

how i
met your

pointer

Hijacking client software for fuzz and profit



OVERVIEW

- Introduction.
- Fuzzing 101. *yawn*
- **The need for a different approach.**
 - Abusing the client.
- **A possible implementation. Boyka.**
- **EXPERIMENT.**
- Conclusion.



\$ WHOAMI

PARTICLE PHYSICIST



BORN NERD



PROSPECTIVE EMPLOYERS



FK THAT SH*T**



\$ WHOAMI



LET'S BE CLEAR

THIS WORK IS SHIT

LET'S BE CLEAR

**THIS WORK IS ~~NOT~~
"ESSENTIALLY FLAWED"**

WHAT THIS IS ABOUT

- Interesting approach to software testing
- Touching things you are not supposed to
- Breaking stuff (if you're lucky!)
- Multiple references to pop culture
 - and chocolate!

Everybody
Luvs
Chocolate

Look for the
ChocoQuiz icon



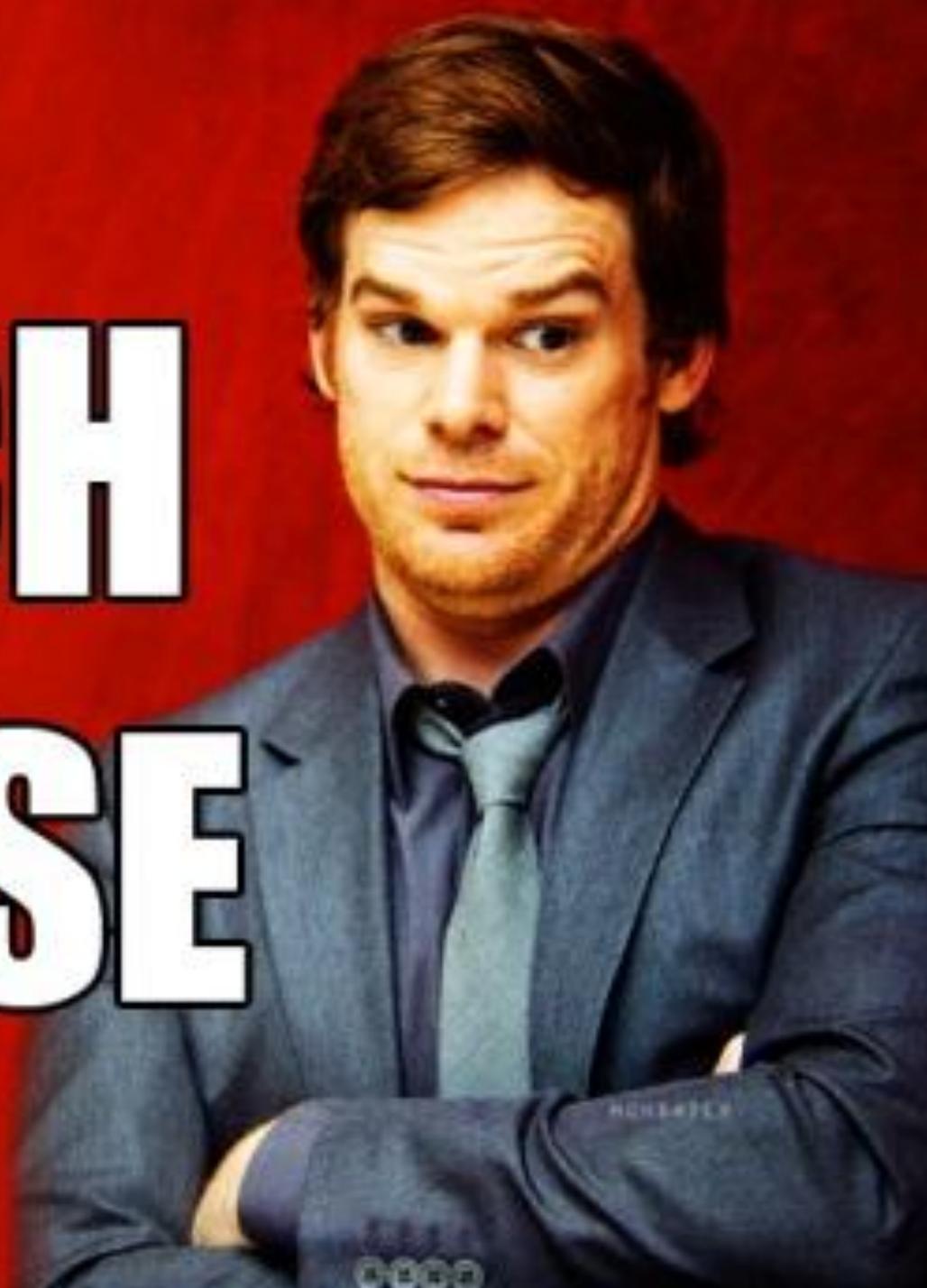
DISCLAIMER

FOR EDUCATIONAL PURPOSES
ONLY...



Fuzzing 101

**BITCH
PLEASE**



**Fuzzing is like violence:
if it doesn't solve your problems,
you are not using enough of it.**

A close-up, low-angle shot of a muscular man in a gym. He is shirtless, has a beard, and is looking directly at the camera with a serious expression. His right hand is raised to his chin. The background is a dimly lit gym with several bright spotlights creating a hazy, atmospheric effect. The text "BRUTE FORCE" is overlaid in a large, bold, yellow, italicized font across the center of the image.

BRUTE FORCE

MONSTER CONFIGURATION

```
1 from sulley import *
2
3 s_initialize("user")
4 s_static("USER")
5 s_delim(" ")
6 s_string("yomama")
7 s_static("\r\n")
8
9 s_initialize("pass")
10 s_static("PASS")
11 s_delim(" ")
12 s_string("issofat")
13 s_static("\r\n")
14
15 s_initialize("cwd")
16 s_static("CWD")
17 s_delim(" ")
18 s_string("c: ")
19 s_static("\r\n")
20
21 s_initialize("dele")
22 s_static("DELE")
23 s_delim(" ")
24 s_string("\\test.txt")
25 s_static("\r\n")
```

```
from sulley import *
from requests import ftp # this is our ftp.py file

sess = sessions.session(session_filename="audits/freefloatftp.session")
target = sessions.target("192.168.1.11", 21)
target.netmon = pedrpc.client("192.168.1.11", 26001) # NetMonitor (packets)
target.procmon = pedrpc.client("192.168.1.11", 26002) # ProcMonitor (crashes)
target.procmon_options = { "proc_name" : "FTPServer.exe" }

sess.add_target(target)

sess.connect(s_get("user"))
sess.connect(s_get("user"), s_get("pass"))

sess.connect(s_get("pass"), s_get("cwd"))
sess.connect(s_get("pass"), s_get("dele"))
sess.connect(s_get("pass"), s_get("mdtm"))
sess.connect(s_get("pass"), s_get("mkd"))

sess.fuzz()
```

ftp.py - protocol

ftp_session.py



And Now



I Wait

CRASH! BOOM! BANG! HAHA!

Windows taskbar and application windows are visible. The active window is **Sulley Fuzz Control**.

Sulley Fuzz Control

Total: 6 of 6,738 [] 0.089%
user: 6 of 1,123 [] 0.534%

Pause

Test Case #	Crash Synopsis
000006	[(INVALID):20202020 Unable to disassemble at 20202020 from thread 472 caused access violation

```
DOS Prompt - c:\Python25\python.exe ftp_session.py
File Edit View Help
DOS Prompt DOS Prompt - c:\Pyth...
[02:10.06] xmitting: [1.3]
[02:10.07] netmon captured 285731 bytes for test case #3
[02:10.08] fuzzing 4 of 1123
[02:10.08] xmitting: [1.4]
[02:10.09] netmon captured 287515 bytes for test case #4
[02:10.09] fuzzing 5 of 1123
[02:10.10] xmitting: [1.5]
[02:10.11] netmon captured 284250 bytes for test case #5
[02:10.11] fuzzing 6 of 1123
[02:10.12] xmitting: [1.6]
[02:10.13] netmon captured 284274 bytes for test case #6
[02:10.13] procmon detected access violation on test case #6
[02:10.13] primitive lacks a name, type: delim, default value:
[02:10.13] [(INVALID):20202020 Unable to disassemble at 20202020 from thread 472 c
ess violation
[02:10.13] restarting target process
[02:10.22] fuzzing 7 of 1123
[02:10.28] failed connecting on socket
Exception caught: error(10061, 'Connection refus
Restarting target and trying again
[02:10.28] restarting target process
[02:10.41] failed connecting on socket
Exception caught: error(10061, 'Connection ref
Restarting target and trying again
[02:10.41] restarting target process
[02:10.55] failed connecting on socket
Exception caught: error(10061, 'Connection
Restarting target and trying again
[02:10.55] restarting target process
Ready
```



PRECISE CRASH INFORMATION

```
http://localhost:26000/view_crash/6 +
[INVALID]:20202020 Unable to disassemble at 20202020 from thread 472 caused access violation
when attempting to read from 0x20202020

CONTEXT DUMP
EIP: 20202020 Unable to disassemble at 20202020
EAX: 00000216 (      534) -> N/A
EBX: 00000002 (        2) -> N/A
ECX: 0014d3c0 (  1364928) -> F unt authority\systemzÅ (heap)
EDX: 7c90e514 (2089870612) -> N/A
EDI: 003b19d5 (  3873237) -> (heap)
ESI: 0040a44e (  4236366) -> N/A
EBP: 003b1298 (  3871384) -> N/A
ESP: 00b2fc2c ( 11729964) ->
+00: 20202020 ( 538976288) -> N/A
+04: 20202020 ( 538976288) -> N/A
+08: 20202020 ( 538976288) -> N/A
+0c: 20202020 ( 538976288) -> N/A
+10: 20202020 ( 538976288) -> N/A
+14: 20202020 ( 538976288) -> N/A

disasm around:
    0x20202020 Unable to disassemble

SEH unwind:
    ffffffff -> kernel32.dll:7c839ad8 push ebp
```



THERE'S ALWAYS A BUT



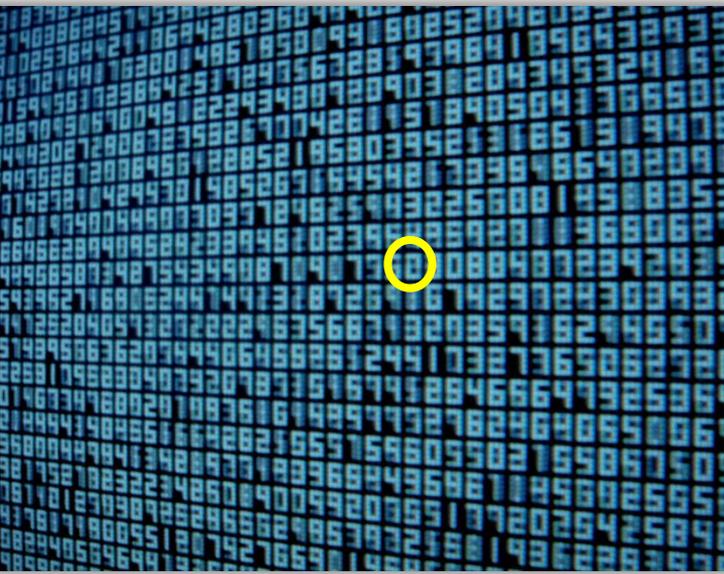
That's all very nice.

**But what if I *don't*
know the protocol?**

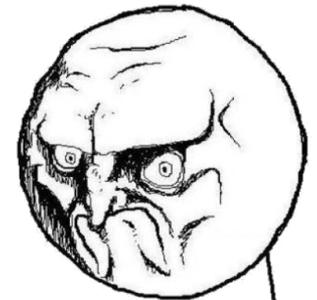


**There is NO
documentation at all.
:(**

I CAN ALWAYS TRY **DUMB** FUZZING!



REJECTED



NO.

THINK ABOUT CHECKSUMS...

Packet

Data

Checksum

Checksum = SHA1(Data)
SHA1: 160 bits
 $P(\text{right}) = 1/2^{160} \approx 1/10^{48}$
 $10^{48} = 1\text{k} \cdot 1\text{T} \cdot 1\text{T} \cdot 1\text{T} \cdot 1\text{T} \cdot 1\text{T}$

Dumb Fuzzing...





F*CK!



IS EVERYTHING LOST ?

**FAIL
HARDER**

**THINK
WRONG**

**The need for
a *different*
approach**



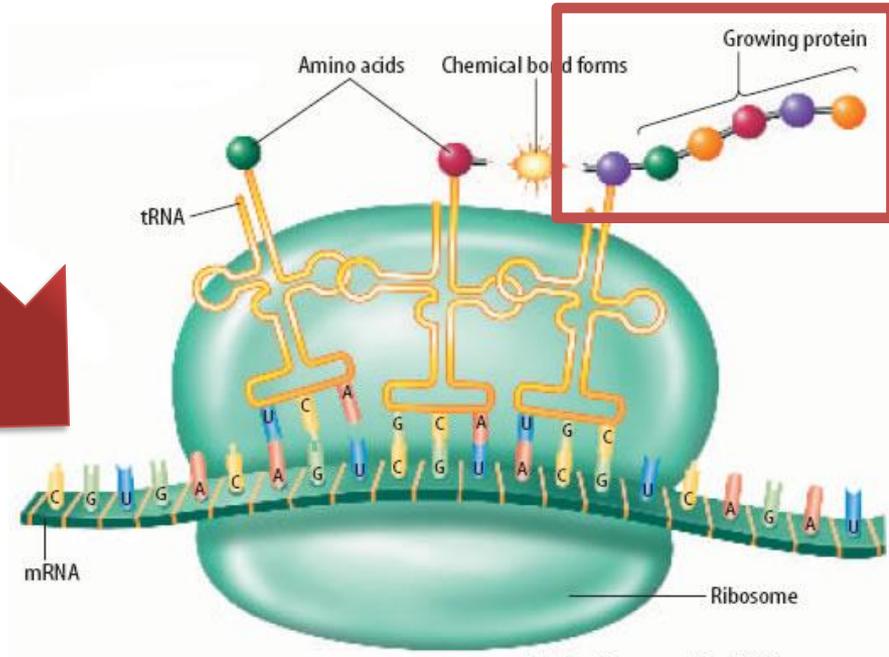
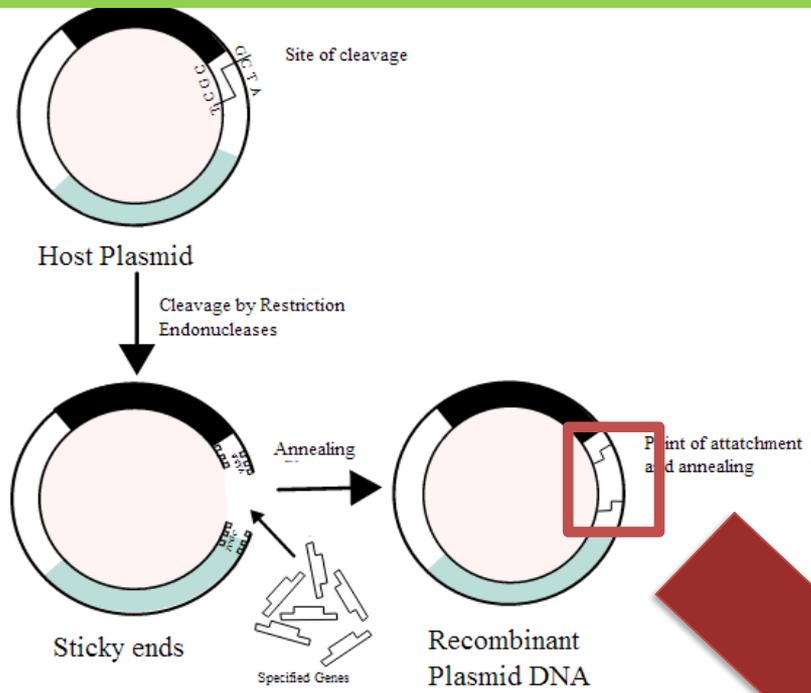
THEN ALONG CAME *my wife*



- **Biochemist**
- Works doing protein... something
- I suspect she really works doing...



BIOTECH WILL SAVE THE WORLD OR KILL US ALL

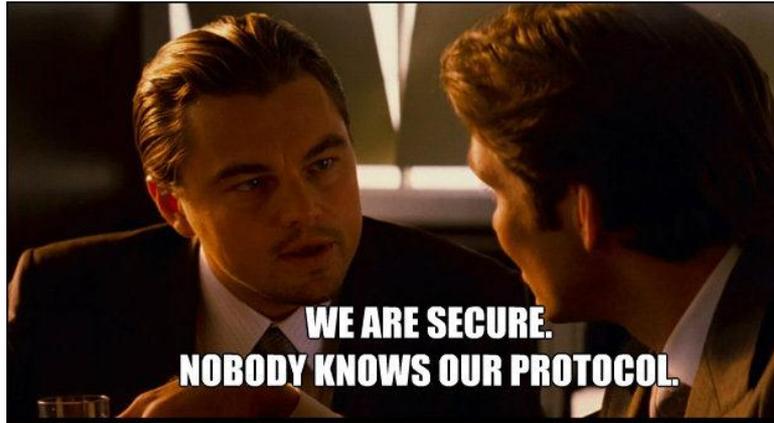


At the ribosome, the RNA's message is translated into a specific protein.





In a nutshell...





IDA View-A Hex View-A Structures

Graph overview

```
Line 2370 of 8806
```

```
fstp qword ptr [esi+58h]
jmp short loc_40DB9C
```

```
loc_40DB9A:
fstp st
```

```
float_arithmetic:
fld qword ptr [esi+58h]
fstcw [esp+110h+var_FE]
movzx eax, [esp+110h+var_FE]
or eax, 0C00h
mov dword ptr [esp+110h+var_FC], eax
fldcw word ptr [esp+110h+var_FC]
fistp [esp+110h+var_FC]
mov edx, dword ptr [esp+110h+var_FC]
fldcw [esp+110h+var_FE]
cmp edx, 0FFFFFFFh
jnb short loc_40DB9A
```

```
fst qword ptr [esi+50h]
```

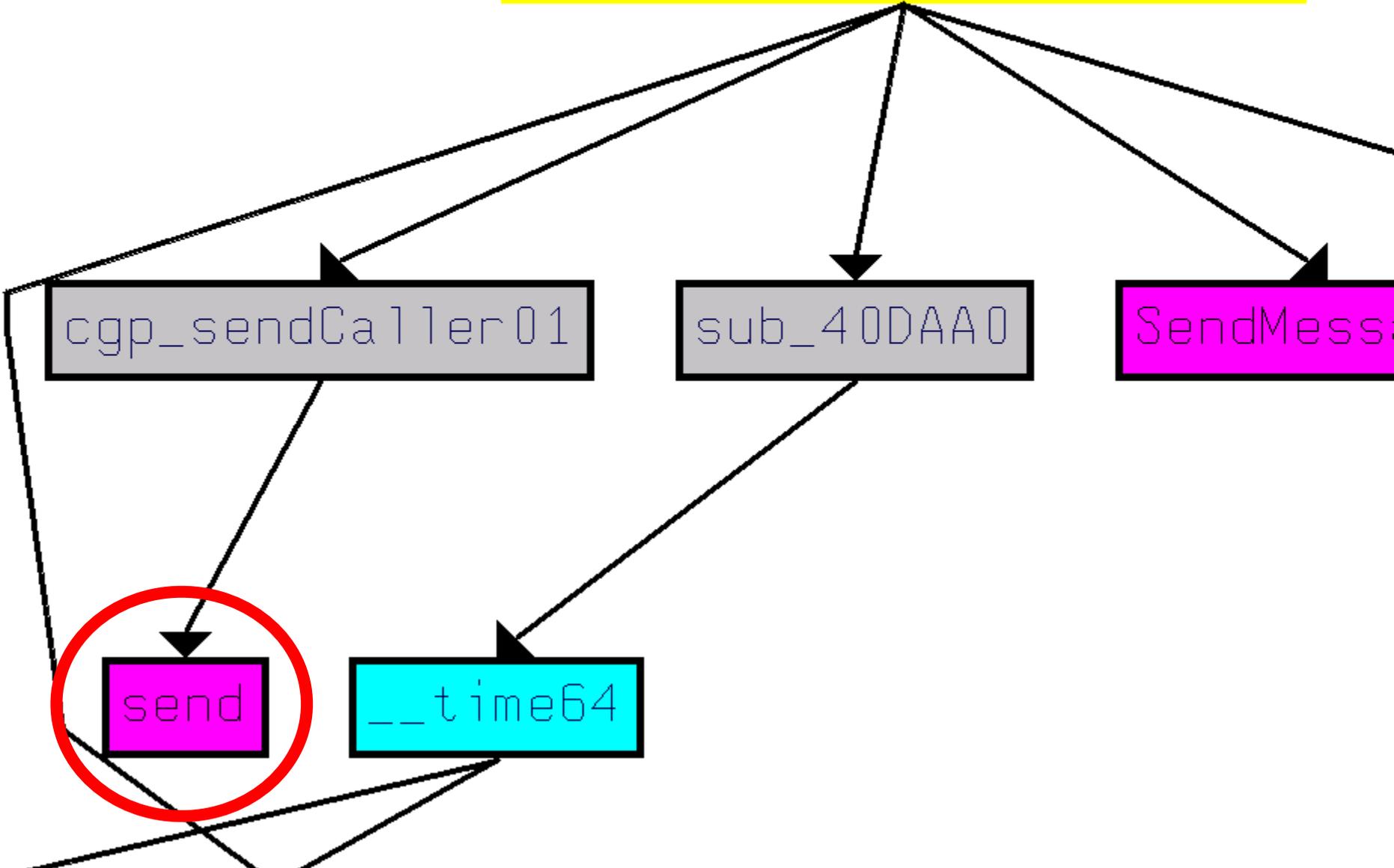
```
fld [esp+110h+len]
mov [esi+4Ch], edi
fstcw [esp+110h+var_FE]
fmul ds:dword_5B32E8
movzx eax, [esp+110h+var_FE]
or eax, 0C00h
mov dword ptr [esp+110h+var_FC], eax
fadd qword ptr [esi+50h]
fst qword ptr [esi+50h]
fldcw word ptr [esp+110h+var_FC]
fistp [esp+110h+var_FC]
mov ecx, dword ptr [esp+110h+var_FC]
fldcw [esp+110h+var_FE]
fldz
cmp ecx, 0FFFFFFFh
jnb short float_arithmetic
```

100.00% (240,901) (212,592) 0000CFBA|0040DBBA: cgp ArithmeticSender01:loc 40DBBA

Output window

MILF initialized

cgp_ArithmeticSender01



SIMPLE ARGUMENTS

```
; int __stdcall cgp_ArithmeticSender01(char *buf, int len, int)
cgp_ArithmeticSender01 proc near

var_FE= word ptr -0FEh
var_FC= qword ptr -0FCh
Dest= dword ptr -0F4h
var_4= dword ptr -4
buf= dword ptr 4
len= dword ptr 8
arg_8= dword ptr 0Ch

sub     esp, 100h
mov     eax, stackCookie
xor     eax, esp
mov     [esp+100h+var_4], eax
push   ebp
mov     ebp, [esp+104h+buf]
```



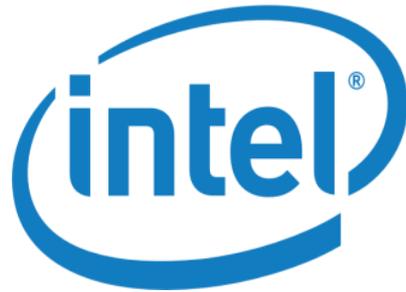
A photograph of two men in a bar setting. The man on the left is wearing a dark suit and tie, looking upwards with a surprised expression. The man on the right is wearing a grey t-shirt with 'WESLEYAN' printed on it, looking upwards with his mouth wide open in a shout or cheer, and his right arm is raised. The background is dark with some colorful lights.

IT GETS EXCITING

Detours

= userland hooking

= amazing stuff



= dynamic binary
instrumentation

= AWESOME stuff !!!



MICROSOFT DETOURS

- **Library for intercepting arbitrary Win32 binary functions.**
- Interception code is applied dynamically **at runtime.**
- **Replaces the first few instructions** of the target function
- with an **unconditional jump** to the detour function.
- **Replace or extend** the target function.



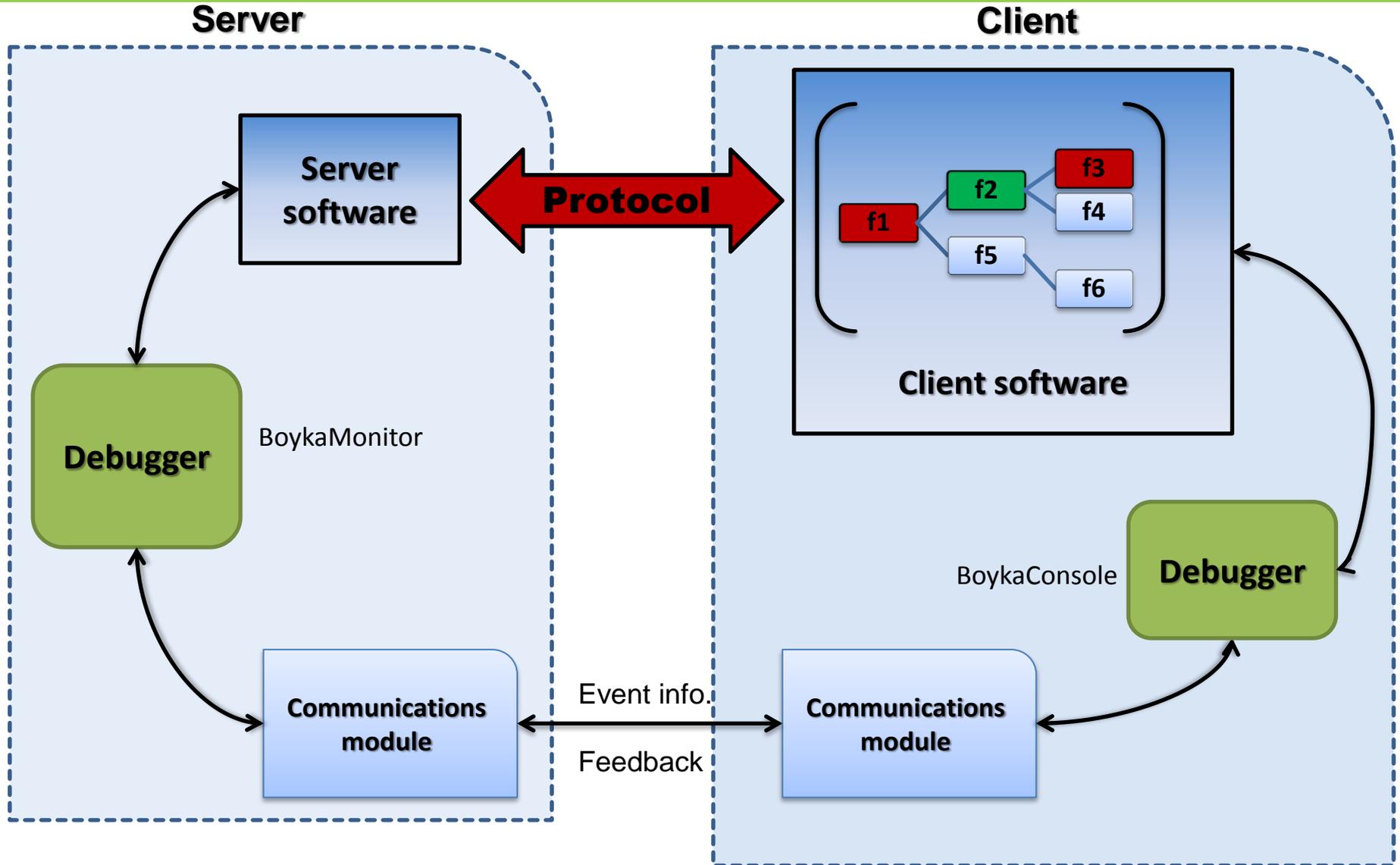
LONG STORY SHORT...



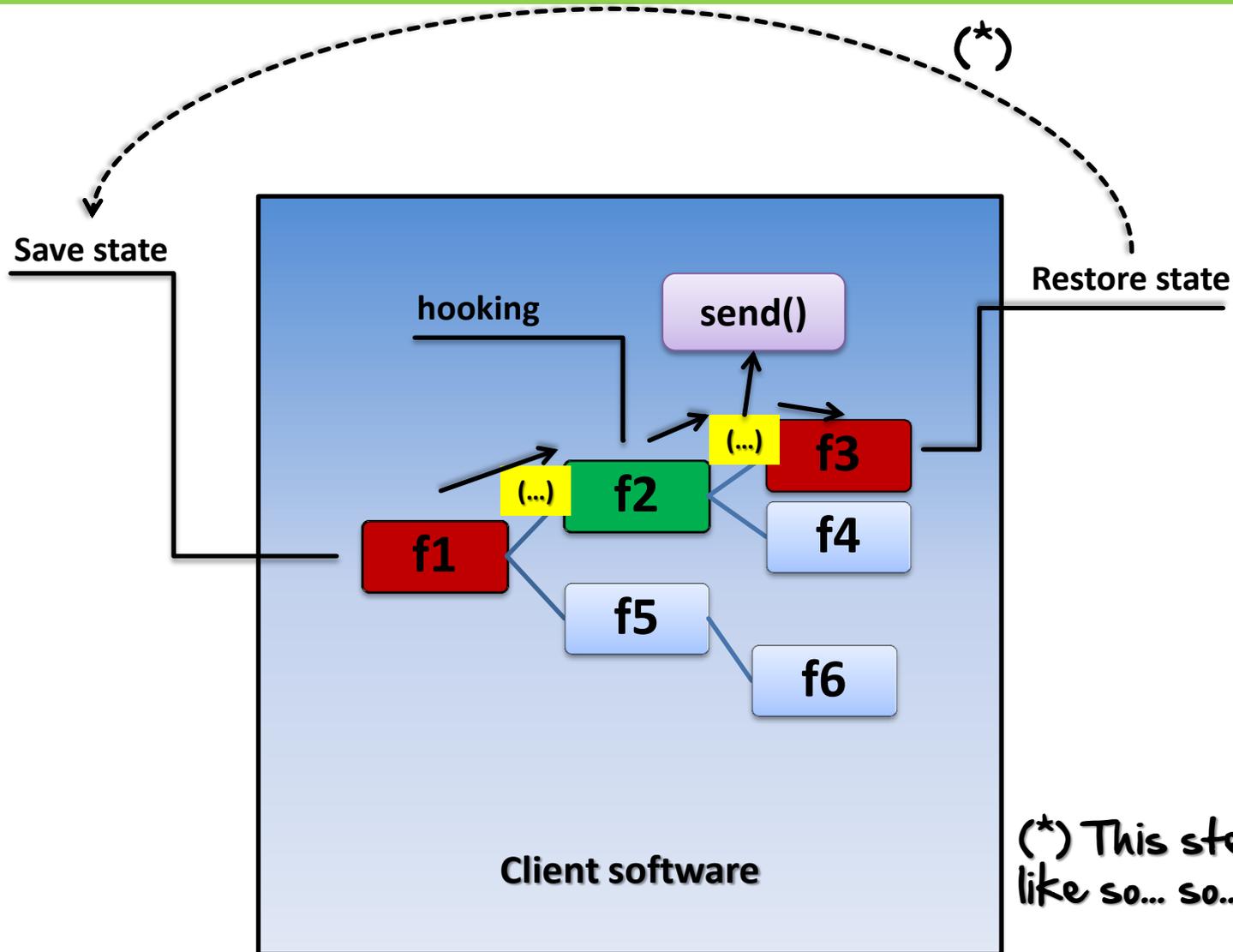


PLUMBING TIME

OVERVIEW (FROM A MILLION MILES AWAY)



OVERVIEW (FROM A THOUSAND MILES AWAY)



NOT SURE IF THIS MAKES SENSE



**OR I'M STARTING TO BELIEVE MY OWN
BULLSHIT**



THE CHALLENGE

- I can "inject" some data into the server
- By hijacking client execution at certain points
- ...
- ... aha...

- **Which. Points. Do. I. Use. ?!?!?!?**

Carlos,
help me!

There are
TOO MANY
Addresses
!!!!!!!





Anyone getting dizzy?

**BRACE YOURSELVES,
SHAMELESS
AUTOPROMOTION IS
COMING.**



M*LF & PIN TRACER

beta



IDASCOPE

ArithLog Rating: Exclude Zeroing

Basic Blocks size: Looped Blocks only

Allowed calls: Group by Functions

28 blocks from a total of 13292 blocks matched with the above settings.

	Address ▾	Name	Block Address	# Instr	Arithmetic/Logic Rating	▲
13	0x433c80	CRC32	0x433f20	8	75.00	
14	0x433c80	CRC32	0x433cc0	116	71.55	
15	0x433c80	CRC32	0x433ed0	19	68.42	
16	0x428791	XorChainEncrypt	0x4287a5	6	50.00	
17	0x4286bb	Base64Decode	0x428767	9	33.33	
18	0x428520	Rc4	0x42854b	19	36.84	
19	0x427b37	DecryptString	0x427b4d	12	33.33	
20	0x427b01	StringEncrypt_...	0x427b16	9	33.33	
21	0x42633c	MersenneTwister	0x4263a2	14	57.14	
22	0x42633c	MersenneTwister	0x426355	14	57.14	
23	0x426307	MersenneTwist...	0x426315	11	54.55	▼



<http://pnx-tf.blogspot.com/>

DIFFERENTIAL DEBUGGING

- Hook every function -> log hits.
- 1st run. Exercise as many functionality as possible.
- 2nd run. Focus on the interesting feature.
- Compare both -> filter out.

Function_1	Function_1
GUI_stuff	GUI_stuff
Windows_stuff	Windows_stuff
Function_2	Login_stuff
Thread_sync	Thread_sync
Function_3	Encryption_stuff
[...]	[...]



DIFFERENTIAL DEBUGGING

- Hook every function -> log hits.
- 1st run. Exercise as many functionality as possible.
- 2nd run. Focus on the interesting feature.
- Compare both -> filter out.

Function_1
GUI_stuff
Windows_stuff
Function_2
Thread_sync
Function_3
[...]

Function_1
GUI_stuff
Windows_stuff
Login_stuff
Thread_sync
Encryption_stuff
[...]

Login_stuff
Encryption_stuff
[...]



A man with a beard and safety glasses, wearing a green work shirt, is working in a workshop. He is using a green welding torch to weld a large, complex metal structure. The structure consists of several large, dark metal pipes or tubes arranged in a vertical stack, with various metal brackets and supports. Sparks are flying from the welding point, and a plume of white smoke or steam is rising from the top of the structure. The background shows a workshop with wooden beams and various tools and equipment.

BUILD YOUR WEAPON

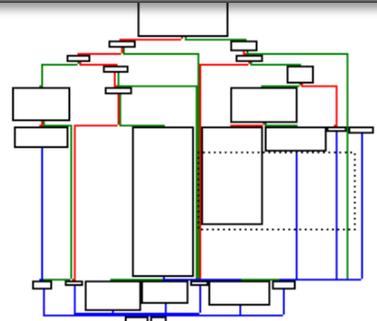
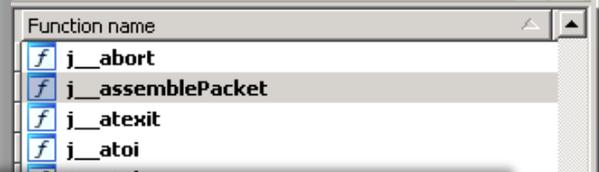
EPIC ASS KICKING



WAKE UP!

You're gonna miss
the good stuff!!!

FINDING POSSIBLE WEAK SPOTS



```
call ds:__imp_GlobalAlloc@8 ; GlobalAlloc(x,x)
cmp esi, esp
call j__RTC_CheckEsp
mov [ebp+login], eax
mov eax, [ebp+len]
add eax, 1
mov esi, esp
push eax ; cchMax
mov ecx, [ebp+login]
push ecx ; lpString
push 3E8h ; nIDDlgItem
mov edx, [ebp+hwnd]
push edx ; hDlg
call ds:__imp_GetDlgItemTextA@16 ; GetDlgItemTextA(x,x,x,x)
cmp esi, esp
call j__RTC_CheckEsp
mov ecx, [ebp+len]
push eax ; len
mov ecx, [ebp+szPacket]
push ecx ; szPacket
mov edx, [ebp+login]
push edx ; login
call j__assemblePacket
add esp, 0Ch
mov [ebp+szEncPacket], eax
mov eax, [ebp+szEncPacket]
push eax ; encryptedString
call j__sendTheShit
add esp, 4
mov esi, esp
mov eax, [ebp+login]
push eax ; hMem
call ds:__imp_GlobalFree@4 ; GlobalFree(x)
cmp esi, esp
call j__RTC_CheckEsp
mov eax, [ebp+szPacket]
push eax ; pUserData
call j__free
```

FINDING POSSIBLE WEAK SPOTS

IDA - S:\Software\c00

File Edit Jump Search View Debugger Options Windows Help

Functions window

- f j__abort
- f j__assemblePacket
- f j__atexit
- f j__atoi
- f j__atol
- f j__encryptLogin
- f j__exit
- f j__fclose
- f j__fflush
- f j__fputc
- f j__free
- f j__get_crtdouble_arg
- f j__get_int64_arg
- f j__get_int_arg
- f j__get_short_arg
- f j__is_wctype
- f j__isleadbyte
- f j__iswalnum
- f j__iswalpha
- f j__iswascii
- f j__iswcntrl

Line 1656 of 1906

Graph overview

```
; Attributes: bp-based frame
; char *__cdecl assemblePacket(char *login, char *szPacket, unsigned int len)
__assemblePacket proc near
var_F8= byte ptr -0F8h
szSecretString= byte ptr -34h
szPktLen= byte ptr -20h
delimiter= dword ptr -0Ch
var_4= dword ptr -4
login= dword ptr 8
szPacket= dword ptr 0Ch
len= dword ptr 10h

push    ebp
mov     ebp, esp
sub     esp, 0F8h
push    ebx
push    esi
push    edi
lea    edi, [ebp+var_F8]
mov     ecx, 3Eh
mov     eax, 0CCCCCCCCh
rep stosd
mov     eax, ___security_cookie
xor     eax, ebp
mov     [ebp+var_4], eax
mov     [ebp+delimiter], offset a:" "
mov     eax, dword ptr ds:
mov     dword ptr [ebp+szSe
mov     ecx, dword ptr ds:
mov     dword ptr [ebp+szSe
mov     edx, dword ptr ds:
mov     dword ptr [ebp+szSecretString+8], edx
lea    eax, [ebp+szSecretString]
push    eax ; src
```

Keep walking.
Nothing to see here...

CHEATING...

Client:

- Calculates login length
- Appends the length (in ASCII) to the login string.
- Appends a “custom” string
- *Encrypts* everything

Server: Length value (**from client**)
used to *malloc()* & *strcpy()*



STAND BACK



**I'M GOING TO TRY
SCIENCE**



WHERE TO GO FROM HERE

- Better static / dynamic analysis
 - Automatization
 - Heuristic based
- Save / restore snapshot
 - Full emulation (Thx @pleed_ !)
 - Qemu-dbi?

EVERYTHING IS ONLINE

You can **lulz** at my code at:

<https://github.com/carlosgprado/Boyka>

@m0n0sapiens

carlos.g.prado@gmail.com

**IT'S
BEEN
LOVELY
BUT
I HAVE TO
SCREAM
NOW**

