Programmatic Security

Anti Reversing For Developers

By Totally_Not_A_Haxxer via SkyPenguinLabs

Section 0x0

Whoami?

- 17 yr old Automotive/IoT/Systems Cybersecurity researcher
- Author of BHGM, BHPM, GHFM
- Content Creator for GuidedHacking & the Safer Internet Project
- Mass contributor / speaker / presenter / educator
- Developer behind the SkyLine programming language, the Aries network suite (custom network protocols), and REplay
- Reverse Engineering / Web[API] security / Exploit Development / Software engineering & PCB maker (sometimes)
- Security Analyst For The DrGreenNFT project
- A kid who has done too much
- I love tearing down systems by the root of their design...
- Written over 300+ articles
- Published over 200+ different tools and automation tools/frameworks on various platforms
- ex Game Hacker turned Good boy

@Totally_Not_A_Haxxer

Section 0x1

What is on the agenda?

- Introduction to the space - what does securing binaries look like?
- Analyzing the landscape of reverse engineers and what they target
- What goes into protecting binaries? And what does it mean to protect them?
- Why should you protect binaries?
- What is the difference between Windows and Linux applications?
- Anti-analysis, Anti-Debug, Dynamic Protections, Anti-Injection, etc.
- How mainstream applications are protected
- Major takeaways
- Thank you SIP!
- Concluding & Ending

Meet Rovax!

- Will be throughout the presentation
- Speaks in PigPen
- Is a galactic voyager from planet 3-C
- Quiet yet extremely loud

Rovax

Section 0x2

What does securing an application look like??

- Securing Apps...
  - Obfuscation
    - A. Binary Packing
    - B. String / Data encryption
  - Anti-analysis
    - A. Process Blacklisting
    - B. Process Whitelisting
    - C. File system scanning
    - D. Thread Hiding
    - Anti-Debug
      - A. Using System APIs
      - B. Checking specific portions of the programs memory
    - Anti-VM
    - Anti-tamper
  - Secure memory handling
  - Secure communications

Section 0x3

Why Should I Care?

- Prevent people from stealing your software easily
- Prevent people from obtaining information they should not have access to
- Prevent ease of exploitation (which can make systems all over vulnerable!)
- Saves you a ton in the future in resources, time and money
- Saves you from a nasty reputation
- Saves users from worrying about the safety of production servers/systems

Caring makes you...

Caring gives you...

- A more experienced developer
- A chance to look from a new perspective
- A better developer
- A chance to test your own applications
- A more proficient problem solver
- A chance to connect with a different world!

A wild FoPwn appearing to steal creds from your apps...

Timestamp - 2:55:19

https://youtu.be/yRdMGwBy8Ok

100% of reverse engineers do not care about your scopes or obfuscation

secure development != obfuscation

~ Haxxer #egocheck

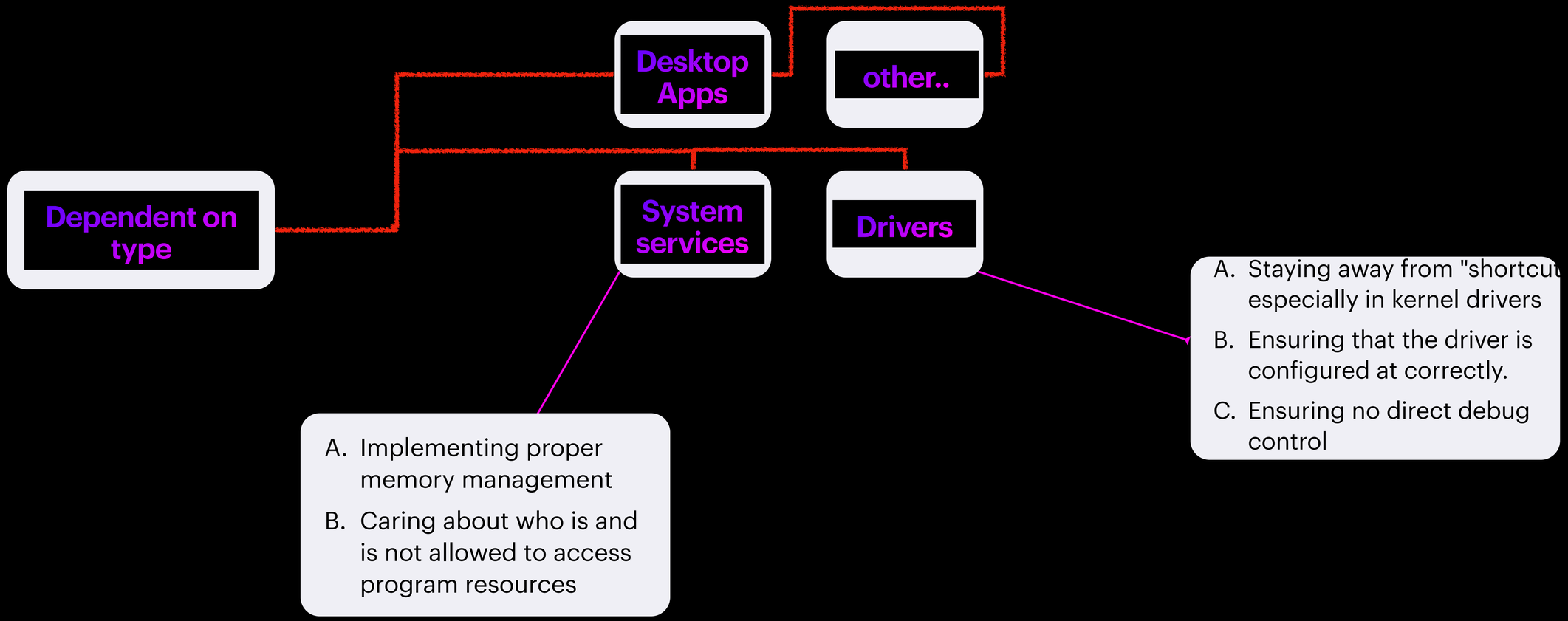
Section 0x4

What the heck do reverse engineers want with me?

- Secret keys or administrative licenses
- To crack or steal your software
- To take advantage of flaws in your software
- To build universal interfaces for your software
- To create copies of your software
- To bypass specific restrictions or find ways around restrictions
- To discover other portions of your infrastructure (I have done this once before on a mass research operation)
- and so much more...

Section 0x5

What does it mean to protect binaries?



Protecting the application means doing whatever possible to ensure the upmost safety for your binary applications

Section 0x5

What goes into binary defense?

Most common forms of protections

Anti-Debug

Methods

Prevents easy information gathering

- A. System APIs
- B. Checking PEB
- C. Checking process flags
- D. Checking hardware breakpoints
- E. Checking software breakpoints
- F. Remote DBG checking
- G. etc..

- A. Checking window names
- B. Checking window class names
- C. Checking for specific file processes
- D. Checking for specific configuratio files or environment variables
- E. Checking and verifying specific programs

Anti-Application

Methods

Prevents apps like IDA from running adjacent to the app

Anti-VM

[Virtual Machine]

Methods

Prevents analysis in a VM

- A. Client side binary hashing
- B. Client side integrity checking
- C. Memory scanning and validation
- D. Static pre-load checks
- E. .text section validation

- A. Checking file system relics
- B. Checking network interfaces or network services
- C. Scanning for specific system services or configuration
- D. Checking for specific hardware identifiers

Anti-Mod

[Anti modification/ tamper]

Methods

Prevents modification or binary patching

Section 0x6

Differences in binary protections for different systems (Windows, Linux, etc) and compilers (gcc, VC++, etc)

What you are securing

What are you trying to secure? (e.g: sensitive information, hardcoded credentials [stupid anyway], data transport, etc.)

How you will secure it

Anti...(debugging, analysis, virtualization, etc)

How will you implement it

Using a system API? From scratch? A known security provider?

Ensuring reliability

Is the implementation of the system easy to bypass or simple? Or is it overly complex and too dangerous? Is it resourceful?

System compatibility

Is the security system cross platform or system / architecture specific?

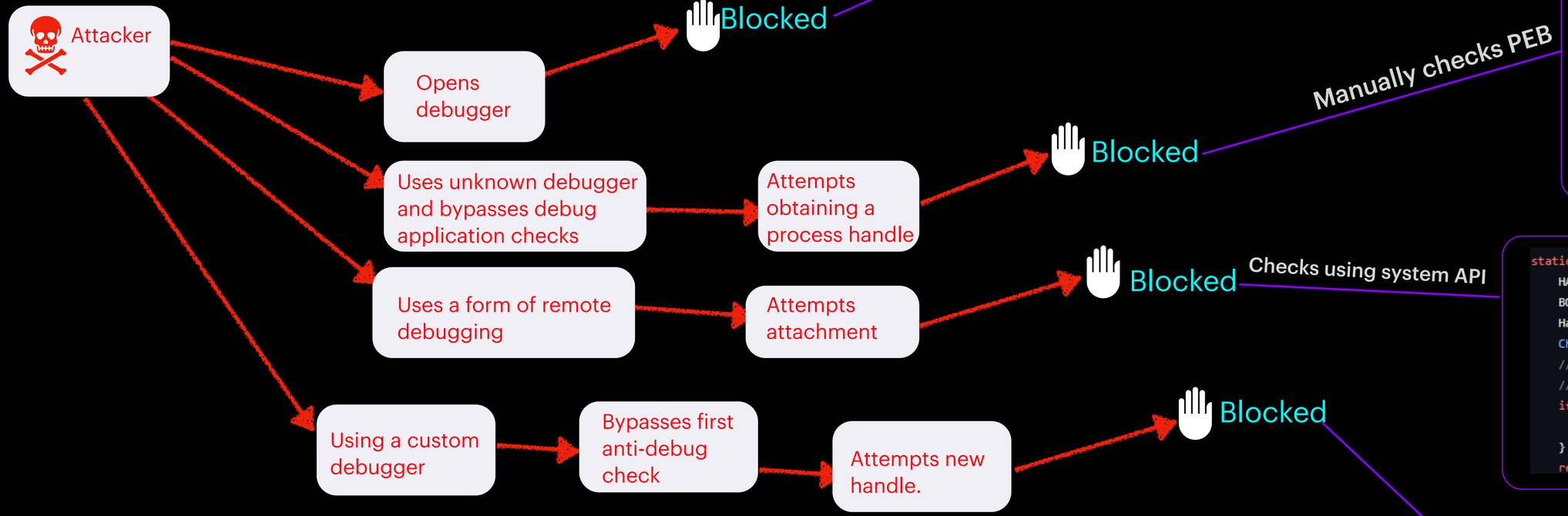
C++ or not, securing applications has a lot of dynamic parts

Before we go to the methods- lets recap...

Up Next: Methods & Their Inner Workings In C++/Windows

Anti-Debug

What does it look like?




```
#include <Windows.h>

void somefunction() {
    if IsDebuggerPresent() {
        AlertFakeCrash()
    } else {
        loop()
    }
}
```

WinAPI Call

```
static bool SkyBin_AntiDebugger_VerifyPEB() {
#if defined (ENV64BIT)
    PPEB pbeb = (PPEB)__readgsqword(0x60);
#elif defined (ENV32BIT)
    PPEB pbeb = (PPEB)__readfsdword(0x30);
#endif
    // modification for framework
    if (pbeb->BeingDebugged == 1) {
        exit(0);
    }
    // return anyway
    return pbeb->BeingDebugged == 1;
}
```

```
static bool SkyBinWin_RemoteDebuggerIsPresent_Check() {
    HANDLE Handler = INVALID_HANDLE_VALUE;
    BOOL IsActive = FALSE; // Is remote debugger preser
    Handler = GetCurrentProcess();
    CheckRemoteDebuggerPresent(Handler, &IsActive);
    //Note: If you are modifying this code, you can ret
    // But due to worker configuration, I decided to ac
    if (IsActive) {
        exit(0);
    }
    return IsActive;
```

```
bool SetSecurityControls() {
    HANDLE TkHandle = nullptr;
    PTOKEN_USER UserTk = nullptr;
    PACL ProcACL = nullptr;
    DWORD CbBuffer = 0;
    DWORD CbACL = 0;
    bool bSuccess = false;
    if (!OpenProcessToken(
        GetCurrentProcess(),
```

Anti-Analysis

What does it look like?

Attacker

Tries running the software in a VM

Runs static analysis tools like IDA to

Tries to analyze the environment interaction

Uses applications like HTTPanalyzer or wireshark

Tries to dump the memory using task manager

Blocked — Scans the system for VM Relics

```
VOID vbox_reg_keys()
{
    /* Array of strings of blacklisted registry keys */
    const TCHAR* szKeys[] = {
        _T("HARDWARE\\ACPI\\DSDT\\VBOX__"),
        _T("HARDWARE\\ACPI\\FADT\\VBOX__"),
        _T("HARDWARE\\ACPI\\RSDT\\VBOX__"),
        _T("SOFTWARE\\Oracle\\VirtualBox Guest Additions"),
        _T("SYSTEM\\ControlSet001\\Services\\VBoxGuest"),
        _T("SYSTEM\\ControlSet001\\Services\\VBoxMouse"),
        _T("SYSTEM\\ControlSet001\\Services\\VBoxService"),
        _T("SYSTEM\\ControlSet001\\Services\\VBoxSF"),
        _T("SYSTEM\\ControlSet001\\Services\\VBoxVideo")
    };

    WORD dwlength = sizeof(szKeys) / sizeof(szKeys[0]);
```

VM Relics

Temporary block — Obfuscation is in place

```
::vector<std::string> BlackListed_Applications = {
    _xor_("ollydbg.exe"),
    _xor_("windbg.exe"),
    _xor_("idaq.exe"),
    _xor_("idaq64.exe"),
    _xor_("ImmunityDebugger.exe"),
```

Blocked — Environment analysis tools are not allowed

Blocked — Is In The Blacklisted App List

```
    _xor_("API Monitor.exe"),
    _xor_("Process Explorer.exe"),
    _xor_("Resource Hacker.exe"),
    _xor_("PEview.exe"),
    _xor_("SysinternalsSuite.exe"),
    _xor_("Regshot.exe"),
    _xor_("Wireshark.exe"),
    _xor_("DumpIt.exe"),
```

Sysinternals / monitoring tools are blacklisted

Memory dump works

Hacker attempts to analyze dump

Dump analysis fails

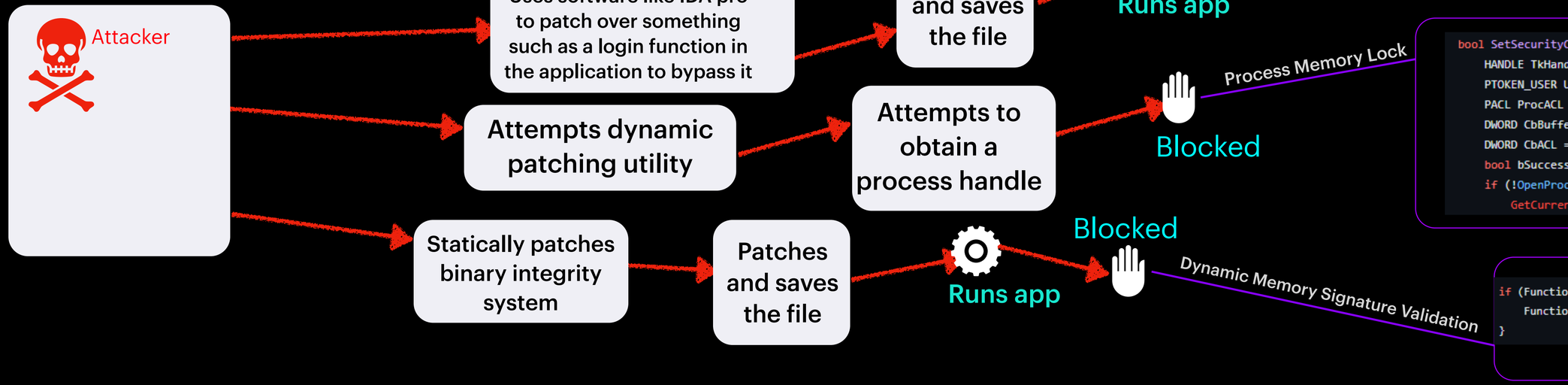
Anti dumping — Kamikaze Files

```
VOID ErasePEHeaderFromMemory()
{
    _tprintf(_T("[*] Erasing PE header from memory\n"));
    DWORD OldProtect = 0;

    // Get base address of module
    char *pBaseAddr = (char*)GetModuleHandle(NULL);

    // Change memory protection
    VirtualProtect(pBaseAddr, 4096, // Assume x86 page size
        PAGE_READWRITE, &OldProtect);

    // Erase the header
    SecureZeroMemory(pBaseAddr, 4096);
}
```

https://github.com/LordNoteworthy/al-khaser/blob/master/al-khaser/AntiDump/ErasePEHeaderFromMemory.cpp

Anti-Tampering

What does it look like?




```
void somefunction() {
    threadroutine() {
        if BinarySig() != MATCH_SIG {
            Fakereport();
        } else {
            continue
        }
        wait(5, time::seconds)
    }()
}
```

```
bool SetSecurityControls() {
    HANDLE TkHandle = nullptr;
    PTOKEN_USER UserTk = nullptr;
    PACL ProcACL = nullptr;
    DWORD CbBuffer = 0;
    DWORD CbACL = 0;
    bool bSuccess = false;
    if (!OpenProcessToken(
        GetCurrentProcess(),
```

```
if (Function_Address == NULL) {
    Function_Address = FindPattern(PBYTE("\x48\x89\x74\x24\
}
```

Anti-Application

Inner Workings

Blocked

Attempt to send kill signal

Person A (maybe a blue teamer)

Opens IDA Pro and a debugger

Goes to grab some energy drinks or something

Meanwhile..

Sends process to processmng

Adds it to the process list

Apps running in the background

IDA.exe

App1

App2

App3

App4

Super secure app

```
static bool SkyBin_Framework_CheckWindowName(std::string Name) {
    HANDLE Window = NULL;
    bool found = FALSE;
    std::wstring_convert<std::codecvt_utf8<wchar_t>, wchar_t> converter;
    std::wstring wideString = converter.from_bytes(Name);
    const wchar_t* WindowName = wideString.c_str();
    Window = FindWindow(NULL, WindowName);
    if (Window) {
        found = TRUE;
    }
    return found;
}
```

WARN: FOUND IDA.exe

"APPLICATION NOT ALLOWED"

Runs a function for each process name

["ida.exe"]

Application starts threaded routine to scan for all currently running processes

SEND USER ERROR

SEND TRIGGER

App 5 with intense anti-application systems

How are mainstream applications protected?

Some resources!

A. https://github.com/LordNoteworthy/al-khaser/tree/master

B. Anti-Debug with Structured Exception Handling + Trap Flag (https://www.youtube.com/watch?v=ww2INI76ydQ&t=1s)

C. Time Based Anti-Debug Techniques (https://www.youtube.com/watch?v=sirFxSNSXDY&t=2s)

D. Windows API (https://learn.microsoft.com/en-us/windows/win32/apiindex/windows-api-list)

Major Takeaways

- Secure now, not later
- Security promotes good development practices
- More security = more trust in product
- Most importantly- why not?
- Keep up on how hackers can bypass your security systems

Resources for securing C++ applications

A. https://github.com/LordNoteworthy/al-khaser

B. https://github.com/orgs/KeyAuth/repositories

C. https://www.youtube.com/watch?v=ww2INI76ydQ (guided hacking, anti-debug with structured exception handling + trap flag

D. https://www.youtube.com/watch?v=ww2INI76ydQ (guided hacking, anti debug techniques)

E. and more...

Thank you SIP!

- Good to be back!
- Will be doing more presentations in the future
- most importantly- join with this link!