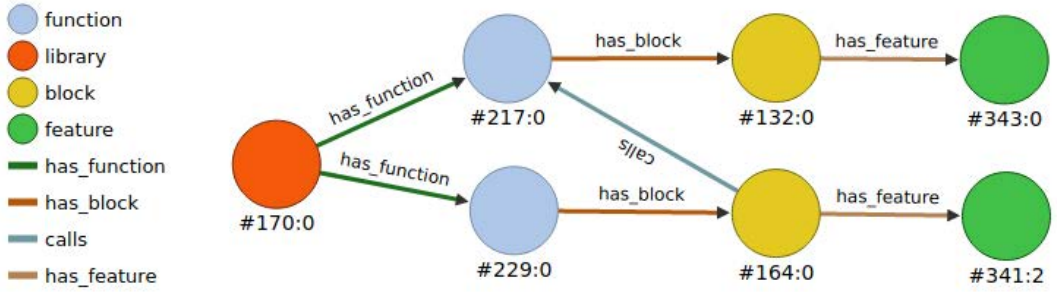
Annotated and Translated Disassembled Code (@DisCo)

@DisCo is a graph based datastore designed to organize firmware and software analysis data across builds, packages and systems, providing a highly scalable platform enabling automated binary software analysis tasks including corpora construction and storage for machine learning.

The size and complexity of the software ecosystem is a major challenge for vendors, asset owners and cybersecurity professionals who need to understand the security posture of their systems. Current tools for binary analysis and reverse engineering have no consistency, no way to capture, preserve, and share information, and are not scalable solutions. @DisCo was created as a solution to this problem. @DisCo can help provide identification of software components included in firmware, change detection, forensics, and augment the reverse engineering process. @DisCo also produces different types of corpora for large scale binary analysis and machine learning tasks which aids in supply chain management by providing a scalable method for storing program binaries in a database, recognizes vulnerable or changed binaries over time, identifies system modification, and allows for early detection of issues within the critical system.

How it works?

@DisCo uses OrientDB as the graph database storage engine and disassembles the binary using the python framework, angr. Once the binary is disassembled and analyzed, functions, blocks, and features are stored in the graph database, creating a complete and queryable control flow graph. Each library is made up of many functions and each function is made up of blocks of code. The code in each block is taken down to the intermediate language, in angr's case, Vex. The feature set of each block is a count of the unique Vex



Caption: The graphical view of a simple "Hello World" program in @DisCo.

Idaho National Laboratory developed @DisCo for the DOE-CESER-CEDS funded Firmware Indicator Translation (FIT) project. The FIT project seeks to provide scalable tools for novel and efficient binary analysis. FIT currently uses @DisCo with advanced machine learning techniques to minimize reverse engineering efforts.

The @Disco application is also being utilized by other DOE projects, Geo Threat Observables (GTO), Grid Modernization Laboratory Consortium (GMLC), Firmware Command and Control, and Deep Learning Malware.

www.inl.gov

Idaho National Laboratory

X86 Assembly

```
main:
nop
push
mov
lea
mov
call
mov
pop
retn
edx, edi
ebp
ebp, esp
edi,[0x0040115d] {"hello world"}
eax, 0x0
printf
eax, 0x0
ebp
```

Vex Intermediate Representation

```
---- IMark(0x401149, 4, 0) ----
---- IMark(0x40114d, 1, 0) ----
t0 = GET:I64(offset=56)
t1 = GET:I64(offset=48)
t2 = Sub64(t1,0x00000008)
PUT(offset=48) = t2"
STle(t2) = t
---- IMark(0x40114e, 3, 0) ----
PUT(offset=56) = t2
---- IMark(0x401151, 7, 0) ----
PUT(offset=72) = 0x00402004
---- IMark(0x401158, 5, 0) ----
PUT(offset=16) = 0x00000000
PUT(offset=184) = 0x0040115d
---- IMark(0x40115d, 5, 0) ----
t3 = Sub64(t2,0x0000000008)
PUT(offset=48) = t3
STle(t3) = 0x00401162
t4 = Sub64(t3,0x00000080)
==== AbiHint(0xt4, 128, 0x00401050) ====
```

Feature Set

Ist_imark	6
Ist_abihint	1
Ist_rdtmp	9
Ist_wrtmp	5
Ist_store	2
Ist_put	6
Iex_get	2
Iex_const	7
Iex_binop	3

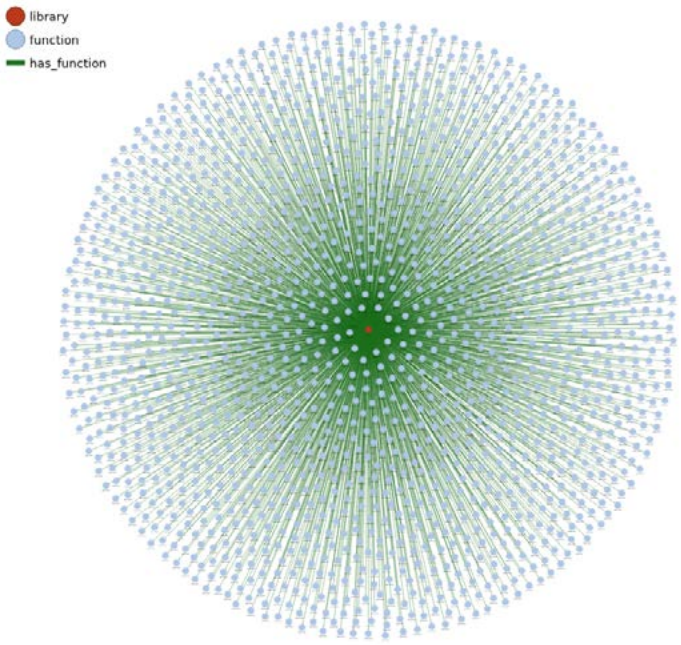
Caption: Progression of program from disassembly to a feature set

instructions within the code block. This feature set creates a vector representation of the block. Each block of code is represented by a vector, and each function is represented by a group of its block feature sets. The top-level binary is then described by each function's vector set, creating a machine learning and binary analysis prepared corpora. The graph database then can be used to capture additional data about the binary, preserve the state of the binary at specific times and is easily sharable.

The feature set of each block is the count of the Vex instructions creating the code. The example above shows the x86 disassembly of a simple "Hello World" program in C, the Vex representation, and finally the feature set for the block of code. For brevity's sake, unused instructions were omitted from the feature set example. @DisCo's use of an intermediate language enables compilation and architecture independent analysis, perfectly suited for building a machine learning corpora.

Use Cases

@DisCo was created under the Firmware Indicator Translation project, whose main goal was to identify ubiquitous third-party libraries in compiled executables. @DisCo created the machine learning corpora for this advancement. Using @DisCo's large-scale corpora of system binaries and third-party libraries, FIT utilized supervised machine learning to identify unknown function libraries used within a binary, by comparing them to third-party libraries found within the corpora. Using @DisCo in this manner created new opportunities for scalable binary analysis and machine learning.



Caption: This image captures all the functions of a single library

For more information

Jed Haile
Senior Cyber Security Researcher & Developer
208-526-4616

Sage Havens
Cyber Security Researcher
208-526-3901

Rita Foster
Infrastructure Security Strategic Advisor
208-526-3179

www.inl.gov

A U.S. Department of Energy National Laboratory

20-50134-R3