



# Inception: Virtual Space in Memory Space in Real Space - Memory Forensics of Immersive Virtual Reality with the HTC Vive

<sup>By</sup> Peter Casey, Rebecca Lindsay-Decusati, Ibrahim Baggili, & Frank Breitinger

> From the proceedings of The Digital Forensic Research Conference **DFRWS 2019 USA** Portland, OR (July 15th - 19th)

DFRWS is dedicated to the sharing of knowledge and ideas about digital forensics research. Ever since it organized the first open workshop devoted to digital forensics in 2001, DFRWS continues to bring academics and practitioners together in an informal environment. As a non-profit, volunteer organization, DFRWS sponsors technical working groups, annual conferences and challenges to help drive the direction of research and development.

https://dfrws.org

# Inception: Virtual Space in Memory Space in Real Space - Memory Forensics of Immersive Virtual Reality with the HTC Vive

Peter Casey, Rebecca Lindsay-Decusati, Ibrahim Baggili, & Frank Breitinger

Graduate Research Assistant & UNHcFREG Member Presenting @ DFWRS 2019, Portland, OR, USA



University of New Haven Cyber Forensics Research & Education Group





## Acknowledgment



 This material is based upon work supported by the National Science Foundation under Grant No. 1748950. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

# Background – Attacks against VR





Identify Security and Privacy Issues

Conduct Forensic Analysis

Develop Educational Material

- Overlay Attack
- Chaperone Attack
- Disorientation Attack
- Human Joystick
- Camera Exfiltration

# Background – Forensics



- Surveyed nonvolatile artifacts
- Analyzed network traffic
- Steam (VR) and Oculus (Home)
- Select social VR applications
- Goal
  - Establish knowledge base of artifact location and content
  - Recreate a timeline of events

## Problem



- Substantial amount of information resides only in memory.
  - Tracked device location / state
  - Virtual environment (VE) geometry and orientation
- What if a crime is committed in VR or a user is harmed while immersed?
- Chaperone, Disorientation, HJ attacks manipulate the live copy of the VE
- Can we recreate the VE from a memory dump?

# Contributions



- To the best of our knowledge this is the primary account for specifically examining the memory forensics of VR systems.
- We share our analysis and findings that may impact future investigations involving VR systems.
- We employ and share a reusable methodology that may be adopted by others to create similar plugins.
- We construct an open source tool Vivedump, that may be used in the analysis of memory dumps of HTC Vive VR systems and share related datasets. The tool is a plug-in for the widely adopted Volatility framework.

# **Objective - Apparatus**



- Can we recreate the VE from a memory dump?
- HTC Vive
  - 2 x Controllers
  - 2 x Base stations
- Windows 10 Workstation
  - 8/16 GB Ram
  - Steam VR
  - Dumplt

# Methodology - Recon



- OpenVR not quite open source...
- But we do have the function interfaces.
- Step 1: Review documentation and header files for relevant data sources. Table 3: OpenVR Data Structures and Enumerators

Data	Contents
TrackedDevicePose_t	Pose and status of tracked device
${ m HmdMatrix34\_t}$	Tracked device tranformation matrix
${ m ETrackedDeviceClass}$	Type of Device
ETrackedDeviceProperty	Static device properties
EVRState	Status of the overall system
$\mathbf{EVREventType}$	Event types
${ m EDeviceActivityLevel}$	Level of Hmd activity
$VREvent\_Notification\_t$	Notification related events
$VREvent_Overlay_t$	Overlay Events
$VREvent\_Ipd\_t$	Ipd change

# Challenges – Plan



- Memory forensics: The path forward (Case and Richard, 2017)
- Future directions
  - Application specific analysis
  - Rapid updates 17 versions of SteamVR in June
- Complex and robust system, solely RE effort not practical.

# Methodology – Scenario Creation



- How do we know what we are searching for in the first place?
- OpenVR Background application

	Listing 1 : TrackedDeviePose_t
1	struct TrackedDevicePose_t
2	{
3	HmdMatrix34_t mDeviceToAbsoluteTracking;
4	$HmdVector3_t$ vVelocity;
5	$HmdVector3_t vAngularVelocity;$
6	${ m ETrackingResult}$ eTrackingResult;
7	bool bPoseIsValid;
8	<b>bool</b> bDeviceIsConnected;
9	};

/** enum values to pass in to VR_Init † * draw a 3D scene. */	to identify whether t
enum EVRApplicationType	
/ {	
<pre>VRApplication_Other = 0, //</pre>	Some other kind of a
<pre>VRApplication_Scene = 1, //</pre>	Application will sub
<pre>VRApplication_Overlay = 2, //</pre>	Application only int
<pre>VRApplication_Background = 3, //</pre>	Application should n
	keep it running if e
<pre>VRApplication_Utility = 4, //</pre>	Init should not try
	interfaces (like IVR
<pre>VRApplication_VRMonitor = 5, //</pre>	Reserved for vrmonit
<pre>VRApplication_SteamWatchdog = 6,//</pre>	Reserved for Steam
<pre>VRApplication_Bootstrapper = 7, //</pre>	Start up SteamVR
VRApplication_Max	
};	

# Methodology – Memory Scanning



- Tracking system information is rapidly updated and has high precision.
- Cheat Engine "Cheat Engine is an open source tool designed to help you with modifying single player games..."
  - Powerful Memory Scanner
  - Extensive Lua scripting support





# Methodology – Memory Scanning

<ul> <li>Search</li> </ul>	
----------------------------	--

- Reduce
- Modify
- Repeat

	310	na ici neo (sourc	e(repostoper		com		·9…	-	0		-
.000000		0.000001:	0.105654	,	2:	0.064373,	3:	0.183311		^	
.000000	,	0.0000001:	0.105629	,	2:	0.064330,	3:	0.183289			
.000000	,	0.0000001:	0.105671	,	2:	0.064368,	3:	0.183240			
.000000	,	0.0000001:	0.105691	ر	2:	0.064348,	3:	0.183208			
.000000	,	0.0000001:	0.105754	ر	2:	0.064326,	3:	0.183273			
		0.0000001:									
.000000	,	0.0000001:	0.105831	,	2:	0.064416,	3:	0.183292			
.000000		0.0000001:	0.105803		2:	0.064393,	3:	0.183298			
.000000		0.0000001:	0.105745		2:	0.064393,	3:	0.183240			
.000000		0.0000001:	0.105722		2:	0.064381,	3:	0.183139			
.000000		0.0000001:	0.105743		2:	0.064365,	3:	0.183122			
.000000		0.0000001:	0.105680		2:	0.064391,	3:	0.183096			
.000000		0.0000001:	0.105645		2:	0.064378,	3:	0.183084			
.000000		0.0000001:	0.105705		2:	0.064409,	3:	0.183104			1
.000000		0.0000001:	0.105677		2:	0.064339,	3:	0.183177			ŀ
.000000		0.0000001:	0.105686		2:	0.064373,	3:	0.183245			
.000000		0.0000001:	0.105682		2:	0.064344,	3:	0.183317			
.000000		0.0000001:	0.105742		2:	0.064396,	3:	0.183295			
		0.000001:									
.000000		0.0000001:	0.105850		2:	0.064360,	3:	0.183208			
.000000		0.0000001:	0.105826		2:	0.064368,	3:	0.183172			
.000000		0.0000001:	0.105824		2:	0.064412,	3:	0.183137			E
.000000		0.0000001:	0.105801		2:	0.064459,	3:	0.183146			4
.000000		0.0000001:	0.105729		2:	0.064493,	3:	0.183114			
.000000		0.0000001:	0.105725		2:	0.064434,	3:	0.183139			
.000000		0.0000001:	0.105738		2:	0.064371,	3:	0.183156			
		0.000001:									
.000000		0.000001:	0.105689		2:	0.064397,	3:	0.183276			

Found: 75					R
Address	Value	Previous ^	New So	an Next Scan	Undo Scan
6F5F29E4	0.106781	0.106781		Value:	Setting
6F5F2FF0	0.105239	0.105239		0.106	
6F5F3220	0.105209	0.105209			
6F5F3EC0	0.105499	0.105499	Scan Type	Exact Value	✓ Not
6F5F4220	0.106948	0.106948	Value Type	Float	O Rounded (default)
6F5F45D0	0.106094	0.106094	Memor	Scan Options	Rounded (extreme)
235D9375778	0.105584	0.106114		•	O Truncated
7FFC991A9884	0.106978	0.106978	Start	000000	Simple values only
7FFC991A9888	0.106934	0.106934	Stop	7ffff	fffffffff Unrandomizer
7FFC991A988C	0.106889	0.106889	🗹 Writa	ble	Executable Enable Speedhack
7FFC991A9890	0.106844	0.106844	Сору	OnWrite	
7FFC991A9894	0.106800	0.106800	✓ Fast S	can 4 Alignme	ent
7FFC991A9898	0.106755	0.106755	✓ Fast 5	can 4 🖂 Last Digi	its
788000170900	0 106711	> 106711	Pause	the game while scanni	ing
Memory view	/		0		Add Address Manually
Active Description	1	Address	Туре	Value	
HMD x_cool	rd	235D9375778	Float	0.1055841446	^

# Methodology – Static Reference



- How to reliably locate data in a memory dump?
- Work backwards from data to a static reference.
- Two strategies:
  - Attach debugger and monitor reads and writes to address
  - Search for pointer chains.

Base Address	Offset 0	Offset 1	Offset 2	Offset 3	Offset 4	Points to:
'vrmonitor.exe"+001F78A0	68					235D9375778
'THREADSTACK1"-00000	2C0	0	50	8	68	235D9375778
'THREADSTACK1"-00000	2C0	0	50	8	68	235D9375778
'THREADSTACK1"-00000	2C0	0	50	8	68	235D9375778
'vrclient_x64.dll"+00220F	60	30	50	8	68	235D9375778
'vrclient_x64.dll"+002278	38	90	50	8	68	235D9375778
'vrmonitor.exe"+001F7798	668	90	50	8	68	235D9375778
'vrclient_x64.dll"+00227B	18	C0	50	8	68	235D9375778
'vrclient_x64.dll"+00227B	20	C0	50	8	68	235D9375778
'vrclient_x64.dll"+00227B	18	C0	50	8	68	235D9375778
'vrclient_x64.dll"+002221	D8	C0	50	8	68	235D9375778
'THREADSTACK1"-00000	428	C0	50	8	68	235D9375778
'vrclient_x64.dll"+002250	18	C0	50	8	68	235D9375778
'vrclient_x64.dll"+00223C	618	C0	50	8	68	235D9375778
'vrclient_x64.dll"+002250	20	C0	50	8	68	235D9375778
'vrclient_x64.dll"+00223C	620	C0	50	8	68	235D9375778
'vrclient_x64.dll"+002250	28	C0	50	8	68	235D9375778
'vrclient_x64.dll"+00223C	628	C0	50	8	68	235D9375778
'vrclient_x64.dll"+002250	30	C0	50	8	68	235D9375778
'vrclient_x64.dll"+00223C	630	C0	50	8	68	235D9375778
	20	<u></u>	50	0	<u>20</u>	22500275770

Inception: Virtual Space in Memory Space in Real Space -- Memory

# Methodology – Static Reference



- References originating from the stack, unreliable.
- DLL functions operated on data passed to them
- Most reliable candidate pointer chains originate directly from executable – vrmonitor.exe

	]	List	ing	2:	Di	sass	semb	oly of Base Address Access
1	48	8b	05	25	d6	10	00	MOV RAX, $[RIP+0x10d625]$
2	48	85	c0					TEST RAX, RAX
3	75	2a						JNZ + 2a
4	sni	р						

# Methodology – Yara Signatures



- YARA "is a tool aimed at (but not limited to) helping malware researchers to identify and classify malware samples"
- Develop signatures based on referencing op-codes.
- Volatility Plugin & malware analysis Thomas Chopitea

```
YARA_HMD = {
    'hmd_pointer': 'rule hmd_pointer { strings: $p = {48 8b 05 25 D6 10 00} condition: $p}',
}
YARA_HMD_ACTIVITY = {
    'hmd_activity': 'rule hmd_activity { strings: $p = {48 8b 05 55 A4 13 00} condition: $p}',
}
```

# Plugin Development



Algorithm 2 Vivedump: Locate target data	
1: $y \leftarrow yara.compile(rule)$	$\triangleright$ Base address signature
2: $r \leftarrow [offset1, offset2,]$	$\triangleright$ Route
3: for $task \in processList$ do	
4: <b>if</b> $task = vrmonitor.exe$ <b>then</b>	
5: for $vad \in getProcessVad$ do	
6: $m \leftarrow y.match(vad)$	$\triangleright$ Address of YARA match
7: for $offset \in r$ do	$\triangleright$ Traverse route
8: $m \leftarrow dereference(m) + offset$	
9: end for	
0: return $m$	$\triangleright$ Address of target data
1: end for	
2: end if	
13: end for	

#### • Optimization – limit search to related process

#### Data Interpretation



- C++ Standard "Non-static data members of a (non-union) class with the same access control are allocated so that later members have higher addresses within a class object."
- Enumerators if not explicitly set, assume default implementation

	Listing 1 : TrackedDeviePose_t		
$\frac{1}{2}$	struct TrackedDevicePose_t	enum ETrackingResult	
$     \begin{array}{c}       2 \\       3 \\       4 \\       5 \\       6 \\       7 \\       8 \\       9     \end{array} $	<pre>{     HmdMatrix34_t mDeviceToAbsoluteTracking;     HmdVector3_t vVelocity;     HmdVector3_t vAngularVelocity;     ETrackingResult eTrackingResult;     bool bPoseIsValid;     bool bDeviceIsConnected; };</pre>	<pre>{     TrackingResult_Uninitialized     TrackingResult_Calibrating_InProgress     TrackingResult_Calibrating_OutOfRange     TrackingResult_Running_OK     TrackingResult_Running_OutOfRange }; </pre>	= 1, = 10 = 10 = 20 = 20

#### Visualization

Mindows PowerShell

PS C:\Users\strat\Source\Repos\volatility>



k

-

σ

#### Data Reliability



- TrackedDevicePose array always available
- TrackedDevice Class route may be misleading
- Correlate device states between structures

Data Structure	YARA Signature	Route
TrackedDevicePose Array	48 8b 05 25 D6 10 00	0x10D62C
—- HmdMatrix34_t		+ 0x5C
—- ETrackingResult		+ 0xA4
—- bPoseIsValid		+ 0xA8
bDeviceIsConnected		+ 0xAA
HMD State	48 8b 05 55 A4 13 00	0x13A45C, 0x1E0, 0x 0B8, 0x68
—- HMD Activity		+ 0x10
TrackedDevice Class	$48 \ 8b \ 05 \ 55 \ A4 \ 13 \ 00$	0x13A45C, 0x190, 0x8, 0x510, 0x3E0, 0x120

## Future Work



- Expand supported recoverable features
- Version Detection
- Application update recovery
- Emulation

#### Questions



#### Peter Casey (pgrom1@unh.newhaven.edu)

#### Thanks to the National Science Foundation, University of New Haven, DFRWS, and MITRE.