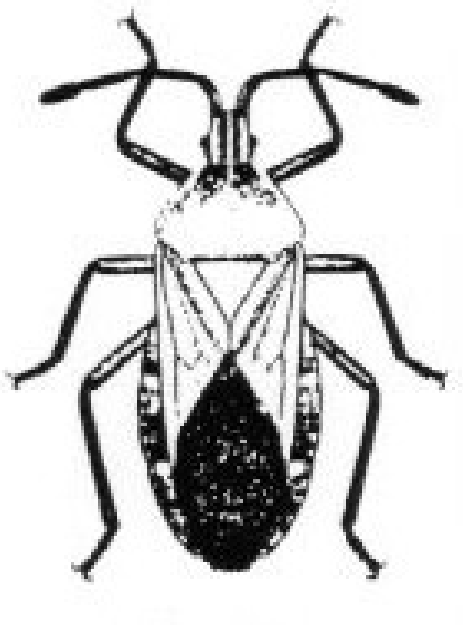
APRESENTA

http://nerv.5p.org.uk/

Bug Hunting



Objetivo: desmistificar alguns dos processos de descoberta de falhas de software, discutir novas idéias para problemas antigos.

Tópicos

- Source code audit
- Fault injection
- Auditing Binary Files

Ferramentas....

gcc, gdb, nasm, windbg, binnavi, ollydbg, rats, softice, visual c++, flawfinder, lint, ida, perl, idc scripts, blast, bindiff, pedump, ltrace, strace, grep, fstat, elfsh, ethereal, ida, eclipse, splint, peach, python, binaudit, gnu-binutils, spike, netcat, tcpdump, vulntrace, bastard, bcc, libdisas, peview... e por ai vai!!

Source Audit

Use the source, luke!

Rats/Flawfinder vs Splint/Blast

Duas abordagens diferentes.

1. Ferramentas como Rats/Flawfinder são bastante limitadas, porém fáceis de usar.

2. Splint/Blast são mais inteligentes, contudo é nescessário aplicar *tags* no código alvo.

Análise manual

Algumas dicas...

1. Use um bom editor de preferência com renomeiação de váriaveis, *code refactor*, correção de identação.. (eclipse?)
2. Loops, quanto maior e mais complexo melhor!
3. Acompanhe os últimos patchs de aplicações opensource...
4. Não acredite no que você vê, compiladores diferentes geram códigos diferentes...

Fault Injection

Fuzzers.

GIVE ME THE INPUT !!

1. Comand line arguments
2. Environment Variables
3. Network protocols
4. Object Files Parsing
5. Shared Memory
6. Distributed Objects
7. Whatever you can write over!!

Fuzzing, Fault injection... (whatever)

Vantagem

- Permite encontrar falhas que podem ser “alcançaveis” em nosso código alvo.

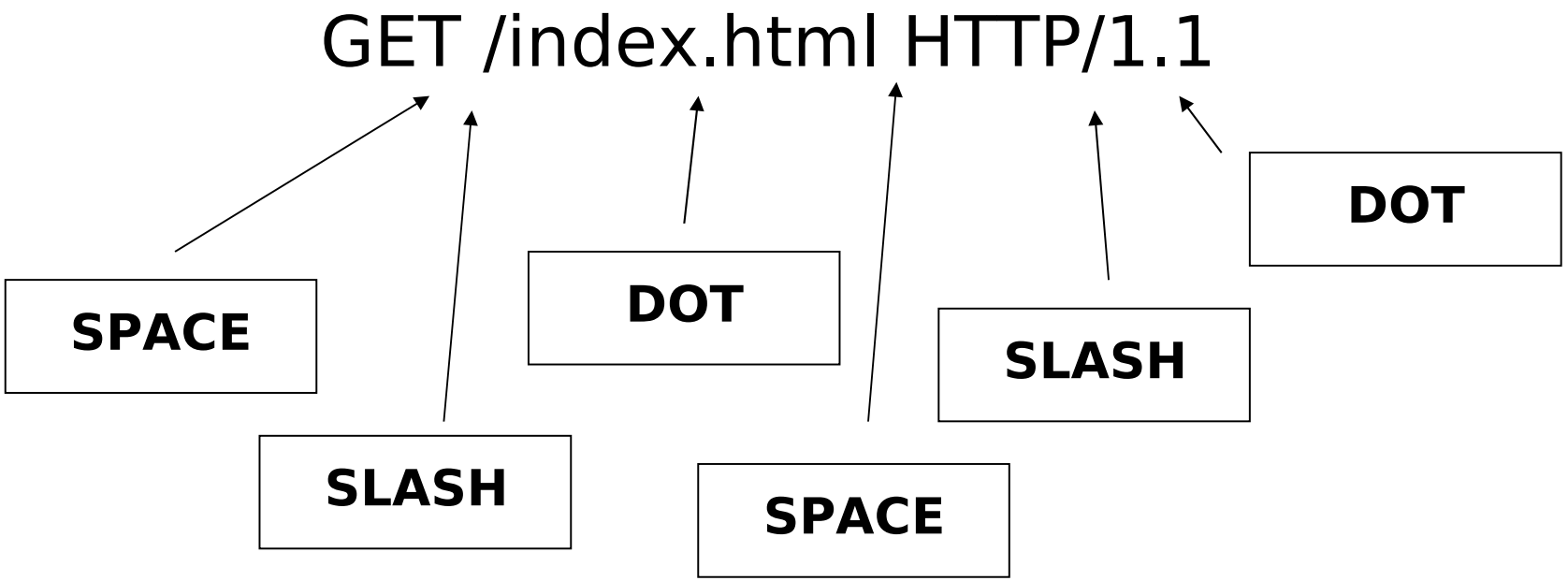
Desvantagens

- Muitas falhas só podem ser alcançadas com uma combinação específica de entradas.

```
If (strcmp(buf1, buf2) { strcpty(buf3, buf4); }
```

Delimiters Logic

Entendendo o protocolo HTTP...



Block Fuzzers

Peach

- Código bem escrito e documentado em python, independente de plataforma...

Spike

- Desenvolvimento rápido em protocolos não populares via ethereal dissectors...

Fuzzers: problemas...

- São poucos amigáveis... scripting language de todos eles não é muito inteligente, muuuito trabalho manual ainda !!

- Na grande maioria são burros... não fazem **uso do feedback** da aplicação !!

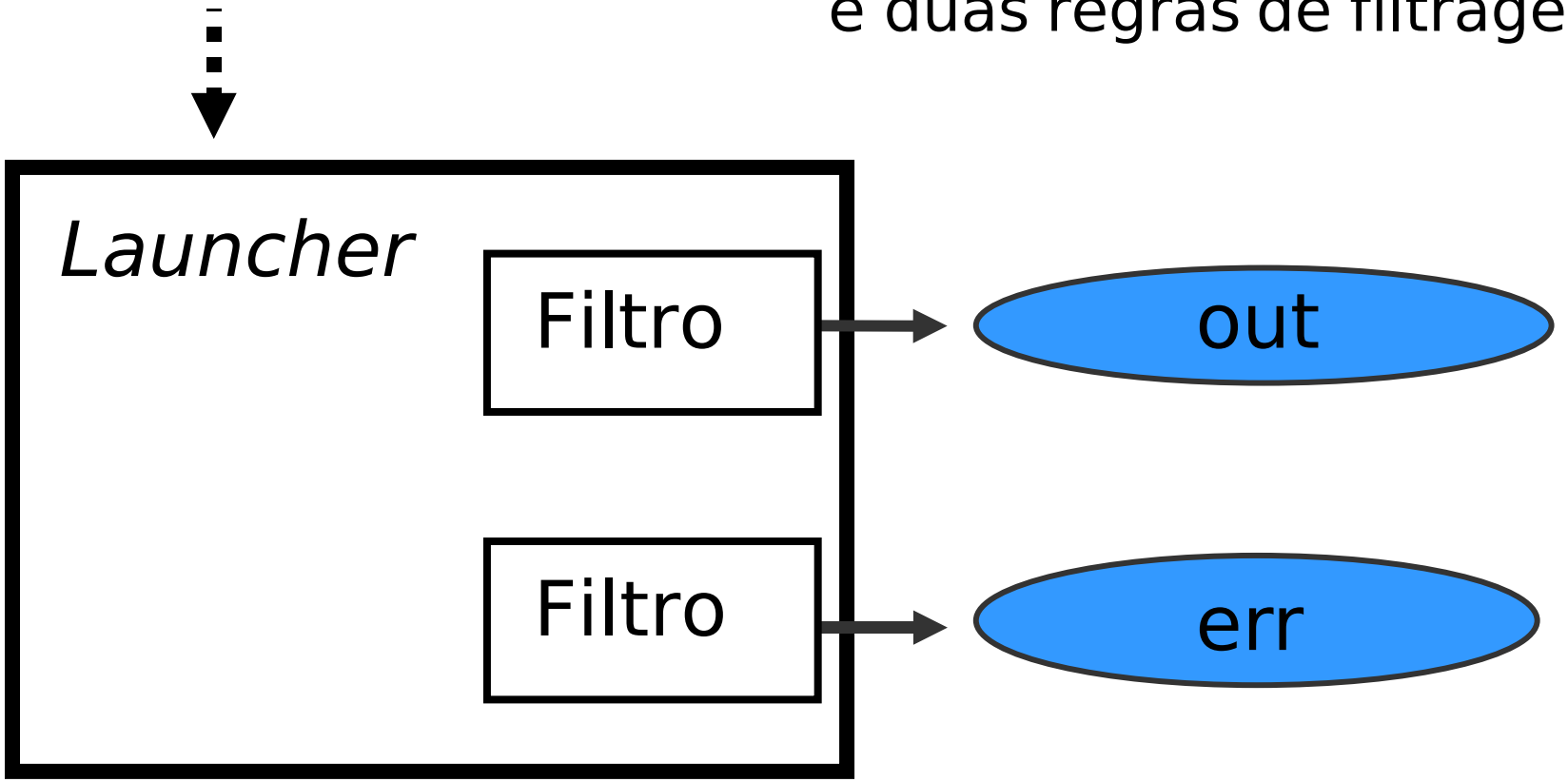
Fuzzers: soluções??

- Reescrever um fuzzer por completo que faz uso do feedback da aplicação?!?!

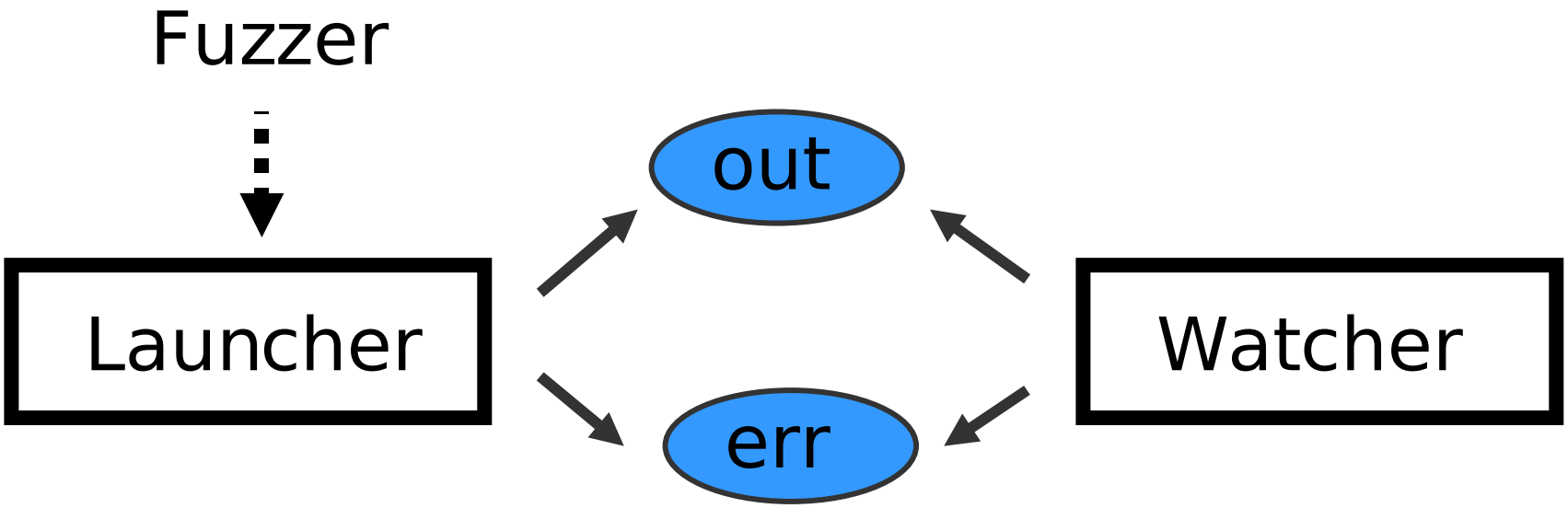
NÃO!! Trabalho manual em execesso!!

Que tal geral um *parser* nos *Ouputs* para “configurar” nossa próxima investida?? Essa é a idéia da **FuzzEngine**...

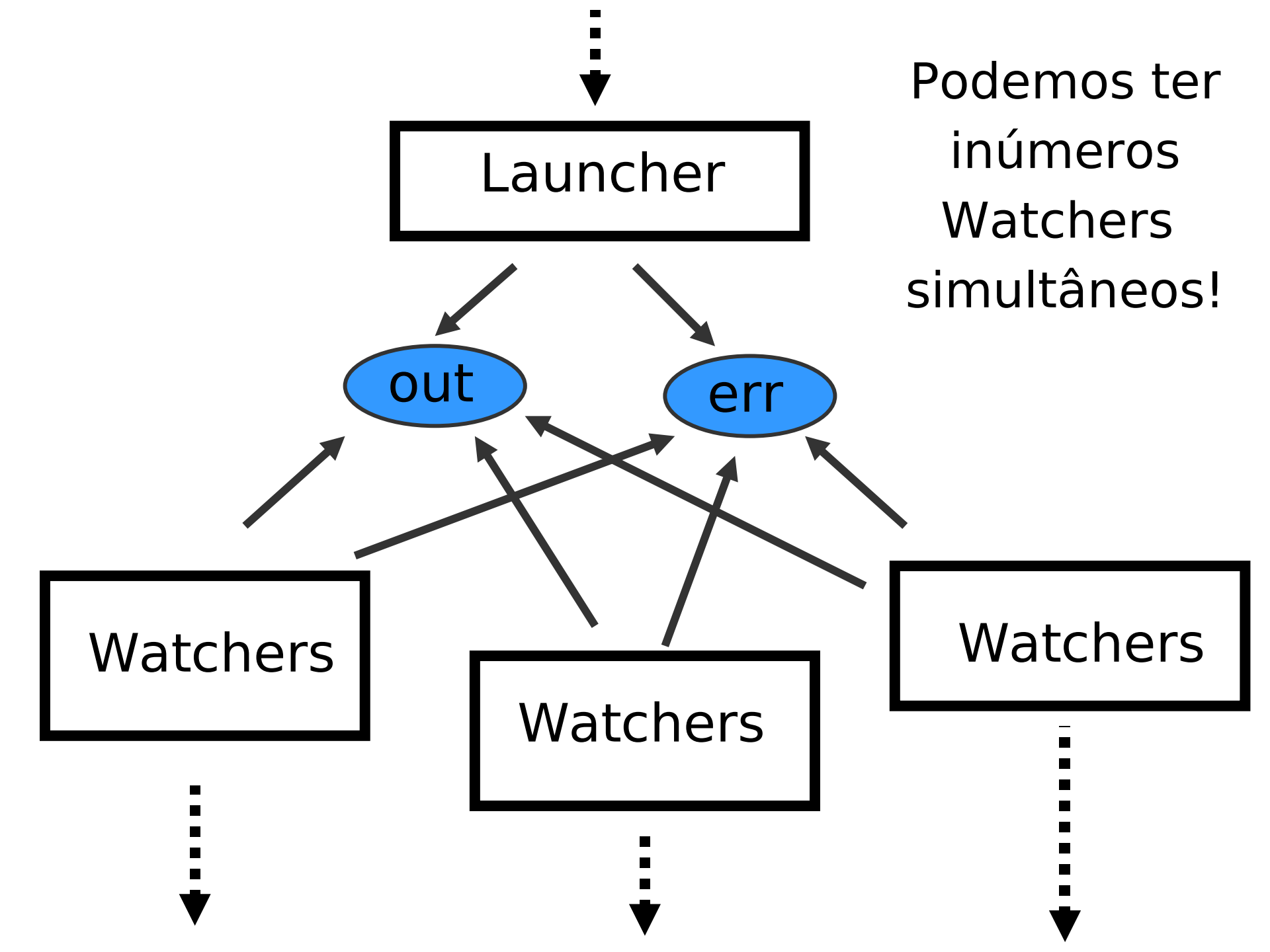
O *launcher* recebe um *fuzzer* para disparar
e duas regras de filtragem...

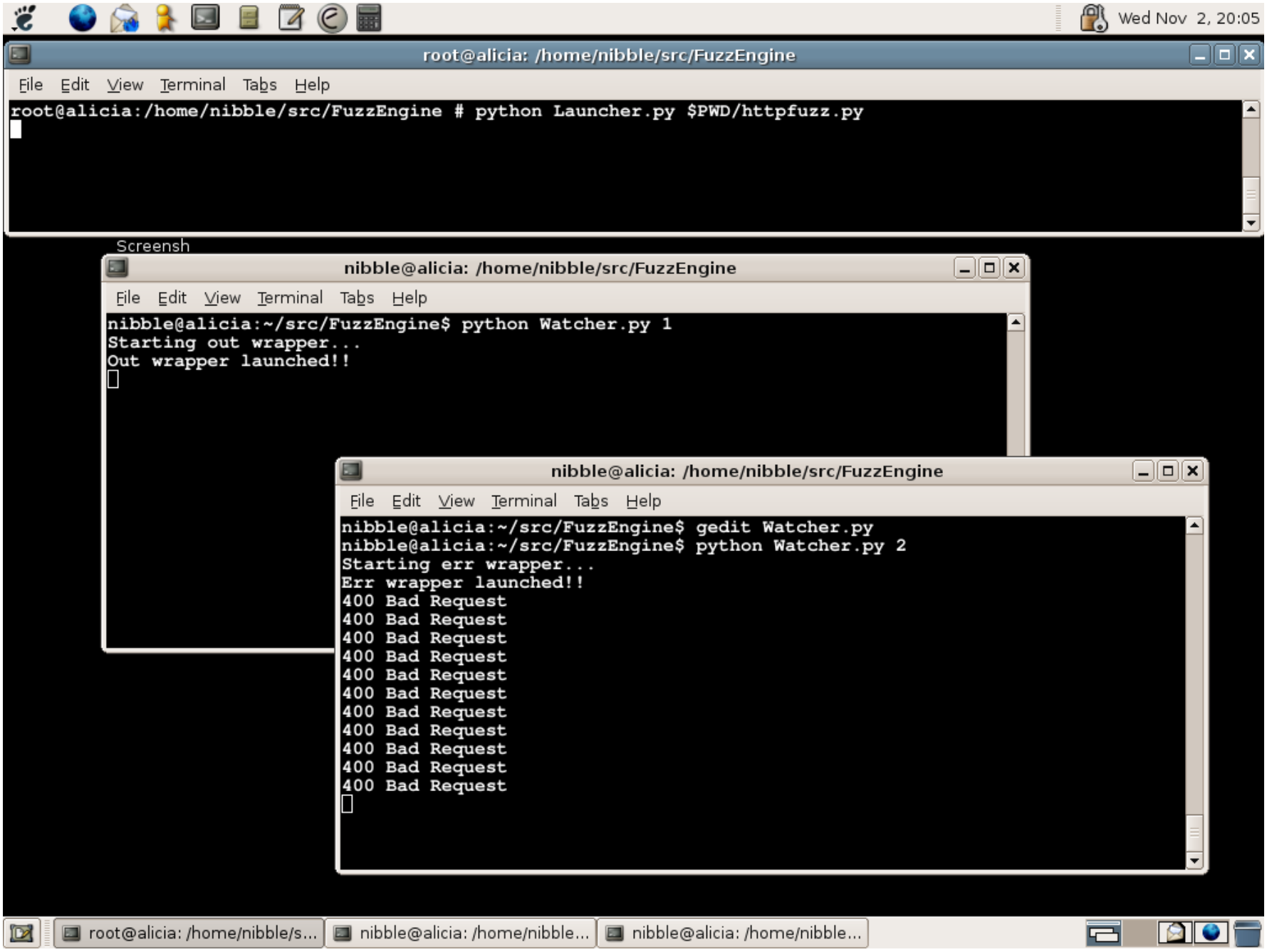


...o feedback da aplicação é filtrado segundo sua regra e direcionado a memória mapeada!!



O *Watcher* análisa, em tempo de execução, as saídas filtradas pelo *Launcher* !!


Launcher
out
err
Watchers
Watchers
Watchers
Podemos ter
inúmeros
Watchers
simultâneos!


Wed Nov 2, 20:05
root@alicia: /home/nibble/src/FuzzEngine
File Edit View Terminal Tabs Help
root@alicia:/home/nibble/src/FuzzEngine # python Launcher.py $PWD/httpfuzz.py
Screensh
nibble@alicia: /home/nibble/src/FuzzEngine
File Edit View Terminal Tabs Help
nibble@alicia:~/src/FuzzEngine$ python Watcher.py 1
Starting out wrapper...
Out wrapper launched!!
nibble@alicia: /home/nibble/src/FuzzEngine
File Edit View Terminal Tabs Help
nibble@alicia:~/src/FuzzEngine$ gedit Watcher.py
nibble@alicia:~/src/FuzzEngine$ python Watcher.py 2
Starting err wrapper...
Err wrapper launched!!
400 Bad Request
400 Bad Request
400 Bad Request
400 Bad Request
400 Bad Request
400 Bad Request
400 Bad Request
400 Bad Request
400 Bad Request
400 Bad Request
root@alicia: /home/nibble/s...
nibble@alicia: /home/nibble...
nibble@alicia: /home/nibble...

FuzzEngine: hoje...

Vantagems

- Facilmente integrável com Peach.
- Distribui filtragem de informações relevantes.
- Permite configurar interpretações diferentes com a mesma base dados visto que pode-se ter diversos *Watchers* simultâneos.
- Simplicidade da implementação.

Desvantagens

- Application-specific, application-specific...

FuzzEngine: futuro...

Para onde caminhar?? Idéias??

1. Inteligência... (onde entra o fator humano? até onde se pode automatizar?)
2. Tornar *scriptable*? Vale a pena?
3. *Launcher, Watchers...* existe mesmo hierarquia?? *Launcher* != *Watcher* ??

Outro case: BugHunt

BugHunt surgiu para tentar solucionar outro problema **supra-impotância**...

Uma **GUI** para *meus fault injectors* !!

S:D

BugHunt: motivos

Alguma coisa contra GUIs?

1. Fuzzers bem feitos precisam de muito pouca ou nenhuma interação durante execusão!!
2. Ferramentas de pen-testing são hoje desenvolvidas em sistemas baseados em GUIs... por que vuln-dev não?
3. As vezes eu só queria clickar!!

BugHunt: implementação

Ferramentas...

- Python
- Eclipse/PyDev
- pyGTK/GTK
- Glade

BugHunt: características

BugHunt!! BugHunt!! BugHunt‼

1. **Cross-platform !!**
2. **Pluginable !!**
3. **User-friendly !!**

Vamos às características de implementação...

BugHunt: cross-platform (1)

Vantagems

- O novo modulo “subprocess” pra série 2.4 de python faz uma interface transparente para execução de aplicações tanto em windows e quanto linux com feedbacking mais interessante do que os antigos metodos popen, popen2, popen3, popen4...

Desvantagens

- O wrapper de sinais (para ter um feedback mínimo) parece não funciona como desejado no windows...

BugHunt: pluginable (2)

Glade is easy!!

- Basta editar o XML dialogs.glade para adicionar um uma nova interface grafíca.
- O bughunt.py precisa ser alterado apenas um lugar (para *adicionar de 5 até 10 linhas*).
- Adicione o fault injector no $PATH e pronto!

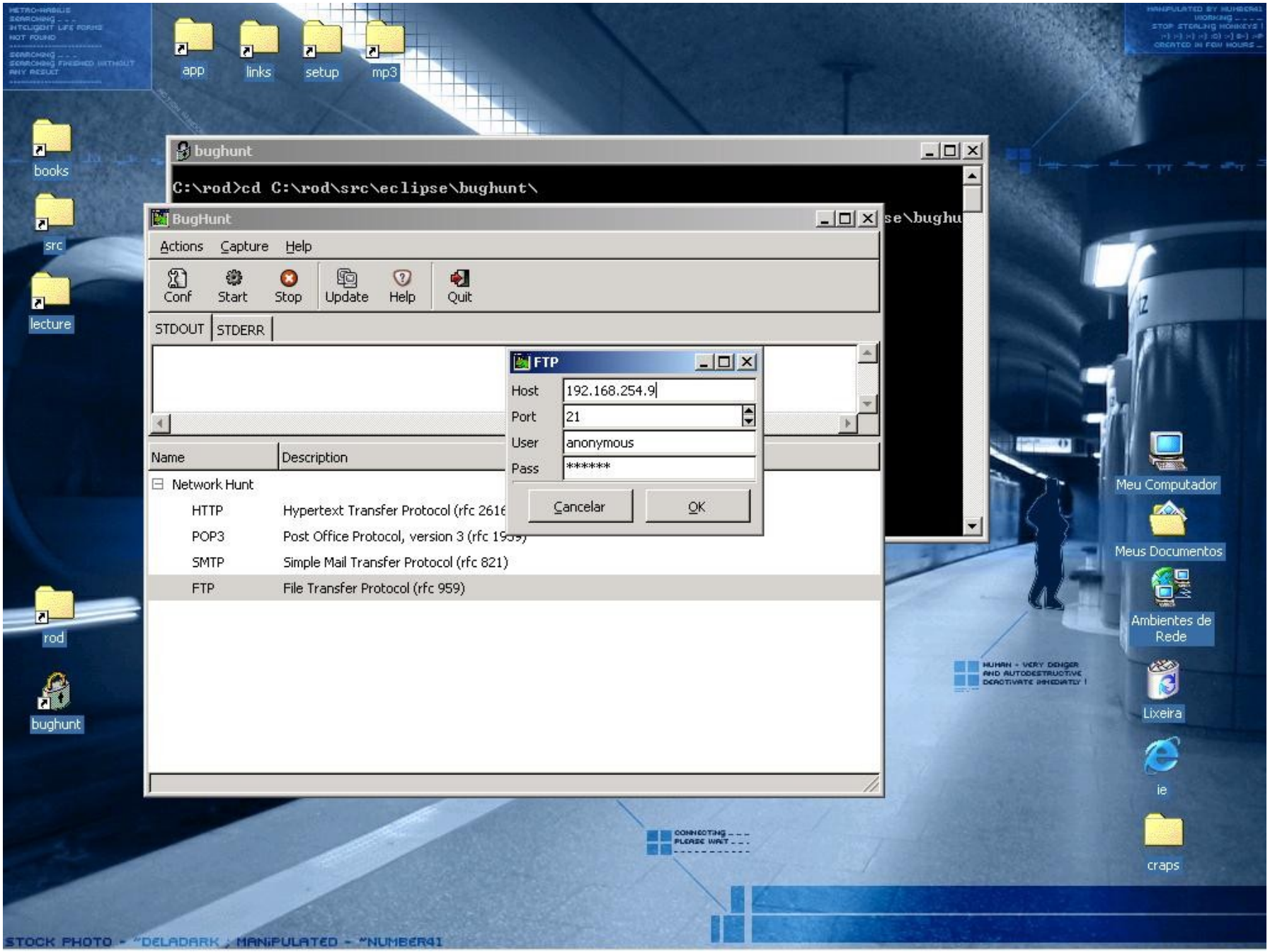
PS: Por isso que live-update é facilmente implementado!!

BugHunt: user-friendly (3)

GUI interface...

- Apenas informações relevantes ao dado fault injector (e.g. login/senha) são requeridas...
- Configuração de automatizada via *janelinhas* de ferramentas concebidas para linha de commando (e.g. script para Peach)...

PS: BugHunt não é um front-end para o Peach... pode ser usado assim... mas NÃO foi feito para tal... não insistam!! :P


app
links
setup
mp3
books
src
lecture
rod
bughunt
bughunt
C:\rod>cd C:\rod\src\eclipse\bughunt\
BugHunt
Actions
Capture
Help
Conf
Start
Stop
Update
Help
Quit
STDOUT
STDERR
Name
Description
Network Hunt
HTTP
Hypertext Transfer Protocol (rfc 2616
POP3
Post Office Protocol, version 3 (rfc 1939)
SMTP
Simple Mail Transfer Protocol (rfc 821)
FTP
File Transfer Protocol (rfc 959)
FTP
Host
192.168.254.9
Port
21
User
anonymous
Pass
******
Cancelar
OK
Meu Computador
Meus Documentos
Ambientes de Rede
Lixeira
ie
craps
HUMAN - VERY DENGER AND AUTODESTRUCTIVE DEACTIVATE IMMEDIATLY !
CONNECTING PLEASE WAIT
STOCK PHOTO - "DELADARK" MANIPULATED - "NUMBER41"

BugHunt: conclusão

Não é tão simples...

não se iludam!!

Caixas preta não são muito interessantes em segurança da informação, cada implementação tem seus detalhes, mesmo sendo um protocolo bem definido, e é justamente nesse ponto que se encontram muitas falhas...

Binary Audit

Reversing.

Binary Static Analysis

- Análise *botton-up* apartir de chamadas à funções de transferência de dados. (e.g. memcpy(), strcpy(), ...)

- Análise top-down, mapeando o fluxo das entrada de dados naturalmente.

  (e.g. input parsing)

Binary Analysis: linking issues

Static Linked

- Uma assinatura da função alvo será necessaria para identificá-las...

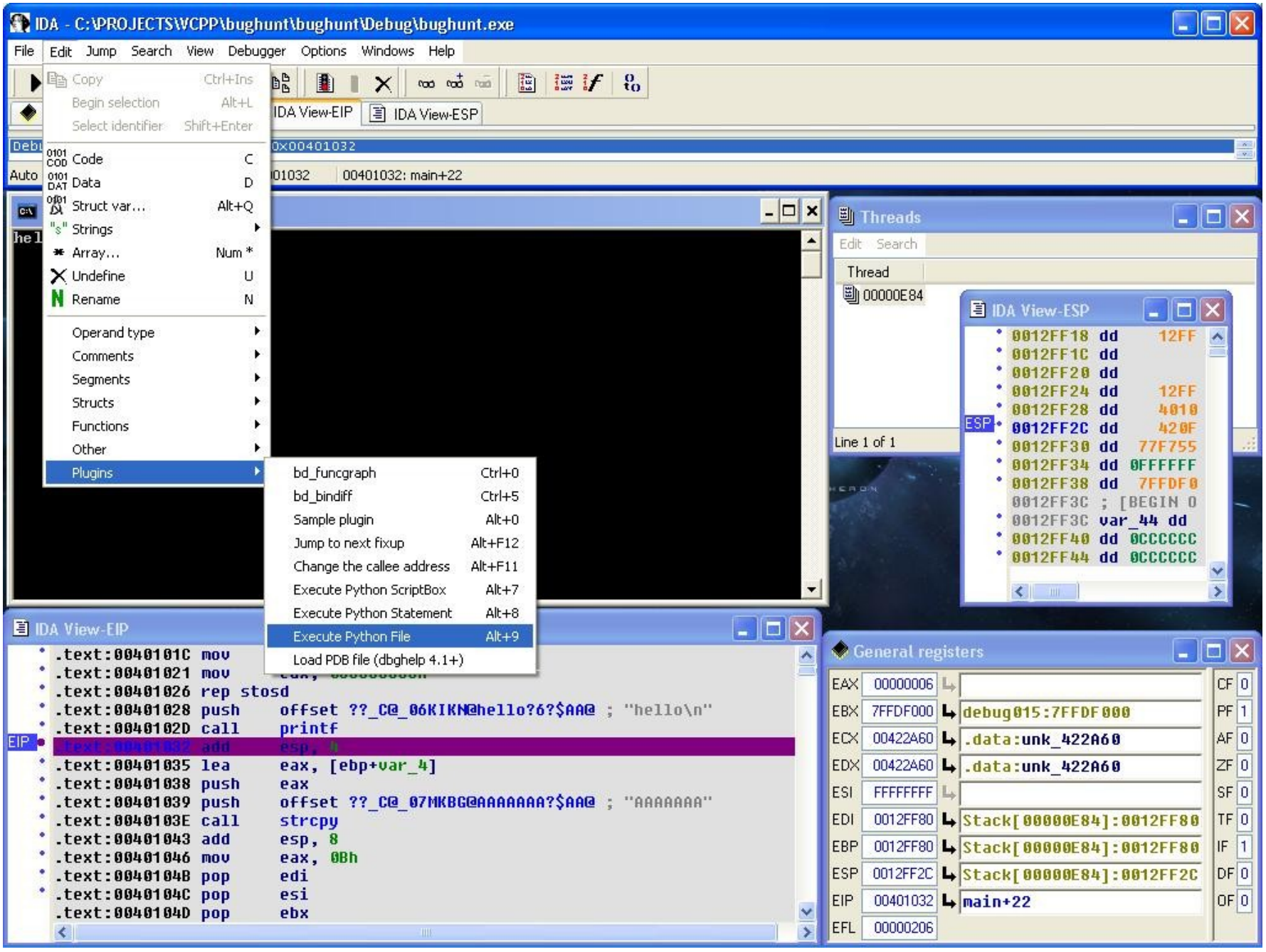
Dynamic Linked

- GOT(elf) / IAT(pe) possuem os endereços da funçoes mapeadas...

Binary Analysis: tracing for vulns

The future is now...

- Machine-code Analysis...
- Code resconstruction...
- Hooking functions for tracing...
- Fingerprinting Systems...


IDA - C:\PROJECTS\WCPP\bughunt\bughunt\Debug\bughunt.exe
File Edit Jump Search View Debugger Options Windows Help
Copy Ctrl+Ins
Begin selection Alt+L
Select identifier Shift+Enter
Code C
Data D
Struct var... Alt+Q
Strings
Array... Num *
Undefine U
Rename N
Operand type
Comments
Segments
Structs
Functions
Other
Plugins
bd_funcgraph Ctrl+0
bd_bindiff Ctrl+5
Sample plugin Alt+0
Jump to next fixup Alt+F12
Change the callee address Alt+F11
Execute Python ScriptBox Alt+7
Execute Python Statement Alt+8
Execute Python File Alt+9
Load PDB file (dbghelp 4.1+)
IDA View-EIP
IDA View-ESP
00401032: main+22
Threads
Edit Search
Thread
00000E84
Line 1 of 1
IDA View-ESP
0012FF18 dd 12FF
0012FF1C dd
0012FF20 dd
0012FF24 dd 12FF
0012FF28 dd 4010
ESP 0012FF2C dd 420F
0012FF30 dd 77F755
0012FF34 dd 0FFFFFF
0012FF38 dd 7FFDF0
0012FF3C ; [BEGIN O
0012FF3C var_44 dd
0012FF40 dd 0CCCCCC
0012FF44 dd 0CCCCCC
IDA View-EIP
.text:0040101C mov
.text:00401021 mov
.text:00401026 rep stosd
.text:00401028 push offset ??_C@_06KIKN@hello?6?$AA@ ; "hello\n"
.text:0040102D call printf
EIP .text:00401032 add esp, 8
.text:00401035 lea eax, [ebp+var_4]
.text:00401038 push eax
.text:00401039 push offset ??_C@_07MKBG@AAAAAAA?$AA@ ; "AAAAAAA"
.text:0040103E call strcpy
.text:00401043 add esp, 8
.text:00401046 mov eax, 0Bh
.text:0040104B pop edi
.text:0040104C pop esi
.text:0040104D pop ebx
General registers
EAX 00000006
EBX 7FFDF000 debug015:7FFDF000
ECX 00422A60 .data:unk_422A60
EDX 00422A60 .data:unk_422A60
ESI FFFFFFFF
EDI 0012FF80 Stack[00000E84]:0012FF80
EBP 0012FF80 Stack[00000E84]:0012FF80
ESP 0012FF2C Stack[00000E84]:0012FF2C
EIP 00401032 main+22
EFL 00000206
CF 0
PF 1
AF 0
ZF 0
SF 0
TF 0
IF 1
DF 0
OF 0

IDAPython... IDC with Python!!

My double-strncpy()
& double-free() Like,
Vuln Wrapper!!

Identifica e retorna
endereços de funções
que possuem chamadas
seqüenciais no código!

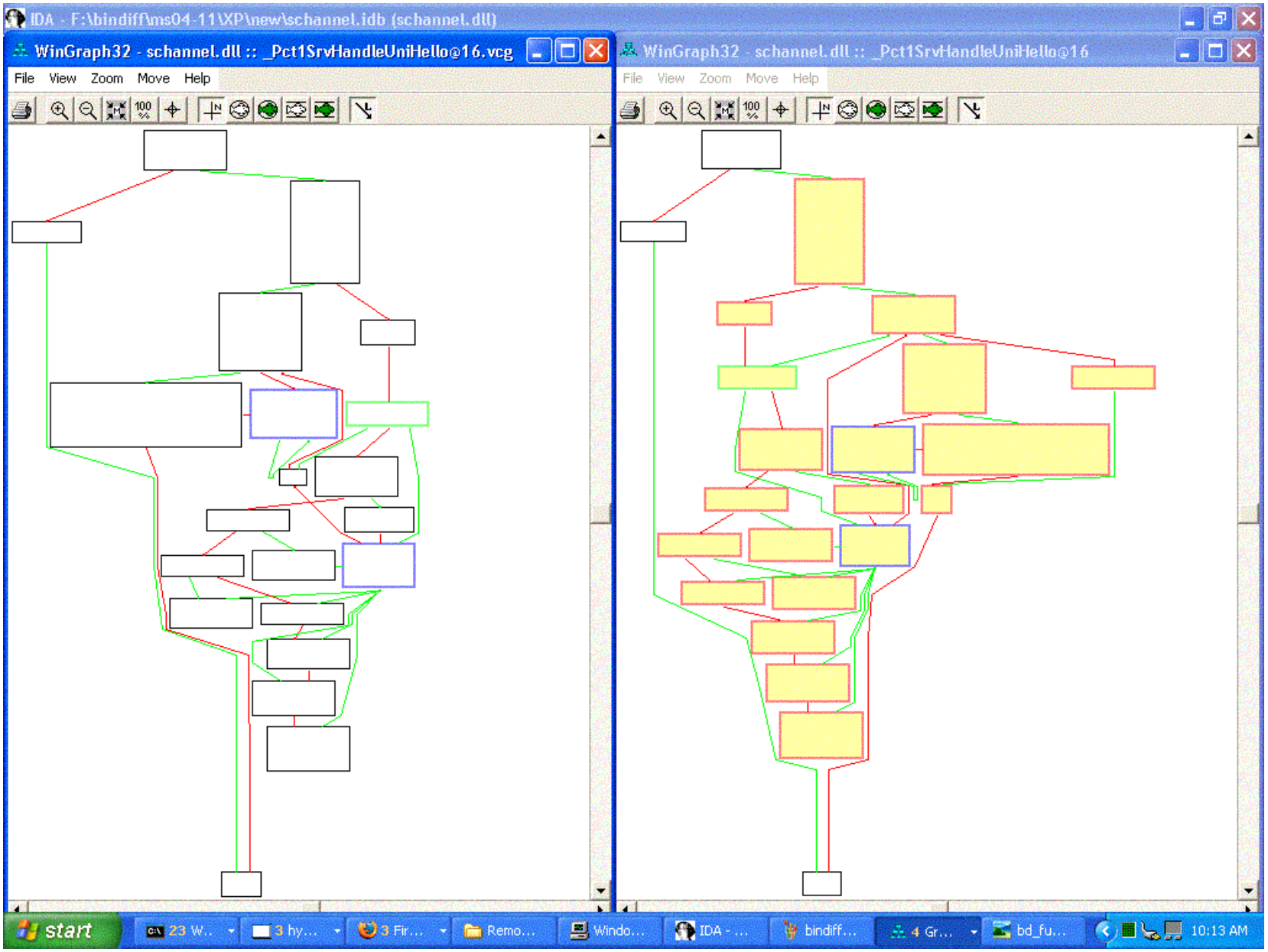
```
# segment's starting address
ea = ScreenEA()

i = 0
f1 = f2 = ""

print "start!"
for function_ea in Functions(SegStart(ea),
        SegEnd(ea)):
    f1 = GetFunctionName(function_ea)
    ea1 = function_ea
    if i == 1: # print f1, f2
        if (f1 == f2):
            print hex(ea1), f1
            print hex(ea2), f2
        f2 = f1
        ea2 = ea1
        i = 0
    i = i +1
```

IDA Run-time Analysis

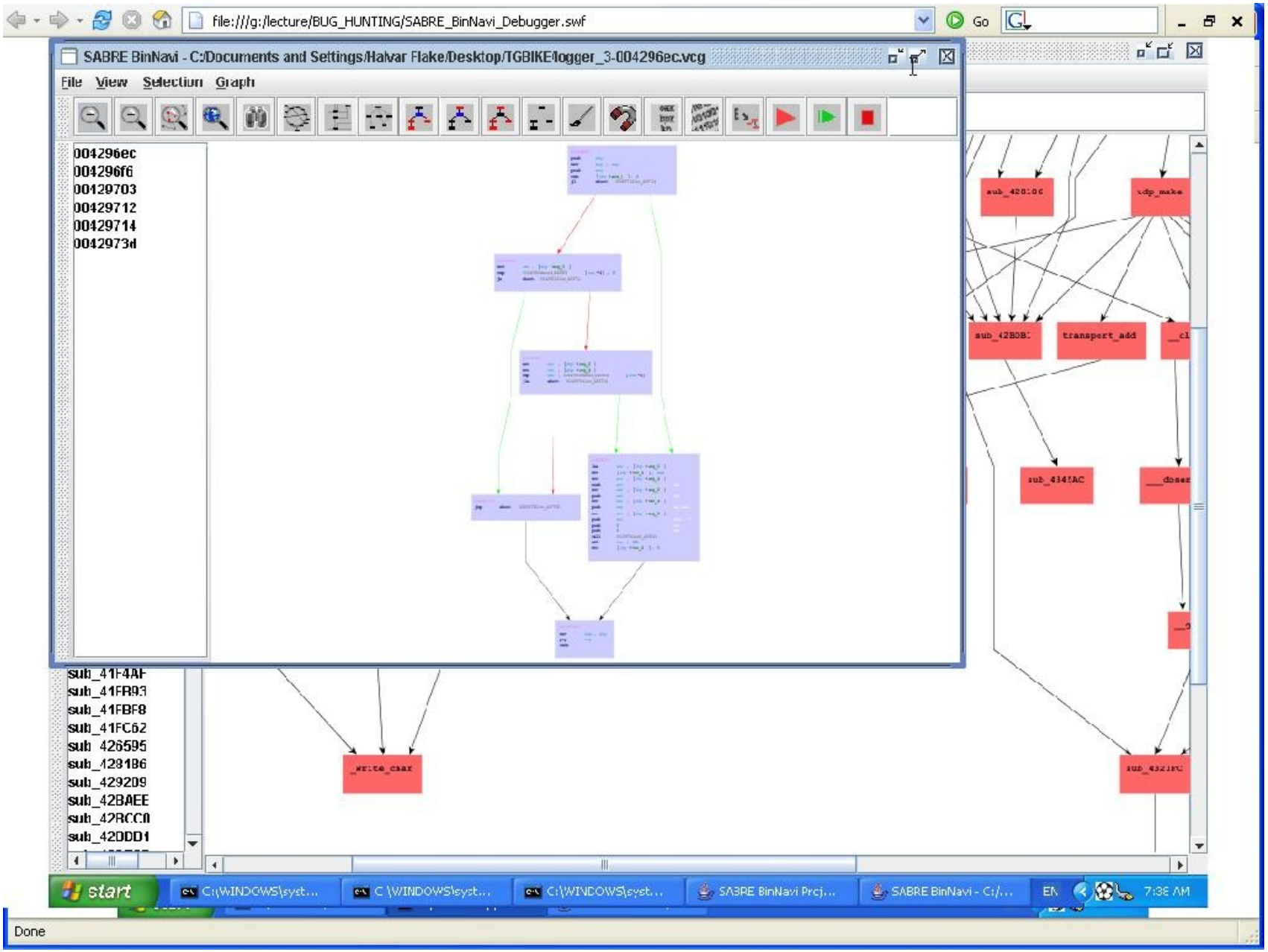
"Toc, Toc... Wake up neo..."

1. Os debuggers permitem alto grau de abstração e capacidade de visualização do fluxo de execução em grafos!!
2. Linguagems de scripting, como IDC, permitem automatizar grande parte do serviço... como identificar os principais pontos críticos!!

BinDiff – binary difference debugging and visualisation.



BinNavi – debugging system based ongraph visualisation.



Por que tanto código ruim?

- Programadores trabalham sobre pressão e prazos curtos...
- Segurança é *requisito não funcional* para os gerentes de projeto...
- Beta-testers são mal-pagos e, às vezes, até mesmo vistos como subempregados!

Onde publicar? Deve-se Publicar?

- Open Source Vulnerabilty DB
  http://osvdb.org/
- Common Vulnerabity and Exposures
  http://www.cve.mitre.org/
- Dentro tantos outros...
  http://xforce.iss.net/xforce/search.php
  http://www.securityfocus.com/bid

Links

http://home.arcor.de/idapalace/index.htm
http://www.idefense.com
http://www.datarescue.com/idadoc/index.htm
http://www.sabre-security.com/products/
http://www.splint.org/

"Security is a process, not a product."

- Bruce Schneier