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 CrowdREα?

- 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 CrowdRE$_\alpha$ 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

… CrowdRE α

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

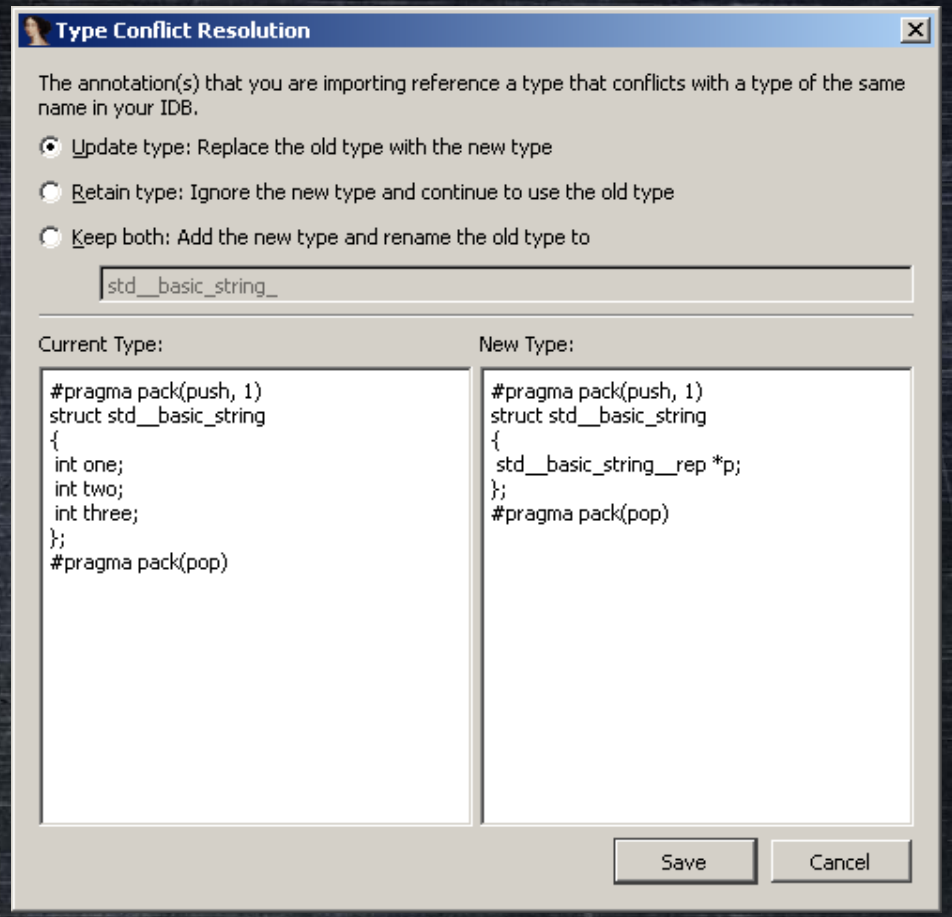
- Function prototype
  - Name
  - Calling convention
  - Return type
  - Parameter types and names
- Stack variables
- Register variables (Hex-Rays)
- Structs, enums
- Comments – IDA and Hex-Rays

```
int __cdecl main(int argc, const char **argv)
{
  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

  parameters_vector.start = 0;                  // these three NULL assigns are inlined vector::vector
  parameters_vector.finish = 0;
  parameters_vector.end_of_storage = 0;
  std::vector<std::string_std::allocator<std::string>>::reserve(&parameters_vector, argc);
  if ( argc > 0 )
  {
    i = 0;
    do
    {
      std::string::string(&tmp_string, argv[i]);
      if ( parameters_vector.finish == parameters_vector.end_of_storage )
      {
        std::vector<std::string_std::allocator<std::string>>::_M_insert_aux(
          &parameters_vector,
          parameters_vector.finish,
          &tmp_string);
      }
```

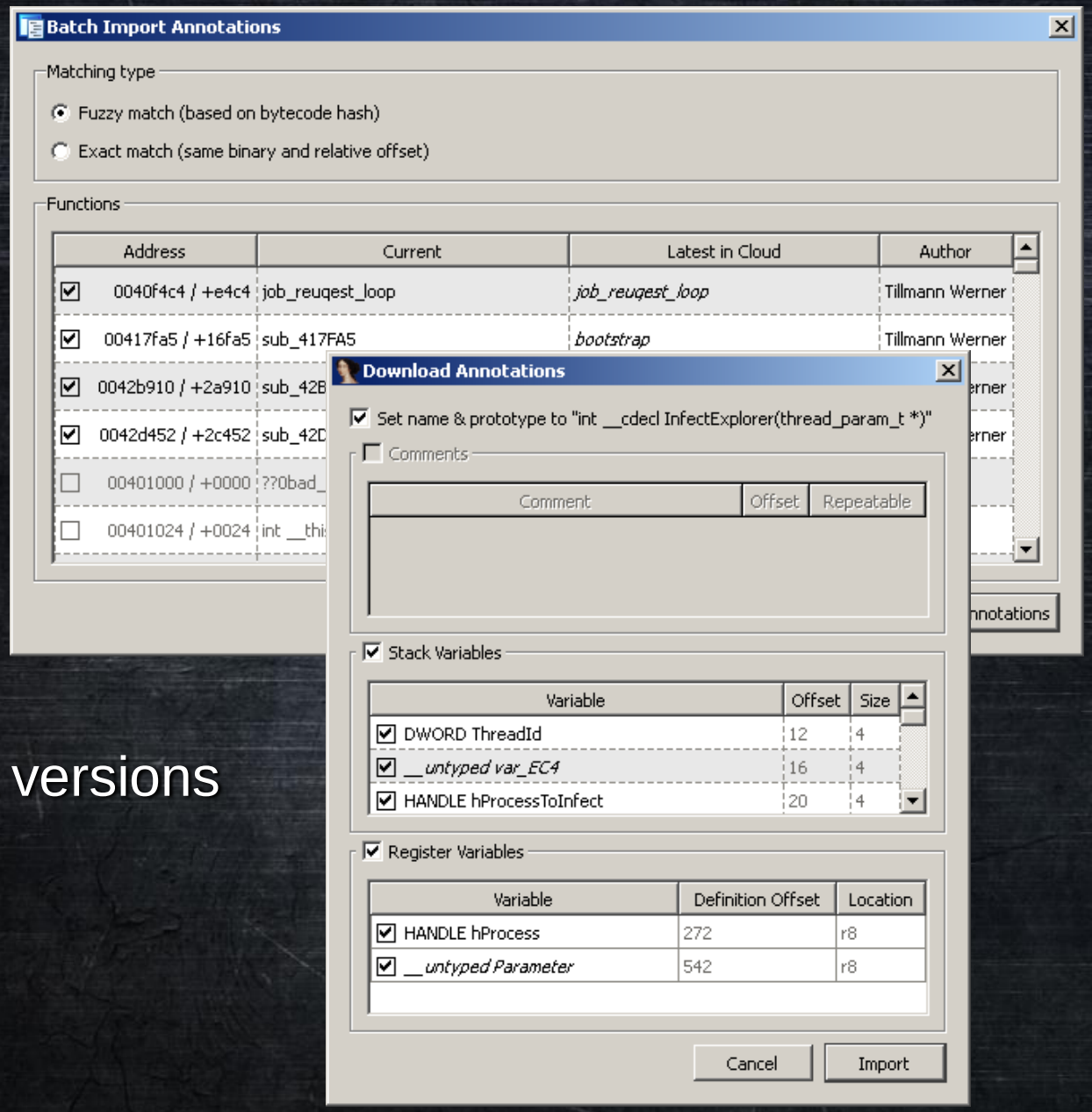
Type Information

- 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



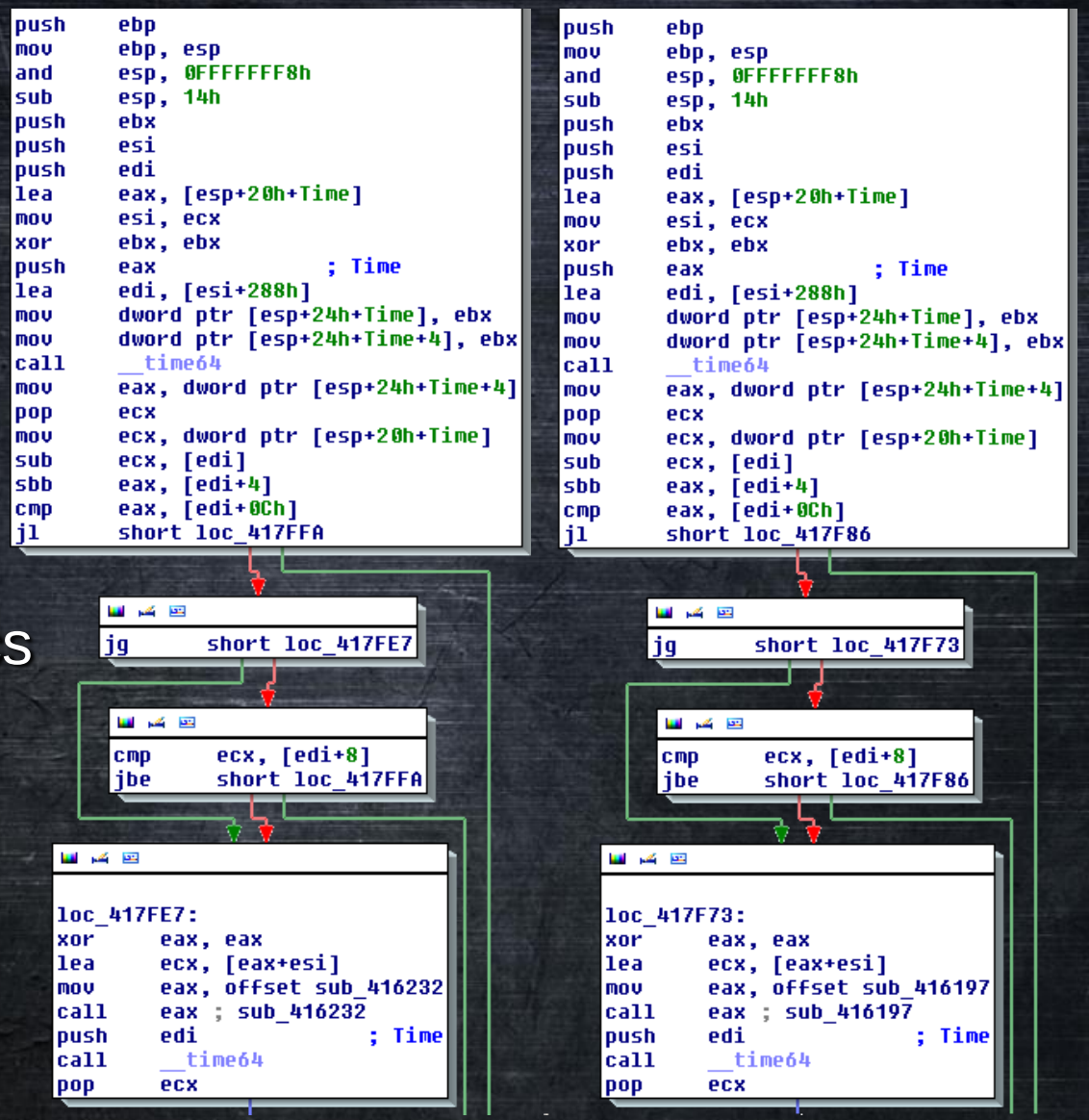
Importing Annotations

- 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



Finding Functions

- Exact matching
  - Binary's hash + function offset
- 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



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

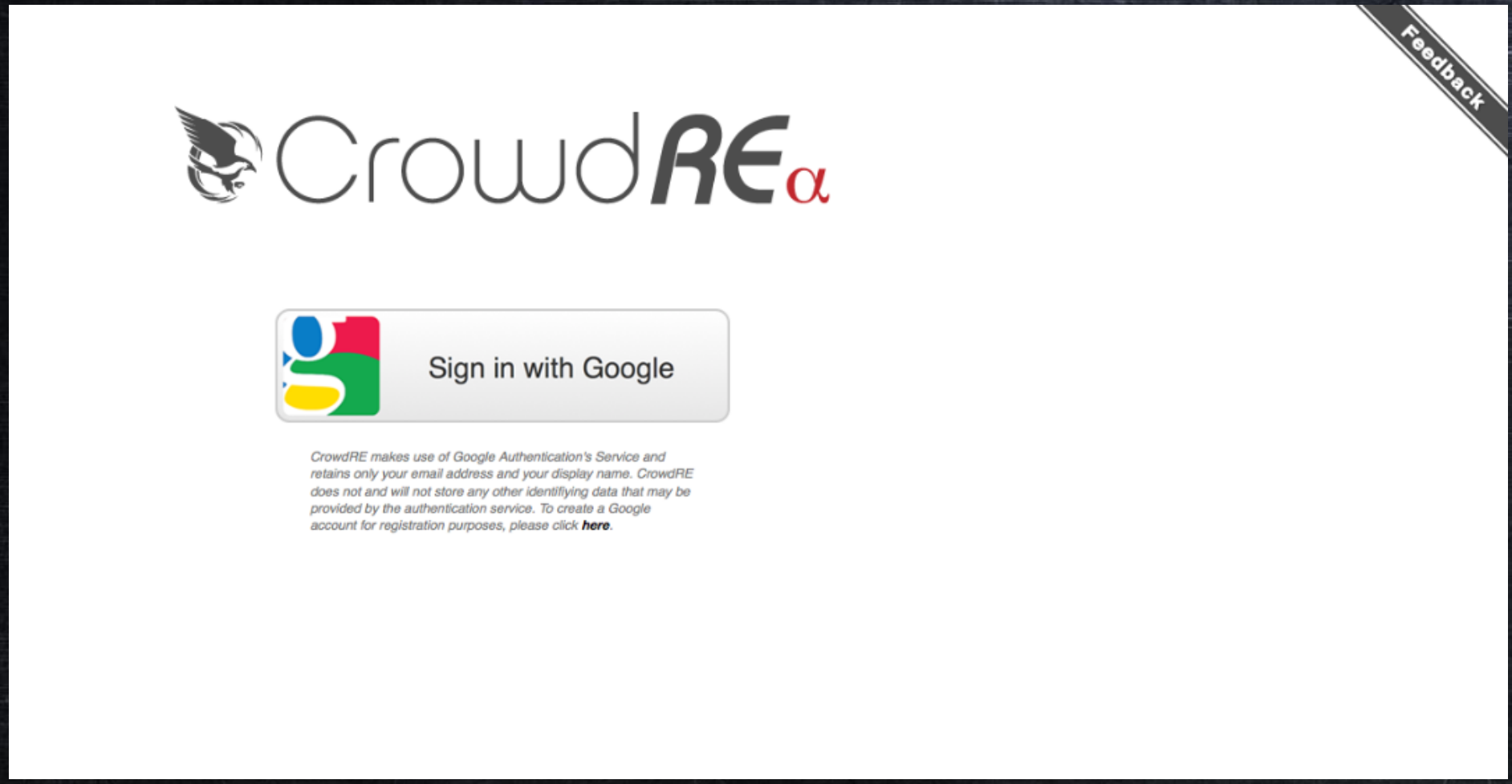
- Basic block reordering poses challenges
  - Define an order on the set of basic blocks
  - Come up with a reordering resilient scheme
- Fuzzy hash serves as pre-filter
  - Matches are usually 100% equal
  - Make fuzzy hash more fuzzy
    - Position-independent representation quite strict
    - Need 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*




CrowdStrike