Finding Bugs Using Your Own Code:

Detecting Functionally-similar yet Inconsistent Code

Mansour Ahmadi, Reza Mirzazade farkhani, Ryan Williams, Long Lu

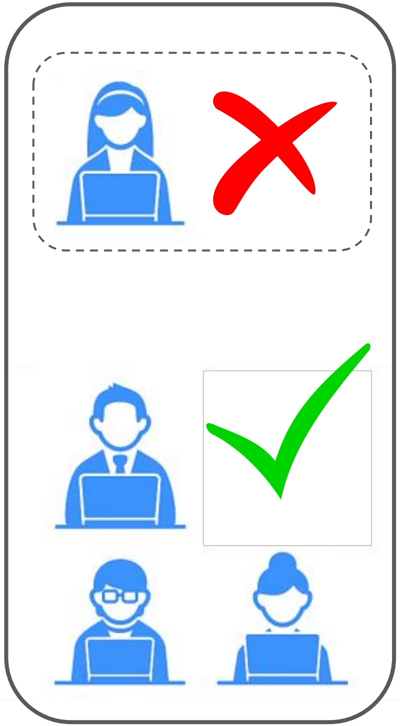
Northeastern University

usenix THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

30TH USENIX SECURITY SYMPOSIUM

Code Inconsistencies

- ➔ **Large** software ⟶ ⇧ Number of developers
  - ◆ <u>Different</u> implementation of the <u>same/similar</u> functionalities
  - ◆ Ex. We found null dereference bugs in OpenSSL because of the inconsistent ways that developers handle null check
- ➔ Inconsistent bug patch
  - ◆ Bug fix is applied <u>only</u> to where the bug was <u>originally</u> discovered
  - ◆ Ex. We found similar missing check bugs to a bug in LibTIFF that had been patched 4 years ago



Existing Solutions

Identifying specific types of coding inconsistencies:

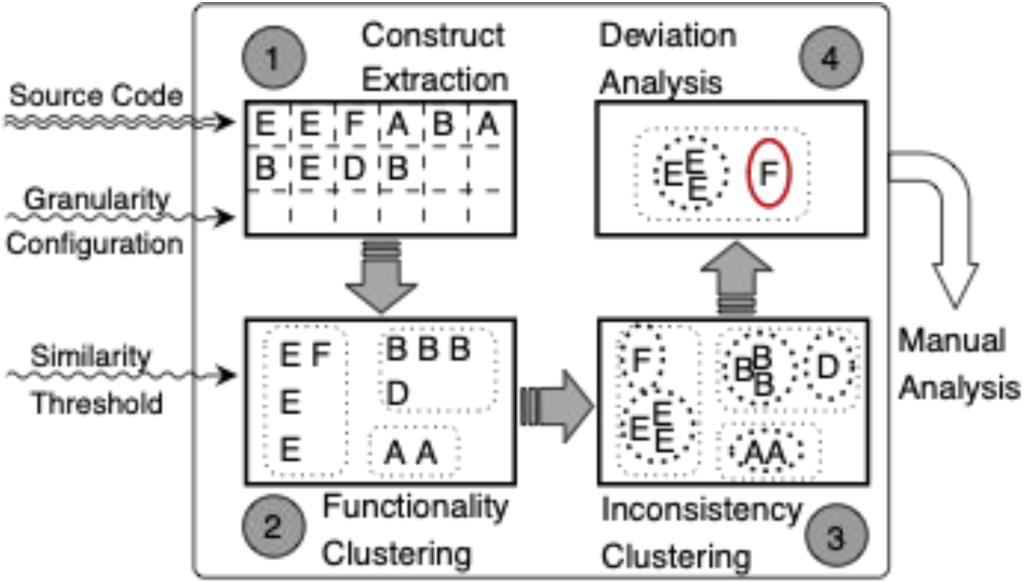
- **Seminal Work:** Bugs as Deviant Behavior [NSDI2001]
- **APIsan, AntMiner, NAR-Miner**: Detect API misuses.
- **Crix, Chucky, LRSan**: Detect missing check inconsistencies

Two **limitations**:

1. They cannot be easily extended to detect inconsistencies in an inconsistency type-agnostic fashion
2. The majority voting-based approach cannot detect one-to-one inconsistencies

Our Solution (FICS)

- FICS is not specific to one or a few types of inconsistencies or bugs
- FICS captures one-to-one inconsistencies

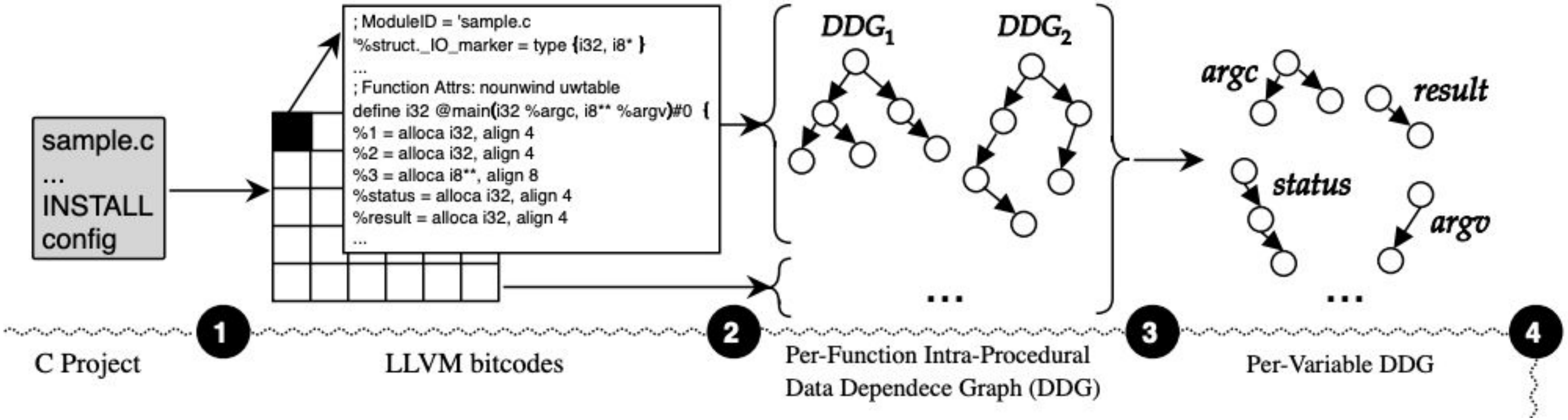


What challenges do we solve?

1. **How to finding proper code granularities?**
   a. **Intra-procedural granularity**
      i. Security-related bugs and patches are often regional or contained in a sub-function scope
   b. **Data Dependence Graph**
      i. Are usually enough to capture the root cause of a wide range of bugs.

2. **How to make the approach scalable?**
   a. **Coarse-grained graph embedding**
      i. Efficient first step clustering (coarse-grained)

FICS Design (A. Abstract Construct Extraction)



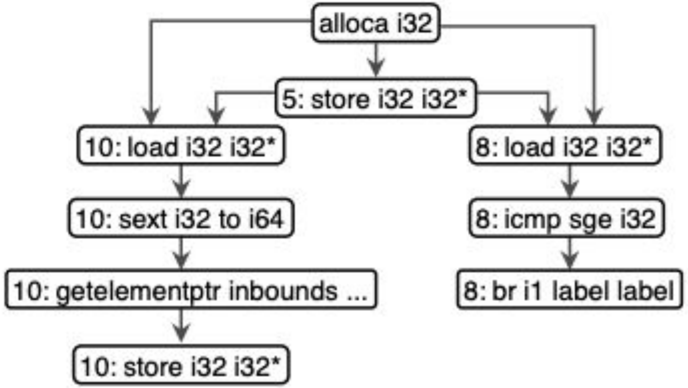
1. Get compilation database
2. Data Dependence Graph (DDG) extraction
3. Construct Extraction (Per variable & Per variable Per Basic Block)
4. Abstraction

Construct definition & example

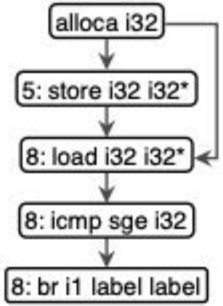
- Traverse the DDG until all subsequent nodes are covered or the Construct max-depth is reached.
- Any variable used in a function can be selected as the root variable for extracting a Construct.

```
5   int data=10;
6   int i;
7   int buffer
        [10] = {0};
8   if (data >= 0)
9   {
10    buffer[data] = 1;
11  }
```

(a) C code example.

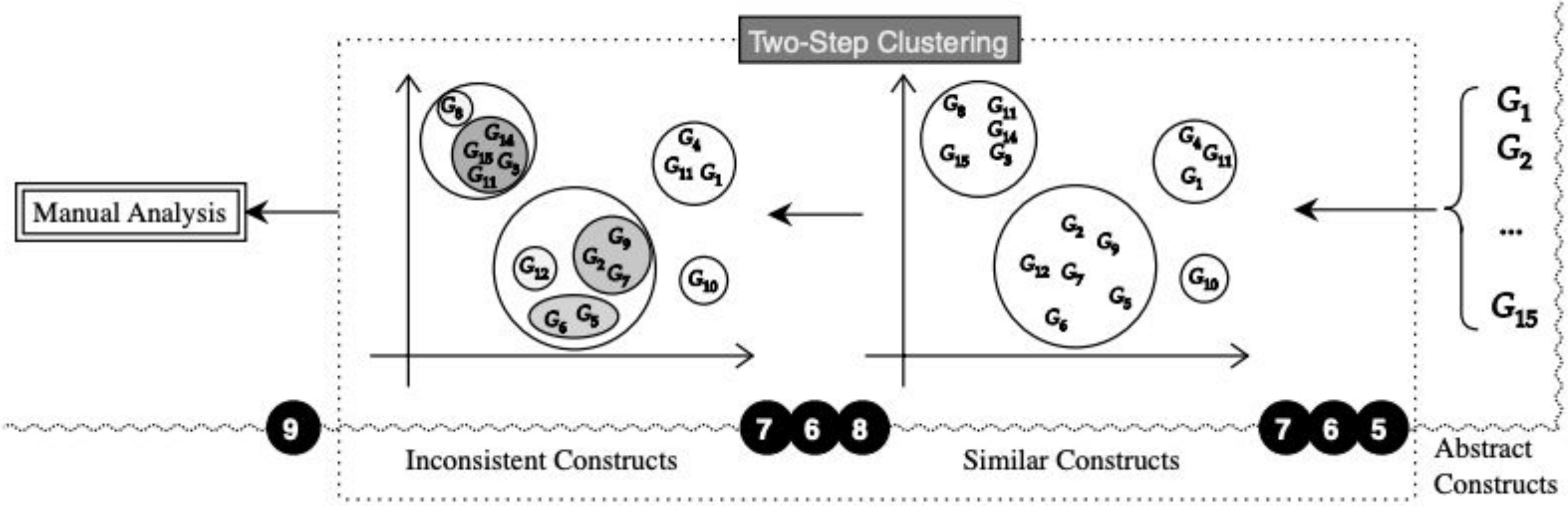


(b) Data Dependence Graph of 'data' Variable.



(c) Data Dependence Graph of 'data' Variable for the first basic block.

FICS Design (B. Finding Similar constructs)



First-step clustering

5. Bag-of-~~words~~nodes (Ignore edges)
6. Cosine similarity between graph embeddings
7. Clustering

Similarity between Bag-of-Nodes embedding

Cosine similarity between the bag-of-nodes embeddings of the correct and the buggy (inconsistent) Constructs.

```
int data = 10; int i;
int buffer[10] = { 0 };
if (data >= 0 && data < 10 ) {
buffer[data] = 1;
for(i = 0; i < 10; i++)
  printIntLine(buffer[i]);   }
```

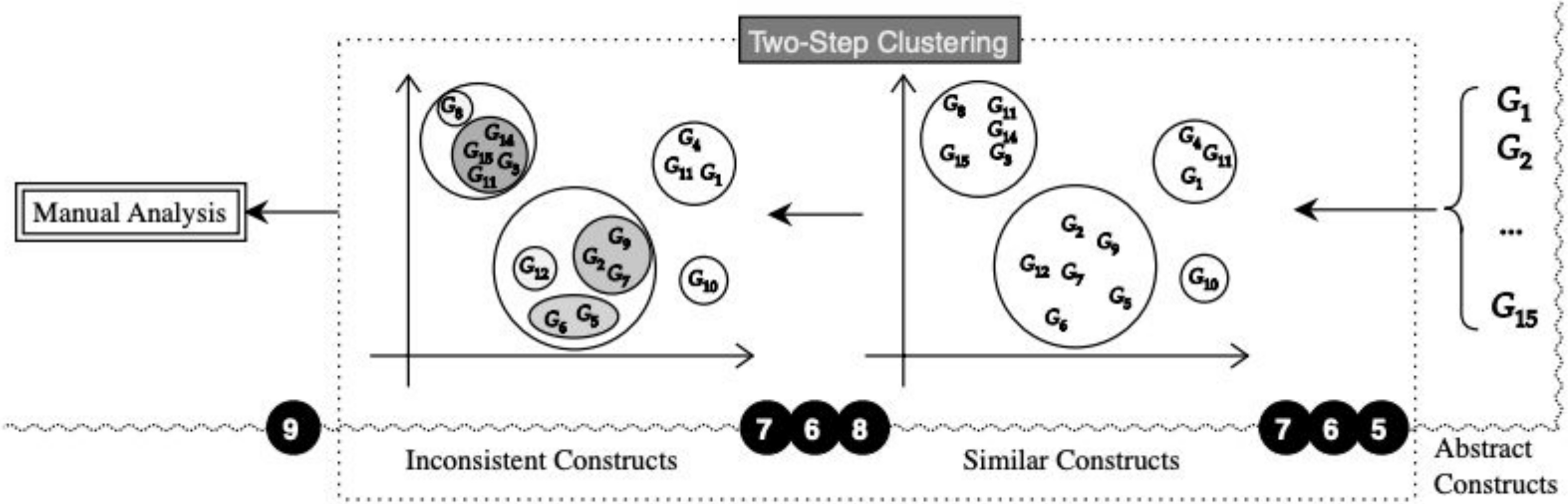
Embedding	alloc...	getelem...	icmp sge...	icmp slt...	load...	sext...	br...	call...	store...
Correct	1	1	1	1	4	1	2	1	3
Buggy	1	1	1	0	3	1	1	1	3

Cosine Similarity: 0.96609 (*i.e.*,>95%)

The buggy construct has :

- One ***load*** and one ***br*** LLVM instruction less
- No ***icmp*** LLVM instruction

FICS Design (C. Finding inconsistent constructs)



First-step clustering

8. Graph2vec (Embed the graph by random walk and Skipgram model)

6. Cosine similarity between graph embeddings
7. Custering

FICS Design (D. Deviation analysis & Filtering)

- **Deviation Analysis**

  Red refers to LLVM instruction

  Orange refers to function call

  ‘*’ means Kleene Star

Inconsistency Type Deviation	Bug Category
Check icmp Node	NULL Pointer Dereference, Undefined Behavior Buffer Errors , Integer Overflow
Memory Handling *free*, *close* Nodes *bzero*, *clear* Nodes	Resource Leak, Double Free Information Leak
Type trunc, bitcast Nodes	Bad Casting
Order Edge	Wrong Order of Operations
Initialization store, memset Nodes	Double Free, Information Leak

- **Filtering**

  removes the inconsistencies that are redundant or likely false.

Evaluation

- Evaluated on five real codebases
- Found 218 valid inconsistencies
    - 123 Potential bugs (22 confirmed so far)
    - 95 Code smells

Name	Total	# Reported inconsistencies Check + Call (Sum)	After Filtering	Valid Cases	Code Smells	Potential Bugs	Confirmed Bugs
QEMU	12,320	3,907 + 3,170 (7,077)	1,206	79	26	53	4
OpenSSL	2,419	1,158 + 347 (1,505)	310	59	24	35	9
wolfSSL	586	296 + 124 (420)	91	23	18	5	3
OpenSSH	1,063	509 + 208 (717)	121	29	18	11	1
LibTIFF	925	390 + 156 (546)	93	28	9	19	5
Total	17,313	6,260 + 4,005 (10,265)	1,821	218	**95**	**123**	**22**

Comparison

- They focus on specific class of inconsistencies or bugs
- They cannot detect one-to-one inconsistencies

	FICS		APIsan		LRSan		Crix	
	#Rep	#B	#Rep	#B	#Rep	#B	#Rep	#B
QEMU	1,206	4	5,805	0	129	0	98	0
OpenSSL	310	9	7,874	0	30	0	54	1
wolfSSL	91	3	1,049	1	62	0	62	0
OpenSSH	121	1	2,740	0	0	0	5	0
LibTIFF	93	5	645	3	12	1	3	0

Table 7: Comparison between FICS, APIsan, LRSan, and Crix on bug detection capability. FICS outperforms its competitors while not reporting too many potential cases. #Rep: Number of reports, #B: Number of bugs.

Summary

- FICS is the first inconsistency-generic, ML-based bug detection system
- FICS does not require external datasets for training nor is limited to certain types of bugs.
- FICS found 22 unknown bugs in 5 popular and well-tested projects

- Limitations
  - If the size of the codebase is too small, the system is less likely to find bugs
  - Certain bugs (e.g., one-liners) may be too small to be captured by FICS
  - Our research prototype currently neither support C++ nor extremely large codebases (e.g., Linux)

Thank you! Q&A

Contact: mansosec@gmail.com

Code: https://github.com/RiS3-Lab/FICS

```
Inconsistency #57 ,  Total subclusters: (3) ,  Granularity: (afs.bb1) ,  Similarity: (0.95) ,  ID: 5dbb7469b6564a0bd8dbec60 ,  Dependency: Data Flow
================================================================================
Subcluster #0 ,  Total items: (1)
--------------------------------------------------
/home/mansour/nfs/data/bcs/openssl-a75be9f/crypto/cms/cms_pwri.c/pdg/_ODD_cms_RecipientInfo_pwri_crypt.pdg_ARG_cms_0x1c6e5a0.afs.bb1.dot
/home/mansour/nfs/data/projects/openssl-a75be9f/crypto/cms/cms_pwri.c (cms_RecipientInfo_pwri_crypt) (ARG_cms)
376: ec->key = key;
377: ec->keylen = keylen;
--------------------------------------------------
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Subcluster #1 ,  Total items: (4)
--------------------------------------------------
/home/mansour/nfs/data/bcs/openssl-a75be9f/crypto/cms/cms_enc.c/pdg/_ODD_cms_EncryptedContent_init_bio.pdg_LCL_tkey_0x293aba0.afs.bb1.dot
/home/mansour/nfs/data/projects/openssl-a75be9f/crypto/cms/cms_enc.c (cms_EncryptedContent_init_bio) (LCL_tkey)
119: /* Use random key */
120: OPENSSL_clear_free(ec->key, ec->keylen);
121: ec->key = tkey;
122: ec->keylen = tkeylen;
123: tkey = NULL;
124: ERR_clear_error();
--------------------------------------------------
/home/mansour/nfs/data/bcs/openssl-a75be9f/crypto/cms/cms_env.c/pdg/_ODD_cms_RecipientInfo_ktri_decrypt.pdg_LCL_ec_0x21c57c0.afs.bb1.dot
/home/mansour/nfs/data/projects/openssl-a75be9f/crypto/cms/cms_env.c (cms_RecipientInfo_ktri_decrypt) (LCL_ec) [411, 412, 413]
/home/mansour/nfs/data/bcs/openssl-a75be9f/crypto/cms/cms_enc.c/pdg/_ODD_cms_EncryptedContent_init_bio.pdg_ARG_ec_0x293aba0.afs.bb1.dot
/home/mansour/nfs/data/projects/openssl-a75be9f/crypto/cms/cms_enc.c (cms_EncryptedContent_init_bio) (ARG_ec) [120, 121, 122]
/home/mansour/nfs/data/bcs/openssl-a75be9f/crypto/cms/cms_kari.c/pdg/_ODD_CMS_RecipientInfo_kari_decrypt.pdg_LCL_ec_0x2293d80.afs.bb1.dot
/home/mansour/nfs/data/projects/openssl-a75be9f/crypto/cms/cms_kari.c (CMS_RecipientInfo_kari_decrypt) (LCL_ec) [245, 246, 247, 248]
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Subcluster #2 ,  Total items: (1)
--------------------------------------------------
/home/mansour/nfs/data/bcs/openssl-a75be9f/crypto/cms/cms_enc.c/pdg/_ODD_cms_EncryptedContent_init_bio.pdg_LCL_calg_0x29455f0.afs.bb1.dot
/home/mansour/nfs/data/projects/openssl-a75be9f/crypto/cms/cms_enc.c (cms_EncryptedContent_init_bio) (LCL_calg)
154: if (!keep_key || !ok) {
155: OPENSSL_clear_free(ec->key, ec->keylen);
156: ec->key = NULL;
157: }
--------------------------------------------------
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```