CrowdStrike

Be Social. Use CrowdRE.

An IDA Plugin for Collaborative Reversing

Tillmann Werner, Jason Geffner

RECON, Montreal, Canada Friday, June 15, 2012

CrowdStrike

Stealth mode startup

Handpicked 'A' team of technical talent

26 Million Series A funding

■ "You don't have a malware problem, you have an adversary problem"[™]

We are hiring!

Special Thanks

Georg Wicherski Sr. Research Scientist

Aaron Putnam Sr. Research Engineer

TJ Little and Harley Sr. UI Engineers Jeff Stambolsky Resident Nerd





Why CrowdRea?

Developers work in teams to build the software we are reversing

- Stuxnet, Flame, Duqu
- RATs like PoisonIvy
- Bots like Zeus
- calc.exe

Code reuse is prevalent in malware variants

Working together, we can reverse more quickly and efficiently

Take a page from developer world and model RE after source control methodologies

Collaborative Reversing

Approach 1: Just-in-time propagation of results

All changes are synchronized to all users instantly

Well-suited for teaching reverse-engineering, demonstrations, etc.

Approach 2: Working on different parts, sharing results on demand
 Distributed tasks

Multiple people can work on different parts simultaneously

Analysis results can be combined at any time

Related Work – Tools of the Trade

IDA Sync, 2005

Real-time synchronization of names, stack variables, comments
 Hooks into IDA hot keys

CollabREate, 2008

- Successor of IDA Sync: IDA Pro "remote-control"
- Snapshot report: replay all updates up until a certain point

BinCrowd, 2010

- Commit-based model
- Supports matching similar functions

The CrowdREa Platform

Community platform to support professional, distributed RE
 Design similar to version control systems
 Commits: annotations per function

Free Cloud service for the reverse engineering community
 People can share their results
 Reverse engineering projects can benefit from community input

IDA Pro plugin

Utilizes the power of the Hex-Rays Decompiler plugin

Integrates smoothly into IDA's Qt GUI

Rewoltke → CrowdRE





= rewoltke

BinNavi Integration



Google is adding integration for CrowdRE to BinNavi

Analysts will be able to use BinNavi to share their analysis results with the CrowdRE community

Our best wishes go to Thomas Dullien for a speedy recovery

Annotations

Function prototype

Name Calling convention Return type Parameter types and names int i; // ebx@2 lpstring new finish; // eax@3 std basic string rep *string header; // edx@5 void *end; // esi@11 lpstring it: // ebx@11 signed __int32 refcount; // [sp+1Ch] [bp-30h]@15 std vector parameters vector; // [sp+2Ch] [bp-20h]@1 std basic string tmp string; // [sp+38h] [bp-14h]@3 // these three NULL assigns are inlined vector::vector parameters vector.start = 0: parameters vector.finish = 0; parameters vector.end of storage = 0; std::vector<std::string std::allocator<std::string>>::reserve(¶meters vector, argc); if (argc > 0){ i = 0: do std::string::string(&tmp_string, argv[i]); if (parameters vector.finish == parameters vector.end of storage)

```
&parameters vector,
parameters vector.finish,
```

```
&tmp string);
```

Stack variables Register variables (Hex-Rays) Structs, enums Comments – IDA and Hex-Rays

int cdecl main(int argc, const char **argv)

```
std::vector<std::string std::allocator<std::string>>:: M insert aux(
```

Type Information

Types
 Structs
 Enums
 User-defined types

Function annotations depend on types
 Dependencies are recursively included
 Checkouts contain dependencies, too
 Name duplicates require conflict resolution
 User is prompted for solution (update, retain, keep)

Future plan: resolving cyclic dependencies

🐙 Type Conflict Resolution

std basic stripg

The annotation(s) that you are importing reference a type that conflicts with a type of the same name in your IDB.

X

- Update type: Replace the old type with the new type
- C Retain type: Ignore the new type and continue to use the old type
- Keep both: Add the new type and rename the old type to

Jaca_basic_scring_		
Current Type:	New Type:	
<pre>#pragma pack(push, 1) struct stdbasic_string { int one; int two; int three; }; #pragma pack(pop)</pre>	<pre>#pragma pack(push, 1) struct std_basic_string { std_basic_string_rep *p; }; #pragma pack(pop)</pre>	
	Save Cancel	

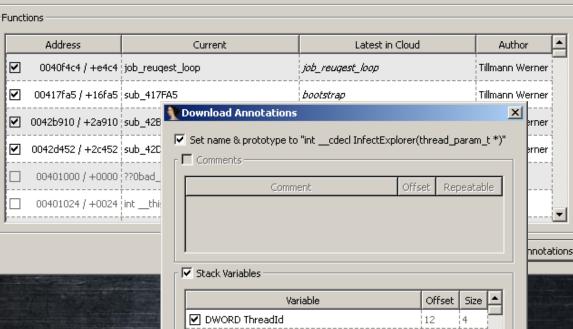
Importing Annotations

Batch import

- The first thing to do when starting to work on a new binary
- Always the most recent commit

Individual imports

- More control over what to import
- User can choose between different versions



untyped var_EC4

Batch Import Annotations

Fuzzy match (based on bytecode hash)
 C Exact match (same binary and relative offset)

-Matching type

✓ HANDLE hProcessToInfect 20 4 ▼ ✓ Register Variables Variable Definition Offset Location ✓ HANDLE hProcess 272 r8 ✓ __untyped Parameter 542 r8

16

Cancel

Import

X

Finding Functions

- Exact matching Binary's hash + function offset
- i1 Fuzzy matching SHA1 hash over sequence of mnemonics
- Position-independent representation Want to cover immediates, too Jump and call operands are zeroed out Same for immediates that generate cross-references

pushebppushebpmovebp, espmovebp, espandesp, 0FFFFF8handesp, 0FFFFF8hsubesp, 14hsubesp, 14hpushebxpushebxpushesipushesipushedileaeax, [esp+20h+Time]leaeax, [esp+20h+Time]leaeax, [esp+20h+Tmovesi, ecxxorebx, ebxpusheax; Timepusheax	
andesp, 0FFFFFF8hsubesp, 14hpushebxpushesipushedileaeax, [esp+20h+Time]movesi, ecxxorebx, ebx	
subesp, 14hsubesp, 14hpushebxpushebxpushesipushesipushedipushedileaeax, [esp+20h+Time]leaeax, [esp+20h+Tmovesi, ecxmovesi, ecxxorebx, ebxxorebx, ebx	
pushebxpushebxpushesipushesipushedipushedileaeax, [esp+20h+Time]leaeax, [esp+20h+Tmovesi, ecxmovesi, ecxxorebx, ebxxorebx, ebx	
pushesipushedileaeax, [esp+20h+Time]movesi, ecxxorebx, ebx	
pushedileaeax, [esp+20h+Time]movesi, ecxxorebx, ebx	
lea eax, [esp+20h+Time] nov esi, ecx xor ebx, ebx	
mov esi, ecx xor ebx, ebx	ime 1
xor ebx, ebx	
push eax ; Time push eax	
	; T
lea edi, [esi+288h] 🛛 🚺 lea edi, [esi+288h]	
mov dword ptr [esp+24h+Time], ebx mov dword ptr [esp+	24h+
mov dword ptr [esp+24h+Time+4], ebx mov dword ptr [esp+	24h+
calltime64call time64	
mov eax, dword ptr [esp+24h+Time+4] mov eax, dword ptr	[esp
pop ecx pop ecx	•••
mov ecx, dword ptr [esp+20h+Time] mov ecx, dword ptr	esp
sub ecx, [edi]sub ecx, [edi]	
sbb eax, [edi+4] sbb eax, [edi+4]	
cmp eax, [edi+OCh]cmp eax, [edi+OCh]	
jl short loc_417FFA jl short loc_417F8	5

🖬 🖂 🖂

🖬 🖂 🔤

CMP

ibe

loc 417FE7:

eax, eax

time64

ecx.

edi

ecx

🛄 🖂 🛄

xor

lea

mov

call

push

call

pop

jg

short loc 417FE7

ecx. [edi+8]

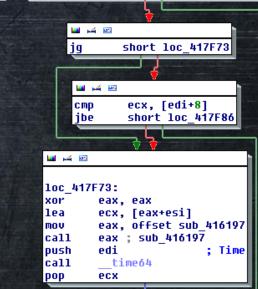
[eax+esi]

eax ; sub 416232

eax, offset sub 416232

: Time

short loc 417FFA



; Time

[esp+24h+Time], ebx

[esp+24h+Time+4], ebx

btr [esp+24h+Time+4]

| ptr [esp+20h+Time]

Dealing with Multiple Matches

Multiple matches – which is the best?

- Quality of the annotation
- Code similarity
 - Compute similarity value for pairs of inputs
 - Rank by this value, let the user choose

Similarity hashing

- Assign consecutive basic blocks to chunks
 - Fixed number of chunks ensures constant sized output
- For each chunk: compute FNV hash
- Combine FNV hashes to final hash

s(a, b) = 100 - normalized_levenshtein(simhash(a), simhash(b))

FNV Hashes

- Fast to compute
- Good Avalanche behavior
- For different word sizes

hash := FNV_BASIS
for byte in input:
 hash ^= byte
 hash *= FNV_PRIME

Similarity Hashing – Details

Basic block reordering poses challenges
 Define an order on the set of basic blocks
 Come up with a reordering resilient scheme

BB1

BB2

BB3

BB4

Fuzzy hash serves as pre-filter
 Matches are usually 100% equal
 Make fuzzy hash more fuzzy
 Position-independent representation quite strict
 Need to take instruction reordering into account

Improved algorithms in future versions

Demo Time!

Future Plans

Integration with other RE tools?

Cloud service

Social ratings of commits

Access control lists

Client

Real time notifications on updated annotations
 New and improved matching algorithms
 Ability to deal with cyclic type dependencies
 Tracking of function/file mappings
 Mass importing of common library code

Where to get it: http://crowd.re



CrowdRE makes use of Google Authentication's Service and retains only your email address and your display name. CrowdRE does not and will not store any other identifying data that may be provided by the authentication service. To create a Google account for registration purposes, please click here.

