Your Engine, Your Rules

Attack Surface Intelligence of Source Code
Simon Roses Fenerling

- Founder & CEO, VULNEX [www.vulnex.com](http://www.vulnex.com)
- @simonroses
- Former Microsoft, PwC, @Stake
- Black Hat, RSA, OWASP, SOURCE, AppSec, DeepSec, TECHNET

VULNEX

- CyberSecurity Startup
- @vulnexsl
- Services & Training
- Products: BinSecSweeper
TALK OBJECTIVES

• GCC & Python, hand to hand

• Transformations: source code to useful data

Give a man a fish and you feed him for a day. Teach a man how to fish and you feed him for a lifetime.

-LAO TZU
AGENDA

1. The need of Attack Surface Intelligence of Source Code
2. GCC Overview
3. GCC-Python-Plugin
4. Source Code Intelligence
5. Tintorera Overview
6. Tintorera Analysis Demos
7. Conclusions
8. Q&A
1. The need of Attack Surface Intelligence of Source Code
## 1. CODE IS GETTING COMPLEX!

<table>
<thead>
<tr>
<th>Software</th>
<th>SLOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firefox</td>
<td>14 Million</td>
</tr>
<tr>
<td>Windows Server 2003</td>
<td>50 Million</td>
</tr>
<tr>
<td>Debian 7.0</td>
<td>419 Million</td>
</tr>
<tr>
<td>Mac OS X 10.4</td>
<td>86 Million</td>
</tr>
<tr>
<td>Linux Kernel 2.6.25</td>
<td>13.5 Million</td>
</tr>
<tr>
<td>Linux Kernel 3.6</td>
<td>15.9 Million</td>
</tr>
</tbody>
</table>
1. DOCUMENTATION

MISSING

IT Documentation

Our server crashed and we’re not super jazzed about it. We don’t know where our backups are, but are pretty sure they exist. Could really use some IT documentation right about now.

CAN YOU HELP?
Please call 778-555-6666
1. TYPICAL CODE REVIEW

TIME
1. WHERE TO START?

- File operations
- Networking
- Processes
- Crypto
- Authentication
- ??
1. TOOLS?

- grep
- rough-auditing-tool-for-security
- eclipse
1. UNDERSTANDING: BIG FAIL
2. GCC Overview
2. GCC

- Compiler system that supports various programming languages
- Popular UNIX variants
- Supports all major languages: C, C++, Java, Objective-C, etc.
- PLUGINS!!
- FREE
2. GCC INTERNALS
2. GCC TERMINOLOGY

- GENERIC is common representation shared by all front ends
  - Each parser must emit GENERIC

- GIMPLE is a simplified version of GENERIC
  - 3 address representation
  - Simplified control flow

- RTL (Register Transfer Language), assembler for an abstract machine
2. GCC PASSES

The lowering passes

*warn_unused_result
*diagnose_omp_blocks
mudflap1
omplower
lower
ehopt
eh
cfg
*warn_function_return
*build_cgraph_edges

gimple_any

gimple_lcf
gimple_ief
gimple_lomp

gimple_ssa

gimple_referenced

The "small IPA" passes

*free_lang_data
visibility
early_local_cleanups
*free_cfg_annotations
*init_datastructures
ompexpp
*referenced_vars
ssa
veclower
*early_warn_uninitialized
*rebuild_cgraph_edges
inline_param
einline
early_optimizations
*remove_cgraph_callee_edges
copyrename
ccp
forwprop
ealias
esra
copyprop
mergephi
cddf
eiga_sra
tail
switchconv
ehcleanup
profile
local-pure-const
fnsplit
release_ssa
*rebuild_cgraph_edges
inline_param
tree_profile_ipa
feedback_fnsplit

3. GCC-Python-Plugin
3. GCC-PYTHON-PLUGIN

- GCC plugin that embeds Python in GCC 😊
- Now your Python script can access GCC passes and perform analysis
- Developed by David Malcolm (Fedora)

# Sample python script, to be run by our gcc plugin
# Show all the passes that get executed
import gcc

def my_pass_execution_callback(*args, **kwargs):
    (optpass, fun) = args
    print(args)

gcc.register_callback(gcc.PLUGIN_PASS_EXECUTION,
                     my_pass_execution_callback)
3. GCC-PYTHON-PLUGIN DEMO
3. GCC-PYTHON-PLUGIN IDEAS

• Write scripts for:
  – malloc/free usage
  – Array boundary checks
  – Code visualizations
  – You name it!
4. Source Code Intelligence
4. CODE UNDERSTATING

- What API are being used?
- Number of functions?
- Inputs / Outputs of functions?
- Function relationship
- What comments said?
- Code complexity
4. CODE METRICS

• Controversial topic but needed

• Metrics:
  – Function complexity (Cyclomatic)
  – Number of:
    • Lines
    • Code
    • Blanks
    • Comments
  – Line Length
  – Number: Bugs per Line
  – You name it....
4. CODE COMPLEXITY

- Counts the number of linearly independent paths through the source code

- Basically we can have an idea of the complexity of functions

- Complexity is security enemy!

- Created by Thomas McCabe
  
## 4. CODE COMPLEXITY THRESHOLD

<table>
<thead>
<tr>
<th>Cyclomatic Complexity</th>
<th>Risk Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-10</td>
<td>a simple program, without much risk</td>
</tr>
<tr>
<td>11-20</td>
<td>more complex, moderate risk</td>
</tr>
<tr>
<td>21-50</td>
<td>complex, high risk program</td>
</tr>
<tr>
<td>greater than 50</td>
<td>untestable program (very high risk)</td>
</tr>
</tbody>
</table>

http://www.sei.cmu.edu/reports/97hb001.pdf
4. SOURCE CODE ANALYSIS FLOWGRAPH NOTATION

- If .. then
- If .. then .. else
- If .. and .. then
- If .. or .. then
- Do .. While
- While .. Do
- Switch
4. SOURCE CODE VISUALS TOO

**Binary Code**
```
xan_1000_validate_struct:
    mov    eax, [esp+xan]
test   eax, eax
jz     short loc_90001015
```

**Source Code**
```
BLOCK 0
entry

BLOCK 2
  data = 0B;
  69    data = NULL;
  72    data = new long[100];

source:
D.4317 = operator new [] (400);
data = D.4317;

BLOCK 3
  data = new long[100];

source:
D.4317 = operator new [] (400);
data = D.4317;

BLOCK 4
  sink:
  if (data != 0B)
  sink:
  if (data != 0B)

true

BLOCK 5
  operator delete [] (data);
  77    delete [] data;

false

BLOCK 6
  return;
  78 }
  return;

BLOCK 1
exit
```

**Visual Diagram**

VULNEX
5. Tintorera Overview
5. TINTORERA – BLUE SHARK

• “Put source code into context”

• Objective: Get a feeling of the code while compiling!!

• Intelligence of source code:
  – Code visualizations
  – Comments analysis
  – API identification
  – Metrics
  – HTML Reports

• C code transformed to JSON files, now you can query and perform analysis on data
5. CODE TRANSFORMATION

SOURCE CODE ➔ JSON FILES ➔ HTML REPORT
5. TRANSFORMED JSON FILES

• 3 files:

1. tintorera_bb_file.json: code basic blocks

2. tintorera_meta_info.json: general information, file size and code & comments not inside functions

3. tintorera_temp_file.json: functions information
5. TINTORERA SOURCE CODE METRICS

• Current metrics:

1. Number of:
   1. Lines
   2. Code
   3. Blanks
   4. Comments
   5. Colons
2. Average line length
3. Minimum line
4. Maximum line
5. Total Basic Blocks
6. Total Cyclomatic Complexity
7. Average Cyclomatic Complexity
6. Tintorera Analysis Demos
6. DEMO I: LOOP TESTER
6. DEMO I: LOOP TESTER
6. DEMO I: LOOP TESTER

IF ELSE

WHILE

SWITCH
6. DEMO II: SENDMAIL CRACKADDR (CVE2002-1337)

Pure Complexity....
Function = sendmail_crackaddr_cve2002_1337.c -> crackaddr

Function Details

<table>
<thead>
<tr>
<th>Function Name</th>
<th>crackaddr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arguments (1)</td>
<td>[u'char *]</td>
</tr>
<tr>
<td>Return Type</td>
<td>char *</td>
</tr>
<tr>
<td>Function LOC</td>
<td>247</td>
</tr>
<tr>
<td>Function Physical LOC</td>
<td>334</td>
</tr>
<tr>
<td>Function Start Line</td>
<td>56</td>
</tr>
<tr>
<td>Function End Line</td>
<td>390</td>
</tr>
<tr>
<td>Comments</td>
<td>54</td>
</tr>
<tr>
<td>Blank Lines</td>
<td>33</td>
</tr>
</tbody>
</table>

Code Details & Metrics

<table>
<thead>
<tr>
<th>Basic Blocks</th>
<th>155</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code Complexity</td>
<td><strong>89</strong></td>
</tr>
<tr>
<td>Metrics-&gt; Count Colons</td>
<td>125</td>
</tr>
</tbody>
</table>
6. DEMO II: SENDMAIL CRACKADDR (CVE2002-1337)
FUNCTION COMPLEXITY
6. DEMO III: OBFUSCATED C CODE ANALYSIS, ENDOH4.C

```c
int
**F,**
V,M,N,i;
#ifndef/**/S
#define S 70,23
#endif/* 000-2E5*/
#define/* 2E5-2E5,2E5
*/_POSIX_C_SOURSE 199309
#include/* 2E5XXX*/<time.h>
/* 2E5-2E5X*/#include<stdio.h>
#include<stdlib.h>/* -2E5-2E5XX*/
struct timespec R={0,1E6};int j,k,m,
#define U/* -2E5X*/rand()*/2./RAND_MAX-1
#define/* 2E5*/0(p,q,i)(P[p*3+i]-P[q*3+i])
*/ IOCC2013 IOCCC2013*/#define B(p,q,
r)(0(q,p,0)*0(r,p,1)-0(q,p,1)*0(r,p, 0))
#define A(t,n)( t*)malloc( sizeof( (t)*n)
#define E(p,q,r,s)B(p,q,r)*0(s ,p,2)+B(\np,r,s)*0(q,p,2)+B(p,s,q)*0( /XX*/r,p,2)
#define D(e,f)(c-a)?s=a, a=e,e=s,s=f,f=\n d,d=s:0;u=a+.5;m=u+1; T[01]=91;T[2]=060;
#define C (Q[u]-X) *a+(Q[u+1]-Y)*b+(Q[u\n+2]-Z)*c,g=e*c- f*b,h=f*a-d*c,f=c,c=d*b\n-e*a,d=a,a=g ,e=b,b=h,P[k]=W/2-q/s/p*3\n W,P[k+1]= H/2+r/s/p*H/2,T[3]=0x48,*T=033
n,u,v, w,t,W,H;double*P,*Q,I,J,K,L,x,y,z ,X, Y,Z,a,b,c,d,e,f,g,h,p,q,r,s ;void o( double x){for(p=q=i=0,s=r=1;i<999;s=(s+x /s)/2)i%2?q+=r,r=-r:(p+=r),r*=3.14*x/+i;}
int G(int p,int q,int s,int g,int f){if((f==1\nnt* v=A(int,N),*a,*b,h=-1,r=h;for(F[f]=V [f]=v; ++h<f;}if(V[h][p]==q){if(s+1&6(E(p ,q,V[h][q ]),s)<1E-4){for(a=F[g],b=F[h];N >++]v[r]=q+ 1?a[q]-r?q:b[p]-r?p:-1:p)=\na[r],q=b[r];for (r=0;<f<r++)F[r]=a|F[ r]=b?F[r]=v:0;};; return f;}for(h=0;h<N
```
# Files Results

<table>
<thead>
<tr>
<th># All Funcs</th>
<th># File Funcs</th>
<th>File Name</th>
<th>Function Name</th>
<th>Basic Blocks</th>
<th>Cyclomatic Complexity</th>
<th>API Calls</th>
<th>Inline ASM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>endoh4.c</td>
<td>main</td>
<td>93</td>
<td>37</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>endoh4.c</td>
<td>G</td>
<td>38</td>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>endoh4.c</td>
<td>o</td>
<td>9</td>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
6. DEMO III: OBFUSCATED C CODE ANALYSIS, ENDOH4.C

The International Obfuscated C Code Contest - http://www.ioccc.org/
6. DEMO III: OBFSUCATED C CODE ANALYSIS, ENDOH4.C

The International Obfuscated C Code Contest - http://www.ioccc.org/
6. DEMO IV: NGINX EXPLORATION

NGINX
7. Conclusions
7. CONCLUSIONS

• Practical code understanding for:
  – Saving time
  – Security reviews
  – Fuzzing: what and where to fuzz

• With GCC Python Plugin you can write your own static analysis toolkit
8. Q&A

- Thanks!
- @simonroses / @vulnexsl
- www.vulnex.com