *(_WORD *)(Memory + 32) + 22;          // integer overflow
  *a3 = v9;
  if ( v8 >= v9 )
  {
    *(_DWORD *)a2 = *(_DWORD *)(v4 + 12);
    *(_OWORD *)(a2 + 4) = *(_OWORD *)(v4 + 16);
    *(_WORD *)(a2 + 20) = *(_WORD *)(v4 + 32) >> 1;
    memcpy((void *)(a2 + 22), (const void *)(v4 + 34), *(unsigned __int16 *)(v4 + 32));//
                                                // OOB Write & Read
```

Reparse Point Pattern: PoC - Triage

- Call stack to RPC interface function – RAiLaunchProcessWithIdentity in Appinfo.dll

Stack	module	function
0	Daxexec.dll	WciReadParsePoint
1	Daxexec.dll	DesktopAppXVFS::WindowsContainerVfsDetails::GetReparsePointData
2	Daxexec.dll	DesktopAppXVFS::WindowsContainerVfs::ConfigureAppDataBindFilter
3	Daxexec.dll	DesktopAppXVFS::VfsProvider::ApplyMappings
4	Daxexec.dll	DesktopAppXVFS::VfsProvider::ConfigureForActivation
5	Daxexec.dll	helium::Container::Start
6	Daxexec.dll	helium::Container::Create
7	Daxexec.dll	helium::AddProcess
8	Daxexec.dll	PostCreateProcessDesktopAppXActivation
9	AppInfo.dll	RAiLaunchProcessWithIdentity

the vulnerable function

the export function

the RPC interface function

Find connections between the vulnerable function and the RPC interface function by traversing the call stack map with cross modules call function reference matching supported.

```
['Appinfo’] Interfaces :
RPC 0497b57d-2e66-424f-a0c6-157cd5d41700 (1.0) -- c:\windows\system32\appinfo.dll
0 -> RAiLaunchProcessWithIdentity
```

Reparse Point Pattern: PoC - Demo

RPC Client
RAiLaunchProcessWithIdentity

```
(e54.12ec): Access violation - code c0000005 (!!! second chance !!!)
ucrtbase!memcpy_repmovs+0xe:
00007ffd`a9538b5e f3a4            rep movs byte ptr [rdi],byte ptr [rsi]
0:003> r
rax=0000021711670ff6 rbx=0000021711673000 rcx=000000000000fff5
rdx=000000000000202c rsi=000021711167302c rdi=0000021711671000
rip=00007ffda9538b5e rsp=000000daf157d8b8 rbp=0000000000000015
 r8=000000000000ffff  r9=0000000000000000 r10=0000021711673022
r11=0000021711670ff6 r12=0000000000000000 r13=000000daf157e048
r14=0000021711670fe0 r15=0000000000000000
iopl=0         nv up ei pl nz na pe nc
cs=0033  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010200
ucrtbase!memcpy_repmovs+0xe:
00007ffd`a9538b5e f3a4            rep movs byte ptr [rdi],byte ptr [rsi]
0:003> k
 # Child-SP          RetAddr           Call Site
00 000000da`f157d8b8 00007ffd`91e34bda ucrtbase!memcpy_repmovs+0xe
01 000000da`f157d8d0 00007ffd`91de872b daxexec!WciReadReparsePointData+0x92
02 000000da`f157d910 00007ffd`91de9615 daxexec!DesktopAppXVFS::WindowsContainerVfsDetails::GetReparsePointData+0x77
03 000000da`f157d980 00007ffd`91de8fc0 daxexec!DesktopAppXVFS::WindowsContainerVfs::ConfigureAppDataBindFilter+0x15d
04 000000da`f157dbc0 00007ffd`91de65f5 daxexec!DesktopAppXVFS::WindowsContainerVfs::WindowsContainerVfs+0x124
05 000000da`f157dc50 00007ffd`91de83a9 daxexec!DesktopAppXVFS::VfsProvider::ApplyMappings+0x99
06 000000da`f157dcb0 00007ffd`91e1670c daxexec!DesktopAppXVFS::VfsProvider::ConfigureForActivation+0x215
07 000000da`f157df30 00007ffd`91e0b5e3 daxexec!helium::Container::Start+0x94c
08 000000da`f157e6f0 00007ffd`91ddc8c7 daxexec!helium::AddProcess+0x587
09 000000da`f157ea10 00007ffd`9cb826d3 daxexec!PostCreateProcessDesktopAppXActivation+0x337
0a 000000da`f157ecb0 00007ffd`ab61a0e3 appinfo!RAiLaunchProcessWithIdentity+0x1573
0b 000000da`f157f130 00007ffd`ab5a27fb RPCRT4!Invoke+0x73
0c 000000da`f157f210 00007ffd`ab5f7838 RPCRT4!NdrAsyncServerCall+0x2ab
0d 000000da`f157f310 00007ffd`ab65d4c2 RPCRT4!DispatchToStubInCNoAvrf+0x18
0e 000000da`f157f360 00007ffd`ab5d9e06 RPCRT4!DispatchToStubInCAvrf+0x12
0f 000000da`f157f390 00007ffd`ab5d9a36 RPCRT4!RPC_INTERFACE::DispatchToStubWorker+0x1a6
10 000000da`f157f470 00007ffd`ab5e7dbf RPCRT4!RPC_INTERFACE::DispatchToStubWithObject+0x186
11 000000da`f157f510 00007ffd`ab5e7378 RPCRT4!LRPC_SCALL::DispatchRequest+0x16f
12 000000da`f157f5e0 00007ffd`ab5e6961 RPCRT4!LRPC_SCALL::HandleRequest+0x7f8
13 000000da`f157f6f0 00007ffd`ab5e63ce RPCRT4!LRPC_ADDRESS::HandleRequest+0x341
14 000000da`f157f790 00007ffd`ab5ea9d2 RPCRT4!LRPC_ADDRESS::ProcessIO+0x89e
15 000000da`f157f8d0 00007ffd`ab970330 RPCRT4!LrpcIoComplete+0xc2
16 000000da`f157f970 00007ffd`ab9a2f26 ntdll!TppAlpcpExecuteCallback+0x260
17 000000da`f157f9f0 00007ffd`aa7d7034 ntdll!TppWorkerThread+0x456
18 000000da`f157fcf0 00007ffd`ab9a2651 KERNEL32!BaseThreadInitThunk+0x14
19 000000da`f157fd20 00000000`00000000 ntdll!RtlUserThreadStart+0x21
```

ACL Overwritten Pattern: Seed Vulnerability

CVE-2021-34462
used in Pwn2Own 2021

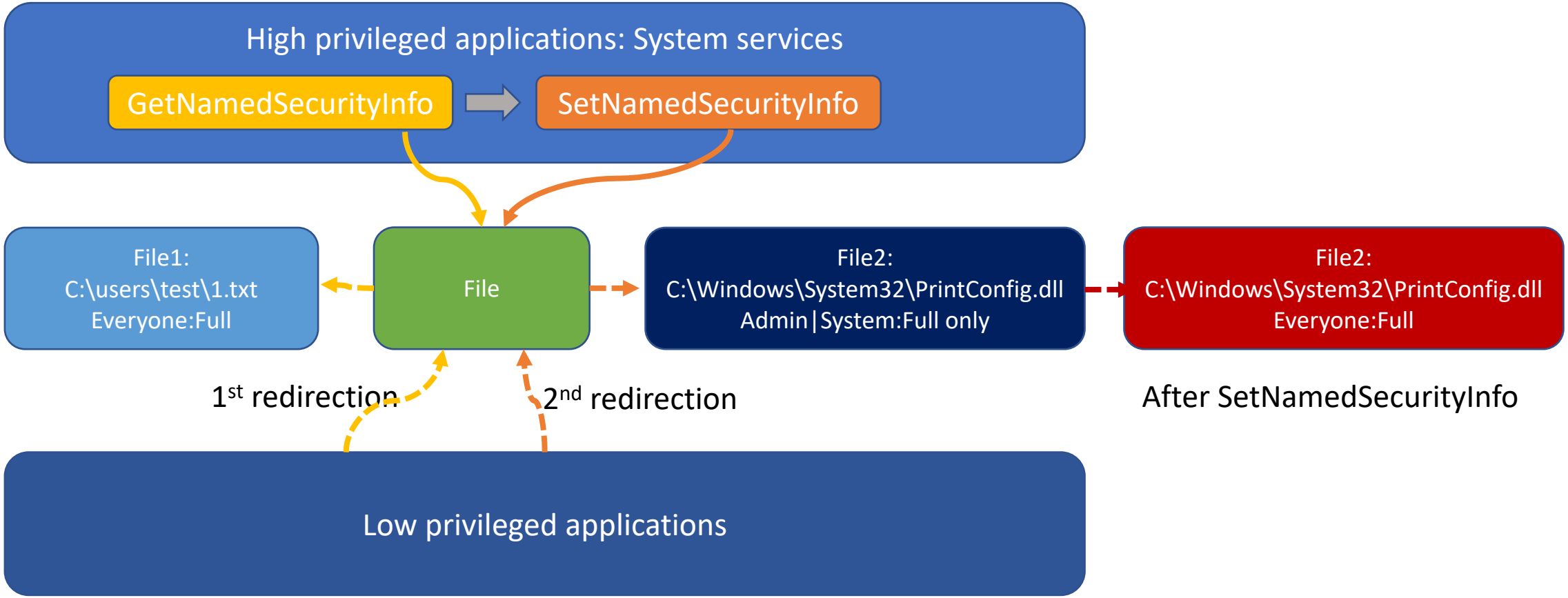
```
154  v20 = GetNamedSecurityInfoW(a1, SE_FILE_OBJECT, 4u, 0i64, 0i64, &pAcl, 0i64, &P);
155  if ( v20 )
156  {
157    v13 = wil::details::in1diag3::Return_Win32(
158            retaddr,
159            (void *)0x405,
160            (unsigned int)"onecore\\admin\\appmodel\\common\\directoryacls.cpp",
161            (const char *)v20,
162            ppsidGroupa);
163LABEL_45:
164    StateRepository::Heap::AutoPtrHeapDeallocate<StateRepository::DatabaseCache>(P);
165    goto LABEL_50;
166  }
167  if ( pAcl )
168  {
169    v21 = CalculateNewDacl(
170            pAcl,
171            v9,
172            v40,
173            ((*(_WORD *)(a3 + 2) ^ *((_WORD *)P + 1)) & 0x3300) == 0,
174            (v39 & 0x20) != 0,
175            &v41,
176            &pDacl);
177    v13 = v21;
178    if ( v21 < 0 )
179    {
180      wil::details::in1diag3::Return_Hr(
181        retaddr,
182        (void *)0x416,
183        (unsigned int)"onecore\\admin\\appmodel\\common\\directoryacls.cpp",
184        (const char *)(unsigned int)v21,
185        ppsidGroupb);
186      v19 = pDacl;
187      goto LABEL_45;
188    }
189    v19 = pDacl;
190    v15 = psidGroup;
191    if ( pDacl )
192      v9 = pDacl;
193  }
194  StateRepository::Heap::AutoPtrHeapDeallocate<StateRepository::DatabaseCache>(P);
195  if ( v41 )
196  {
197LABEL_55:
198    v23 = SetNamedSecurityInfoW(a1, SE_FILE_OBJECT, v18, psidOwner, v15, v9, v10);
```

File: SettingsContainer|mcg: everyone full

Redirect the file to system file: such as PrintConfig.dll

PrintConfig.dll: everyone full

ACL Overwritten Pattern: Vulnerability Modeling



ACL Overwritten Pattern

- Extract the vulnerability pattern
  - Workspace: elevated service
  - GetNamedSecurityInfo
    - 2nd arg is SE_FILE_OBJECT
  - SetNamedSecurityInfo
    - 2nd arg is SE_FILE_OBJECT
  - GetNamedSecurityInfo and SetNamedSecurityInfo have the common caller
    - in the same function directly
    - in the same function indirectly

ACL Overwritten Pattern

- Code query construction – query mode

```
select * from workspace.elevated_service
where func has getnamedsecurityinfo and
func has setnamedsecurityinfo and
getnamedsecurityinfo.arg2.value = 1 and
setnamedsecurityinfo.arg2.value = 1 and
flow_src is getnamedsecurityinfo.arg1 and
flow_sink is setnamedsecurityinfo.arg1 and
trace_indirect = 1
```

ACL Overwritten Pattern Result: 32/637

Module	Function
Vmms.exe	VmFileAccess::Details::CopyAndUpdateFileAclForSid
Vmms.exe	VmFileAccess::Details::FixDirectoryAclForSid
Vmms.exe	VmFileAccess::RevokeDirectoryAccessBySid
Vmms.exe	VmFileAccess::RevokeFileAccessByVirtualMachine
Vmms.exe	FrUndoLogConfigurationManager::RegisterUndoLogConfiguration
Vmms.exe	ResetPhoneWorkerCallback
Vmms.exe	FailoverReplicationTracker::ProcessIncomingDelta
Vmms.exe	FrCtApi::CopyFileAcls
Vmms.exe	FrCtApi::CreateLogFile
Vmms.exe	VmmsSettings::SetDefaultVirtualHardDiskPath
Vmms.exe	VmmsAsyncCDPReferencePointExportTask::CreateFiles
Vmms.exe	WmiMsvmVirtualSystemManagementServiceSettingData::SetDefaultVirtualHardDiskPath
…	…

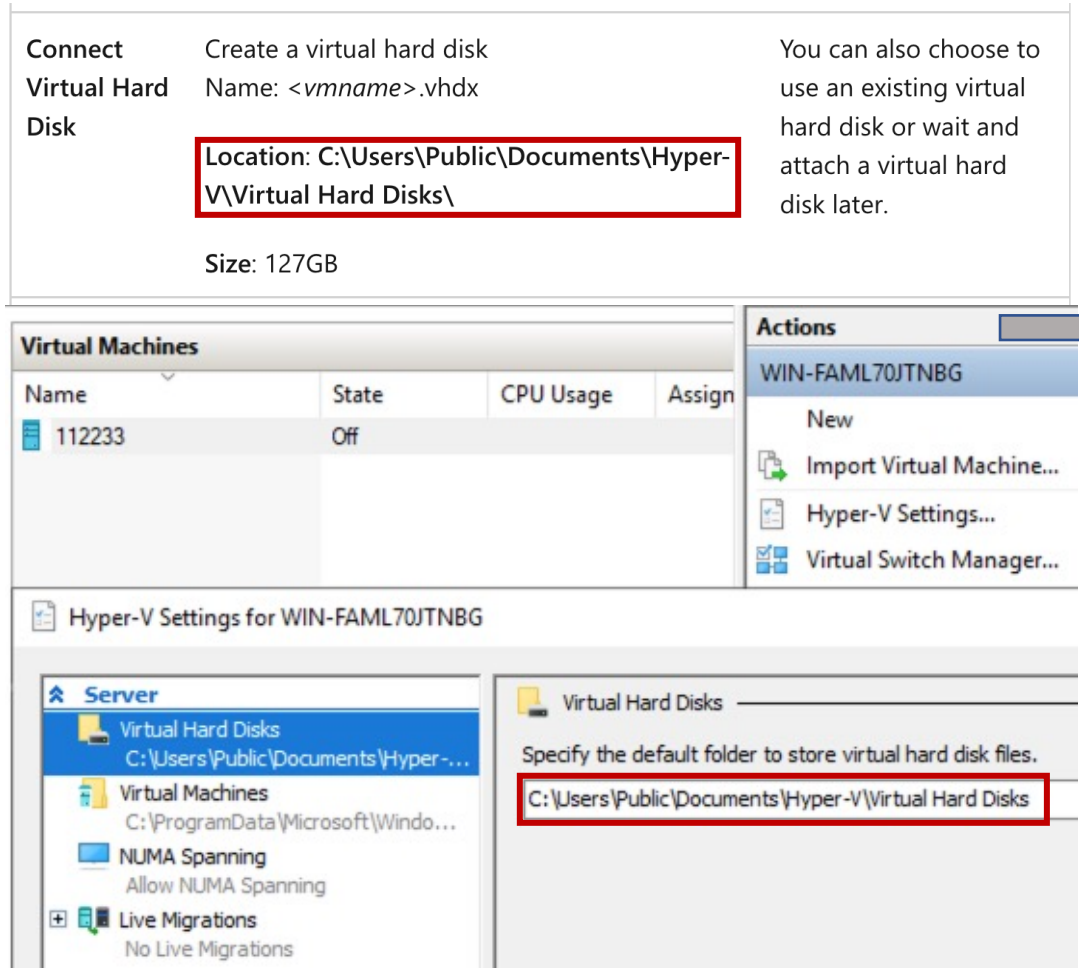
ACL Overwritten Pattern Vulnerability Confirmation

```
v23 = GetNamedSecurityInfoW(v22, SE_FILE_OBJECT, 4u, 0i64, 0i64, 0i64, 0i64, ppSecurityDescriptor);
if ( v23 )
  wil::details::in1diag3::Throw_Win32(
    retaddr,
    (void *)0x2B3,
    (unsigned int)"vm\\service\\mgmt\\vmmssettings.cpp",
    (const char *)v23,
    dwCreationDispositionb);
v24 = Vml::VmSecurityDescriptor::GetDacl((Vml::VmSecurityDescriptor *)ppSecurityDescriptor);
hMem[0] = 0i64;
Vml::VmAcl::Assign((Vml::VmAcl *)hMem, v24);
Vml::VmAcl::UpdateAccess((Vml::VmAcl *)hMem, v42[0], 0x10000000u, 3u, GRANT_ACCESS);
v25 = hMem[0];
Vml::VmSecurityDescriptor::SetDacl(
  (Vml::VmSecurityDescriptor *)ppSecurityDescriptor,
  (const struct _ACL *)hMem[0]);
Vml::VmSecurityDescriptor::MakeSelfRelative((Vml::VmSecurityDescriptor *)ppSecurityDescriptor);
v26 = (WCHAR *)pszBuf;
if ( v41 >= 8 )
  v26 = pszBuf[0];
v27 = (struct _ACL *)Vml::VmSecurityDescriptor::GetDacl((Vml::VmSecurityDescriptor *)ppSecurityDescriptor);
v28 = SetNamedSecurityInfoW(v26, SE_FILE_OBJECT, 4u, 0i64, 0i64, v27, 0i64);
if ( v28 )
  wil::details::in1diag3::Throw_Win32(
    retaddr,
    (void *)0x2C7,
```

```
A ?SetDefaultVirtualHardDiskPath@VmmsSettings@@QEAAXPEBG@Z:134 (14049D32A)
```

Is the file fully controllable by a standard user?

What is default virtual hard disk path?



Not a vulnerability?

```
C:\Users\test>icacls "c:\users\Public\Documents\Hyper-V\Virtual Hard Disks"
c:\users\Public\Documents\Hyper-V\Virtual Hard Disks: Access is denied.
Successfully processed 0 files; Failed processing 1 files

c:\>icacls "c:\users\Public\Documents\Hyper-V\Virtual Hard Disks"
c:\users\Public\Documents\Hyper-V\Virtual Hard Disks
BUILTIN\Administrators:(I)(F)
BUILTIN\Administrators:(I)(OI)(CI)(IO)(F)
BUILTIN\Hyper-V Administrators:(I)(F)
BUILTIN\Hyper-V Administrators:(I)(OI)(CI)(IO)(F)
NT AUTHORITY\SYSTEM:(I)(F)
NT AUTHORITY\SYSTEM:(I)(OI)(CI)(IO)(F)
CREATOR OWNER:(I)(OI)(CI)(IO)(F)
NT VIRTUAL MACHINE\Virtual Machines:(I)(R,WD,AD)
NT VIRTUAL MACHINE\Virtual Machines:(I)(CI)(IO)(GR,WD,AD)
S-1-15-3-1024-2268835264-3721307629-241982045-173645152-1490879176-104643441-
S-1-15-3-1024-2268835264-3721307629-241982045-173645152-1490879176-104643441-
```

But …

- How about a bad standard user create the hyper-v default virtual hard disk folder before the hyper-v installation?

Before hyper-v installation

```
C:\Users\test>dir c:\users\Public\Documents
 Directory of c:\users\Public\Documents
11/22/2022  08:58 AM    <DIR>          .
11/22/2022  08:58 AM    <DIR>          ..
               0 File(s)              0 bytes
               2 Dir(s)  87,537,799,168 bytes free

C:\Users\test>mkdir c:\users\Public\Documents\Hyper-V

C:\Users\test>mkdir "c:\users\Public\Documents\Hyper-V\Virtual Hard Disks"

C:\Users\test>icacls "c:\users\Public\Documents\Hyper-V\Virtual Hard Disks"
c:\users\Public\Documents\Hyper-V\Virtual Hard Disks
BUILTIN\Administrators:(I)(OI)(CI)(F)
GA1OIS\test:(I)(F)
CREATOR OWNER:(I)(OI)(CI)(IO)(F)
NT AUTHORITY\SYSTEM:(I)(OI)(CI)(F)
NT AUTHORITY\INTERACTIVE:(I)(OI)(CI)(M,DC)
NT AUTHORITY\SERVICE:(I)(OI)(CI)(M,DC)
NT AUTHORITY\BATCH:(I)(OI)(CI)(M,DC)
```

install hyper-v

After hyper-v installation

```
C:\Users\test>icacls "c:\users\Public\Documents\Hyper-V"
c:\users\Public\Documents\Hyper-V: Access is denied.
Successfully processed 0 files; Failed processing 1 files

C:\Users\test>icacls "c:\users\Public\Documents\Hyper-V\Virtual Hard Disks"
c:\users\Public\Documents\Hyper-V\Virtual Hard Disks
BUILTIN\Administrators:(I)(OI)(CI)(F)
GA1OIS\test:(I)(F)
CREATOR OWNER:(I)(OI)(CI)(IO)(F)
NT AUTHORITY\SYSTEM:(I)(OI)(CI)(F)
NT AUTHORITY\INTERACTIVE:(I)(OI)(CI)(M,DC)
NT AUTHORITY\SERVICE:(I)(OI)(CI)(M,DC)
NT AUTHORITY\BATCH:(I)(OI)(CI)(M,DC)
```

WoW!

What can a bad standard user do with a fully controllable hyper-v virtual disk folder?

- Read: load the virtual hard disk and read sensitive information inside it leading to info leak.
- Write: replace and inject the malware into the virtual hard disk file leading to code execution.
- Delete: corrupt the virtual hard disk file leading to DoS.
- Bad enough?
- Reported to MSRC, BUT… ALWAYS… YOU KNOW…
- Alright, let’s see something more interesting…

Revisit ACL Overwritten Pattern Result

Module	Function
Vmms.exe	VmFileAccess::Details::CopyAndUpdateFileAclForSid

```
void __fastcall VmFileAccess::Details::CopyAndUpdateFileAclForSid(VmFileAccess::Det
{
  unsigned int v4; // er14
  bool v8; // bl
  struct _ACL *v9; // r8
  HLOCAL hMem[4]; // [rsp+30h] [rbp-A8h] BYREF
  PSID pSid1[10]; // [rsp+50h] [rbp-88h] BYREF

  hMem[2] = (HLOCAL)-2i64;
  v4 = (unsigned int)a4;
  Vml::VmSid::VmSid((Vml::VmSid *)pSid1, WinAnonymousSid, (void *)a3);
  v8 = !EqualSid(pSid1[0], (PSID)a3);
  Vml::VmSid::~VmSid((Vml::VmSid *)pSid1);
  if ( v8 )
  {
    VmFileAccess::Details::GetFileObjectDacl(hMem, (const char *)this, 1);//
                                                // call GetNamedSecurityInfoW
    Vml::VmAcl::UpdateAccess((Vml::VmAcl *)hMem, (void *)a3, 0, 0, REVOKE_ACCESS);
    Vml::VmAcl::UpdateAccess((Vml::VmAcl *)hMem, (void *)a3, v4, 0, GRANT_ACCESS);
    VmFileAccess::Details::SetFileObjectDacl((VmFileAccess::Details *)a2, (const ur
                                                // call SetNamedSecurityInfoW
    if ( hMem[0] )
      LocalFree(hMem[0]);
  }
}
```

Module	Function
Vmms.exe	VmFileAccess::RevokeFileAccessByVirtualMachine

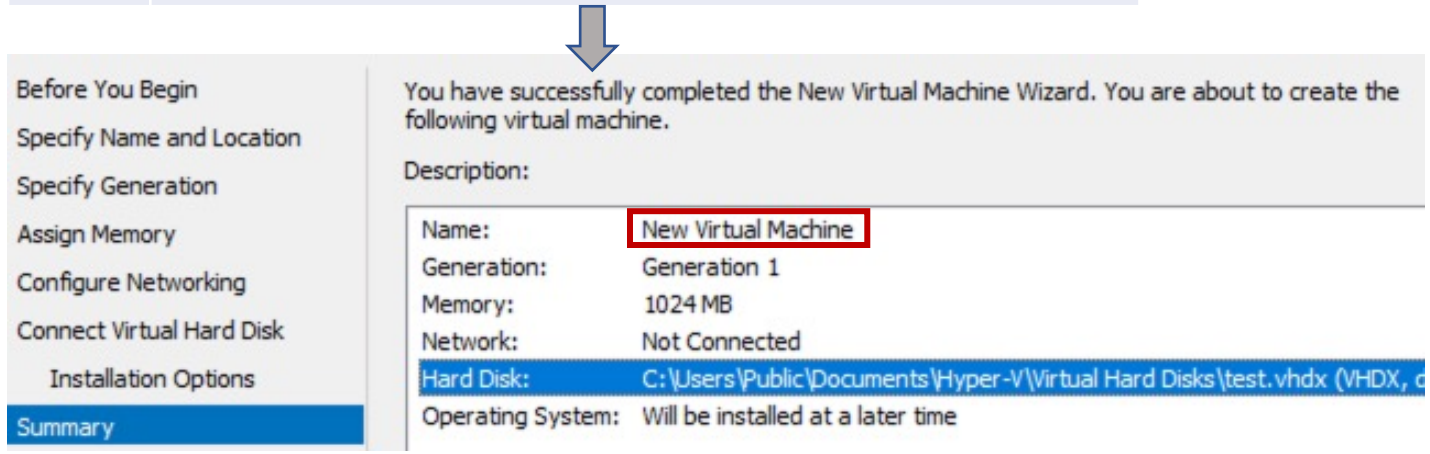
```
void __fastcall VmFileAccess::RevokeFileAccessByVirtualMachine(VmFileAccess *this, const u
{
  HLOCAL v5; // rcx
  struct _ACL *v6; // r8
  HLOCAL hMem[4]; // [rsp+30h] [rbp-88h] BYREF
  void *v8[10]; // [rsp+50h] [rbp-68h] BYREF

  hMem[2] = (HLOCAL)-2i64;
  VmFileAccess::Details::GetFileObjectDacl(hMem, (const char *)this, 0);//
                                                // call GetNamedSecurityInfoW
  v5 = hMem[0];
  if ( hMem[0] )
  {
    VmFileAccess::Details::GetVirtualMachineSidForPath(v8, this, a2);
    Vml::VmAcl::UpdateAccess((Vml::VmAcl *)hMem, v8[0], 0, 0, REVOKE_ACCESS);
    VmFileAccess::Details::SetFileObjectDacl(this, (const unsigned __int16 *)hMem[0], v6);
                                                // call SetNamedSecurityInfoW
    Vml::VmSid::~VmSid((Vml::VmSid *)v8);
    v5 = hMem[0];
  }
  if ( v5 )
    LocalFree(v5);
}
```

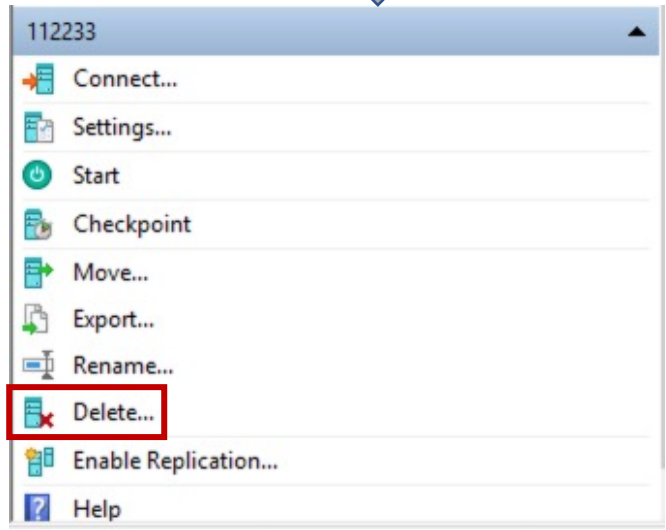
C:\users\public\document\hyper-v\virtual hard disks\test.vhdx

ACL Overwritten Pattern: PoC - Triage

Stack	Function
0	VmFileAccess::Details::CopyAndUpdateFileAclForSid
1	VmFileAccess::FixFileAclForVirtualMachine
2	SetVhdAcls
3	AddVmAccess
4	WmiStorageUtilities::WriteAttachmentInfo
5	WmiStorageUtilities::CreateNewStorageAttachment



Stack	Function
0	VmFileAccess::RevokeFileAccessByVirtualMachine
1	VmmsRealizedVirtualMachine::RemoveVmIdPermissionsFromVhds
2	VmmsRealizedVirtualMachine::DeleteSelf



ACL Overwritten Pattern: PoC – Demo (CVE-2022-TBD)

vmms.exe	3452	CreateFile	C:\Users\Public\Documents\Hyper-V\Virtual Hard Disks\112233.vhdx	SUCCESS	Desired Access: Read Attributes, Read Control, Disposition: Open, Options: Open Reparse Point, Attributes: n
vmms.exe	3452	QueryRemoteProtocolInformation	C:\Users\Public\Documents\Hyper-V\Virtual Hard Disks\112233.vhdx	INVALID PARAMET...	
vmms.exe	3452	QuerySecurityFile	C:\Users\Public\Documents\Hyper-V\Virtual Hard Disks\112233.vhdx	SUCCESS	Information: DACL
vmms.exe	3452	CloseFile	C:\Users\Public\Documents\Hyper-V\Virtual Hard Disks\112233.vhdx	SUCCESS	
vmms.exe	3452	CreateFile	C:\Users\Public\Documents\Hyper-V\Virtual Hard Disks\112233.vhdx	REPARSE	Desired Access: Read Attributes, Read Control, Write DAC, Disposition: Open, Options: Open Reparse Point,
vmms.exe	3452	CreateFile	C:\Windows\System32\license.rtf	SUCCESS	Desired Access: Read Attributes, Read Control, Write DAC, Disposition: Open, Options: Open Reparse Point,
vmms.exe	3452	QueryBasicInformationFile	C:\Windows\System32\license.rtf	SUCCESS	CreationTime: 6/25/2022 6:48:29 AM, LastAccessTime: 6/25/2022 6:48:29 AM, LastWriteTime: 4/30/2020
vmms.exe	3452	QueryNameInformationFile	C:\Windows\System32\license.rtf	SUCCESS	Name: \Windows\System32\license.rtf
vmms.exe	3452	CreateFile	C:\Windows\System32	SUCCESS	Desired Access: Read Control, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode:
vmms.exe	3452	QueryRemoteProtocolInformation	C:\Windows\System32	INVALID PARAMET...	
vmms.exe	3452	QuerySecurityFile	C:\Windows\System32	SUCCESS	Information: DACL
vmms.exe	3452	CloseFile	C:\Windows\System32	SUCCESS	
vmms.exe	3452	QueryRemoteProtocolInformation	C:\Windows\System32\license.rtf	INVALID PARAMET...	
vmms.exe	3452	QuerySecurityFile	C:\Windows\System32\license.rtf	SUCCESS	Information: Owner, Group, DACL
vmms.exe	3452	SetSecurityFile	C:\Windows\System32\license.rtf	SUCCESS	Information: DACL
vmms.exe	3452	CloseFile	C:\Windows\System32\license.rtf	SUCCESS	

```
Breakpoint 1 hit
vmms!VmFileAccess::Details::GetFileObjectDacl:
00007ff6`b6d138d4 488bc4          mov     rax,rsp
0:019> g
Breakpoint 0 hit
vmms!VmFileAccess::Details::SetFileObjectDacl:
00007ff6`b6d13858 4c8bdc          mov     r11,rsp
0:019> gu
vmms!VmFileAccess::Details::CopyAndUpdateFileAclForSid+0xbf:
00007ff6`b6d1381b 90              nop
0:019> k
 # Child-SP          RetAddr           Call Site
00 000000b8`9727d6c0 00007ff6`b6d19d02 vmms!VmFileAccess::Details::CopyAndUpdateFileAclForSid+0xbf
01 000000b8`9727d7a0 00007ff6`b6d166a0 vmms!VmFileAccess::FixFileAclForVirtualMachine+0x56
02 000000b8`9727d890 00007ff6`b6d6a9f9 vmms!SetVhdAcls+0x318
03 000000b8`9727dab0 00007ff6`b6d00655 vmms!AddVmAccess+0x18d
04 000000b8`9727dbf0 00007ff6`b6cfcf34 vmms!WmiStorageUtilities::WriteAttachmentInfo+0x7b5
05 000000b8`9727e4a0 00007ff6`b6c70786 vmms!WmiStorageUtilities::CreateNewStorageAttachment+0x3d8
06 000000b8`9727eb00 00007ff6`b6d2a6c4 vmms!WmiMsvmIdeControllerResolver::GetObjectSelf+0x1b6
07 000000b8`9727eb90 00007ff6`b6ce335e vmms!WmiResolverBase::GetObjectW+0xd4
08 000000b8`9727ec10 00007ff6`b6cbc45f vmms!GetObjectFromVirtualDevice+0xbe
09 000000b8`9727ec90 00007ff6`b6d1516c vmms!WmiMsvmVirtualSystemManagementService::AddOneResourceSetti
0a 000000b8`9727f110 00007ff6`b6d15964 vmms!WmiMsvmVirtualSystemManagementService::AddResourceSettings
0b 000000b8`9727f290 00007ff6`b6d156d6 vmms!WmiMsvmVirtualSystemManagementService::ExecuteStandaloneHo
0c 000000b8`9727f320 00007ff6`b6d72be2 vmms!WmiMsvmVirtualSystemManagementService::ExecuteMethodSelf+0
0d 000000b8`9727f380 00007ff6`b6d63aae vmms!VmWmiObjectBase::ExecuteMethod+0x92
```

What's the magic?
The (hyper-v) admin operates VM in the admin session.
The exploiter (standard user) redirects the file in the test user session :)

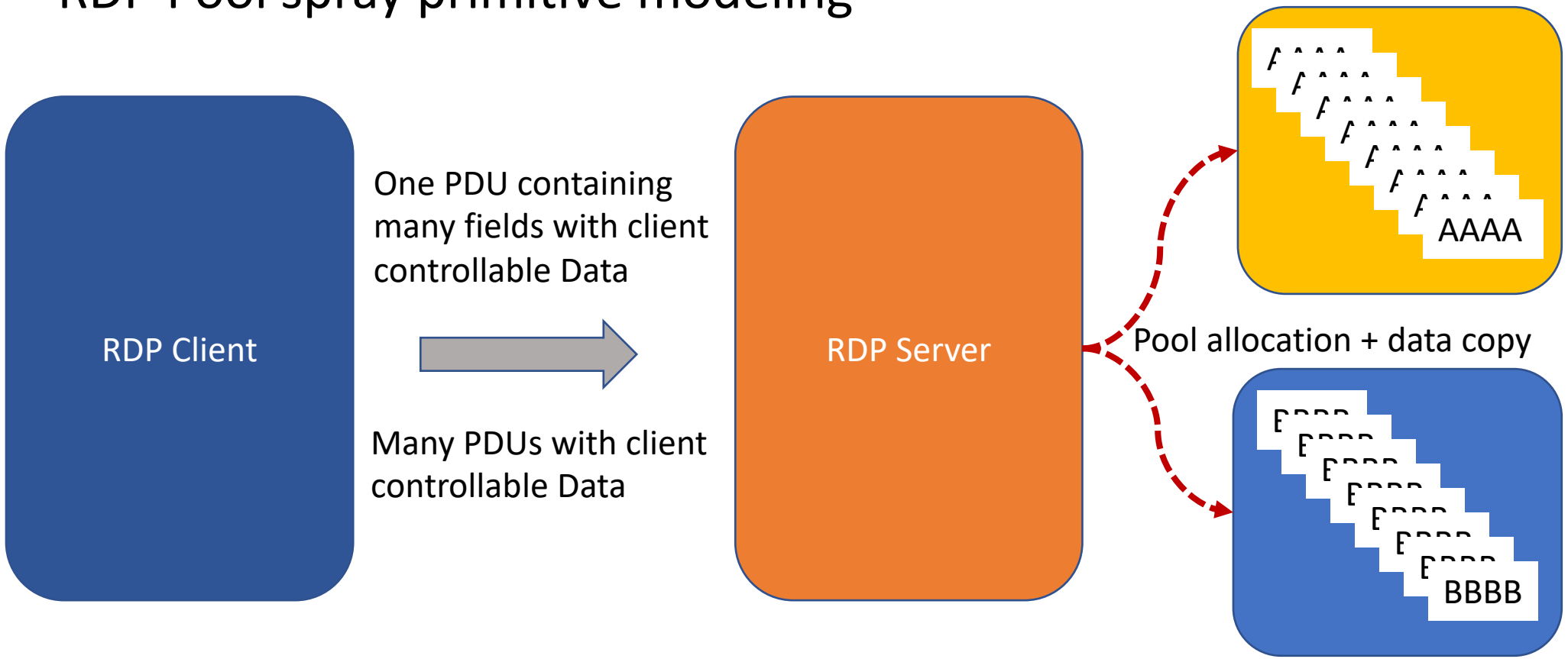
```
C:\Windows\system32>icacls c:\windows\system32\license.rtf
c:\windows\system32\license.rtf BUILTIN\Administrators:(F)
                                GA1OIS\test:(F)
                                NT AUTHORITY\SYSTEM:(F)
                                NT AUTHORITY\INTERACTIVE:(M,DC)
                                NT AUTHORITY\SERVICE:(M,DC)
                                NT AUTHORITY\BATCH:(M,DC)
                                NT AUTHORITY\SYSTEM:(I)(F)
                                BUILTIN\Administrators:(I)(F)
                                BUILTIN\Users:(I)(RX)
```

Last Words For ACL Overwritten Pattern

- An interesting story
  - The initial finding did NOT meet the Microsoft criteria for a patch.
  - Found a new attacking vector based on the initial finding and reported 5 new vulnerabilities.
  - MSRC confirmed new vulnerabilities and developing the patches.
  - Eligible for the bounty program from Microsoft.
  - Request for vulnerabilities disclosure in Black Hat but have to hold the details until Jan 2023 because Microsoft are fixing the design issue and need more time for the patch.
- Not limited to here…
  - Other hyper-v VM operations lead to more path redirection issues with our new magic.
  - There will be more similar issues if Microsoft will not fix the issue that the hyper-v virtual hard disk folder could be fully controllable by a standard user.
- A new attacking surface?
  - Path redirection never die under the new magic?

Pool Spray Pattern

- RDP Pool spray primitive modeling



RDP Pool Spray Primitive Pattern Extraction

- Workspace: RDP service
- Find data copy
  - Kernel pool allocation
    - ExAllocatePoolWithTag or its wrapper
  - Memory copy to the allocated kernel pool
    - Memcpy
    - Memory write in a loop
- Connect RDP PDU to data copy
  - ~~Xflow_src: RDPWD!WDLIB_MCSIcaRawInput~~
  - Xflow_src: RDPWD!SM_MCSSendDataCallBack
  - Xflow_sink: the data copy function

A typical PDU processing call stack

Virtual function call

```
RDPWD!SM_MCSSendDataCallback+0x175
RDPWD!HandleAllSendDataPDUs+0x115
RDPWD!RecognizeMCSFrame+0x32
RDPWD!MCSIcaRawInputWorker+0x3b4
RDPWD!WDLIB_MCSIcaRawInput+0x13
termdd!IcaRawInput+0x5a
tssecsrv!CRawInputDM::PassDataToServer+0x2b
tssecsrv!CFilter::FilterIncomingData+0xdd
tssecsrv!ScrRawInput+0x60
termdd!IcaRawInput+0x5a
tdtcp!TdInputThread+0x34d
termdd!_IcaDriverThread+0x53
termdd!_IcaStartInputThread+0x6c
termdd!IcaDeviceControlStack+0x629
termdd!IcaDeviceControl+0x59
termdd!IcaDispatch+0x13f
```

Pool Spray Pattern: Find Data Copy With Memcpy

- Code query construction – query mode

```
select * from workspace.rdp_service
where func has exallocatepoolwithtag and
func has memcpy and
flow_src is exallocatepoolwithtag.ret and
flow_sink is memcpy.arg1
```

- Memcpy result and confirmation

module	function
Termdd.sys	_IcaLoadSdWorker
	IcaChannelInputInternal
	_IcaPushStack
	_IcaTraceWrite
	_IcaOpenTraceFile

```
  v11 = ExAllocatePoolWithTag(NonPagedPool, a6 + 32, 0x63695354u);
if ( v11 )
{
  v32 = InputBuffer;
  v11[3] = v23;
  v11[4] = v23;
  v11[2] = v11 + 8;
  memcpy(v11 + 8, v32, v23);
```

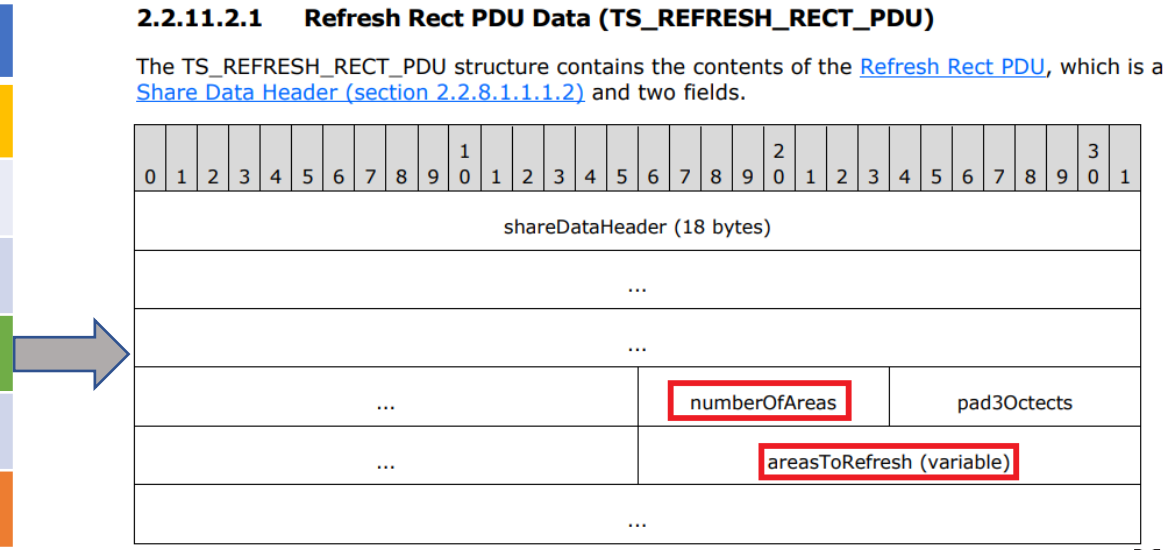
Connect RDP PDU To Data Copy With Memcpy

- Code query construction – query mode

```
select * from workspace.rdp_service
where xflow_src is SM_MCSSendDataCallback
and xflow_sink is IcaChannelInputInternal
```

- Find connection from RDP PDU to memcpy data copy #1

Stack	module	function
0	Termdd.sys	IcaChannelInputInternal
1	Termdd.sys	IcaChannelInput
2	RDPWD.sys	WDICART_IcaChannelInputEx
3	RDPWD.sys	WDW_InvalidateRect
4	RDPWD.sys	ShareClass::SC_OnDataReceived
5	RDPWD.sys	SM_MCSSendDataCallback



2.2.11.2.1 Refresh Rect PDU Data (TS_REFRESH_RECT_PDU)

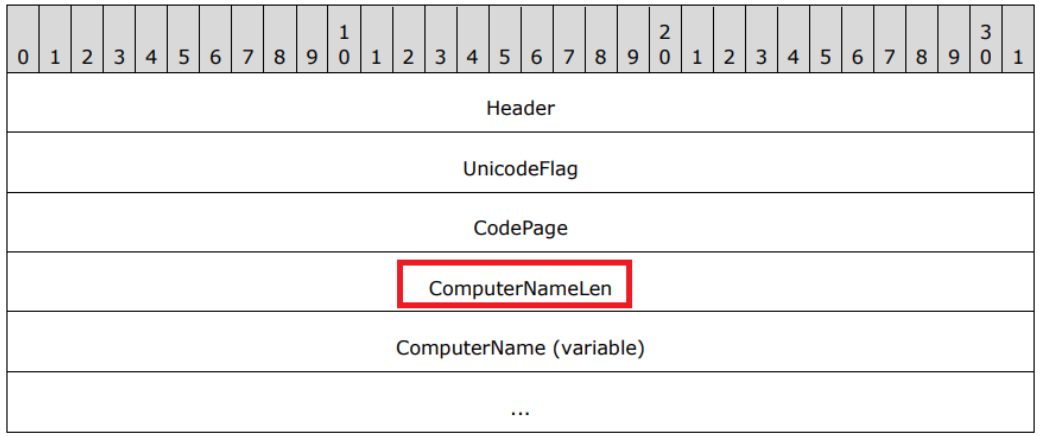
The TS_REFRESH_RECT_PDU structure contains the contents of the Refresh Rect PDU, which is a Share Data Header (section 2.2.8.1.1.1.2) and two fields.

Find Connection From RDP PDU To Memcpy Data Copy #2

Stack	module	function
0	Termdd.sys	IcaChannelInputInternal
1	Termdd.sys	IcaChannelInput
2	RDPWD.sys	WDICART_IcaChannelInputEx
3	RDPWD.sys	WDW_OnDataReceived
4	RDPWD.sys	SM_MCSSendDataCallback

2.2.2.4 Client Name Request (DR_CORE_CLIENT_NAME_REQ)

The client announces its machine name.



[MS-RDPEA]: Remote Desktop Protocol: Audio Output Virtual Channel Extension
[MS-RDPEAI]: Remote Desktop Protocol: Audio Input Redirection Virtual Channel Extension
[MS-RDPECLIP]: Remote Desktop Protocol: Clipboard Virtual Channel Extension
[MS-RDPEDC]: Remote Desktop Protocol: Desktop Composition Virtual Channel Extension
[MS-RDPEDYC]: Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension
[MS-RDPEECO]: Remote Desktop Protocol: Virtual Channel Echo Extension
[MS-RDPEFS]: Remote Desktop Protocol: File System Virtual Channel Extension
[MS-RDPEGDI]: Remote Desktop Protocol: Graphics Device Interface (GDI) Acceleration Extensions
[MS-RDPELE]: Remote Desktop Protocol: Licensing Extension
[MS-RDPEMC]: Remote Desktop Protocol: Multiparty Virtual Channel Extension
[MS-RDPEMT]: Remote Desktop Protocol: Multitransport Extension
[MS-RDPEPC]: Remote Desktop Protocol: Print Virtual Channel Extension
[MS-RDPEPNP]: Remote Desktop Protocol: Plug and Play Devices Virtual Channel Extension
[MS-RDPEPS]: Remote Desktop Protocol: Session Selection Extension
[MS-RDPERP]: Remote Desktop Protocol: Remote Programs Virtual Channel Extension
[MS-RDPESC]: Remote Desktop Protocol: Smart Card Virtual Channel Extension
[MS-RDPESP]: Remote Desktop Protocol: Serial and Parallel Port Virtual Channel Extension
[MS-RDPEUDP]: Remote Desktop Protocol: UDP Transport Extension
[MS-RDPEUSB]: Remote Desktop Protocol: USB Devices Virtual Channel Extension
[MS-RDPEV]: Remote Desktop Protocol: Video Redirection Virtual Channel Extension
[MS-RDPEVOR]: Remote Desktop Protocol: Video Optimized Remoting Virtual Channel Extension
[MS-RDPEXPS]: Remote Desktop Protocol: XML Paper Specification (XPS) Print Virtual Channel Exte
[MS-RDPNSC]: Remote Desktop Protocol: NSCodec Extension
[MS-RDPRFX]: Remote Desktop Protocol: RemoteFX Codec Extension
[MS-RDWR]: Remote Desktop Workspace Runtime Protocol Specification
[MS-RDPEGT]: Remote Desktop Protocol: Geometry Tracking Virtual Channel Protocol Extension
[MS-RDPEGFX]: Remote Desktop Protocol: Graphics Pipeline Extension
[MS-RDPEI]: Remote Desktop Protocol: Input Virtual Channel Extension
[MS-RDPADRV]: Remote Desktop Protocol: Audio Level and Drive Letter Persistence Virtual Channel

Pool Spray Pattern: Find Data Copy With Memory Write In Loop

- Code query construction – script mode

```
# start: ret of a target function call
src_expr = {
    CONDITIONS:[{EXPR:"expr.op", TARGET_VALUE:"call", OPERATOR:"eq"}, {EXPR:"expr.call", TARGET_VALUE:func_name, OPERATOR:"eq"}]
}
```

ExAllocatePoolWithTag and its wrapper

*(_DWORD *)(AllocatedKernelPool + index) = *(_DWORD *)(PDUData + index)

```
# end: assign op (aka mem write) in a loop
sink_expr = {
    CONDITIONS:[{EXPR:"expr.op", TARGET_VALUE:"asg", OPERATOR:"eq"}, {EXPR:"expr.op.type", TARGET_VALUE:"dword", OPERATOR:"eq"}],
    NESTED:{
        CONDITIONS:[{EXPR:"this.rhs.type", TARGET_VALUE:"dword", OPERATOR:"eq"}, {EXPR:"this.rhs.op", TARGET_VALUE:"ptr", OPERATOR:"eq"},
                    {EXPR:"this.lhs.type", TARGET_VALUE:"dword", OPERATOR:"eq"}, {EXPR:"this.lhs.op", TARGET_VALUE:"ptr", OPERATOR:"eq"}],
        NEXT:"this.lhs",
        NESTED:{
            CONDITIONS:[{EXPR:"this.lhs.op", TARGET_VALUE:"cast", OPERATOR:"eq"}, {EXPR:"this.lhs.type", TARGET_VALUE:"dword *", OPERATOR:"eq"}],
        }
    },
    IN_LOOP:True, # in this case, check if it's asg -> expr -> block -> do/while
    GET_VARS:"lhs",
}
```

Find Data Copy With Memory Write In Loop: Result and Confirmation

module	function
rdpwd.sys	ShareClass::SBC_DumpBitmapKeyDatabase
	ShareClass::SBC_HandlePersistentCacheList

```
bitmapCacheListPoolLen = 0xC * (totallen + 4);
*((_DWORD *)v4 + 0x553) = bitmapCacheListPoolLen;
bitmapCacheListPool = WDLIBRT_MemAlloc(bitmapCacheListPoolLen, 0x64775354u);
```

```
thisc = (struct TS_BITMAPCACHE_PERSISTENT_LIST *)((char *)TS_BITMAPCACHE_PERSISTENT_LIST_2
                                                  + 8 * key_index
                                                  + 0x2E);// get bitmap cache key address
do                                  // bitmapCacheListPool is in 0x522 offset
{
  *(_DWORD *)(0xC
            * (*(_DWORD *)(*((_DWORD *)v4 + 0x552) + index_2 + 0x18)
             + *(_DWORD *)(*((_DWORD *)v4 + 0x552) + index_2 + 4)
             + v25
             + 4)
            + *((_DWORD *)v4 + 0x552)) = *((_DWORD *)thisc - 1);// copy low 32-bits
  *(_DWORD *)(0xC
            * (v25
             + *(_DWORD *)(*((_DWORD *)v4 + 0x552) + index_2 + 0x18)
             + *(_DWORD *)(*((_DWORD *)v4 + 0x552) + index_2 + 4))
            + *((_DWORD *)v4 + 0x552)
            + 52) = *(_DWORD *)thisc;// copy high 32-bits
```

Copy keys DWORD by DWORD in a loop

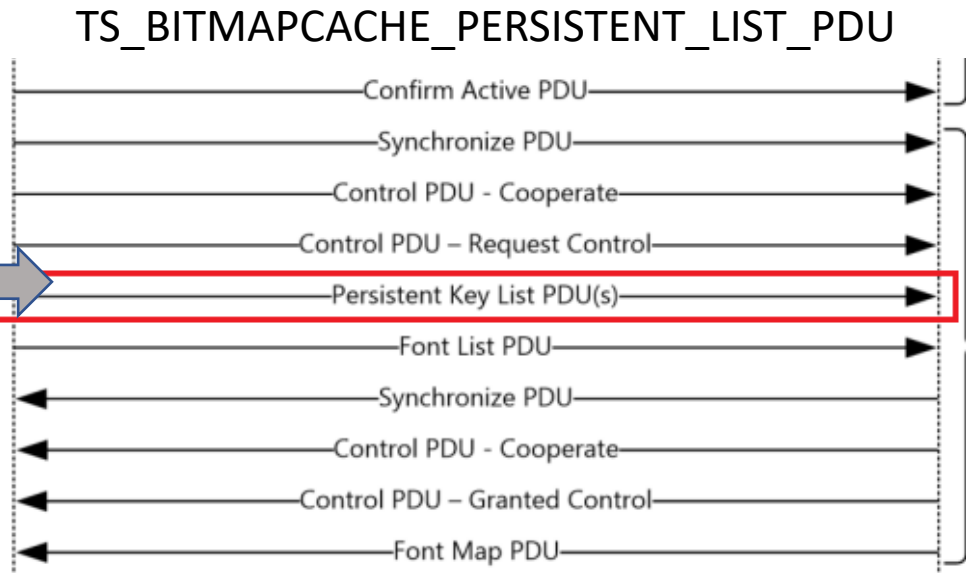
Connect RDP PDU To Data Copy With Memory Write In Loop

- Code query construction – query mode

```
select * from workspace.rdp_service
where xflow_src is SM_MCSSendDataCallback and
xflow_sink is ShareClass::SBC_HandlePersistentCacheList
```

- Find connection from RDP PDU to memory write in loop

Stack	module	function
0	RDPWD.sys	ShareClass::SBC_HandlePersistentCacheList
1	RDPWD.sys	ShareClass::SC_OnDataReceived
2	RDPWD.sys	SM_MCSSendDataCallback



black hat EUROPE 2022

Pool Spray Pattern: PoC - Demo

Refresh Rect PDU

```
8656e02c  e3ffc300 86c1002c 5c5c5c5c 5c5c5c5c     86c0102c  e3ffc300 86c1002c 5c5c5c5c 5c5c5c5c
8656f02c  e3ffc300 86c1002c 5c5c5c5c 5c5c5c5c     86c0202c  e3ffc300 86c1002c 5c5c5c5c 5c5c5c5c
8657002c  e3ffc300 86c1002c 5c5c5c5c 5c5c5c5c     86c0302c  e3ffc300 86c1002c 5c5c5c5c 5c5c5c5c
8657202c  e3ffc300 86c1002c 5c5c5c5c 5c5c5c5c     86c0402c  e3ffc300 86c1002c 5c5c5c5c 5c5c5c5c
8657302c  e3ffc300 86c1002c 5c5c5c5c 5c5c5c5c     86c0502c  e3ffc300 86c1002c 5c5c5c5c 5c5c5c5c
8657402c  e3ffc300 86c1002c 5c5c5c5c 5c5c5c5c     86c0602c  e3ffc300 86c1002c 5c5c5c5c 5c5c5c5c
8657802c  e3ffc300 86c1002c 5c5c5c5c 5c5c5c5c     86c0702c  e3ffc300 86c1002c 5c5c5c5c 5c5c5c5c
8658a02c  e3ffc300 86c1002c 5c5c5c5c 5c5c5c5c     86c0802c  e3ffc300 86c1002c 5c5c5c5c 5c5c5c5c
8658b02c  e3ffc300 86c1002c 5c5c5c5c 5c5c5c5c     86c0902c  e3ffc300 86c1002c 5c5c5c5c 5c5c5c5c
8658c02c  e3ffc300 86c1002c 5c5c5c5c 5c5c5c5c     86c0a02c  e3ffc300 86c1002c 5c5c5c5c 5c5c5c5c
8658d02c  e3ffc300 86c1002c 5c5c5c5c 5c5c5c5c     86c0b02c  e3ffc300 86c1002c 5c5c5c5c 5c5c5c5c
```

RDPDR Client Name Request PDU

```
86850838  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86850c38  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86851038  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86851438  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86851838  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86851c38  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86852038  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86852438  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86852838  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86852c38  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
```

Bitmap Cache PDU

```
b7e03894 41414141 41414141 00000003 41414141 41414141 00000004 41414141 41414141
b7e038b8 41414141 41414141 00000006 41414141 41414141 00000007 41414141 41414141
b7e038dc 41414141 41414141 00000009 41414141 41414141 0000000a 41414141 41414141
b7e03900 41414141 41414141 0000000c 41414141 41414141 0000000d 41414141 41414141
b7e03924 41414141 41414141 0000000f 41414141 41414141 00000010 41414141 41414141
b7e03948 41414141 41414141 00000012 41414141 41414141 00000013 41414141 41414141
b7e0396c 41414141 41414141 00000015 41414141 41414141 00000016 41414141 41414141
b7e03990 41414141 41414141 00000018 41414141 41414141 00000019 41414141 41414141
b7e039b4 41414141 41414141 0000001b 41414141 41414141 0000001c 41414141 41414141
```

Summary

- A new efficient and effective tool to search code patterns in binaries.
- Code pattern search not only work for memory corruption vulnerabilities, but also for logic vulnerabilities, new attack surfaces and even for exploitation primitives.
- The entire process of how to use code patterns to automatically find vulnerabilities in close source software.
  - Seed vulnerability
  - Vulnerability modeling
  - Vulnerability pattern extraction
  - Query/script construction
  - Result review and triage finding
  - PoC creation
- No silver bullet for bug hunting, deeply comprehending the target is still the core.

Links and References

- From Logic to Memory: Winning the Solitaire in Reparse Points
  - https://www.blackhat.com/eu-21/briefings/schedule/#from-logic-to-memory-winning-the-solitaire-in-reparse-points-24731
- Pool Feng Shui in Windows RDP Vulnerability Exploitation
  - https://github.com/ga1ois/BlueHat-2019-Seattle/blob/master/Pool%20Fengshui%20in%20Windows%20RDP%20Vulnerability%20Exploitation%20-%20submission.pdf
  - https://unit42.paloaltonetworks.com/exploitation-of-windows-cve-2019-0708-bluekeep-three-ways-to-write-data-into-the-kernel-with-rdp-pdu/
  - https://unit42.paloaltonetworks.com/cve-2019-0708-bluekeep/

Q & A