Neural Reverse Engineering of Stripped Binaries using Augmented Control Flow Graphs

Yaniv David, Uri Alon, Eran Yahav
Technion, Israel
Great Progress Using ML on Source Code

[“code2seq”, Alon et al., ICLR’2019]

void updateView() {
    // ...
    View v = ViewUtil.findView(name);
    if (v == null) {
        return;
    }
    data.send(((UpdatableView) v)
                .getContentMgt(), "UPDATE");
    // ...
}

[“Getafix”, Bader et al., OOPSLA’2019]

[Program graphs, Allamanis et al., ICLR’2018]

[Edit Completion, Brody et al., OOPSLA’2020]
Not a Lot of Progress for Binaries

• Why?
Programmers follow common patterns (sort, merge, ...)

These patterns employ sequences of the language’s syntactic structures

Many global anchors: types, class structures, frameworks

Source code in a high-level language

void f(int[] a) {
    boolean s = true;
    for (int i = 0; i < a.length && s; i++) {
        s = false;
        for (int j = 0; j < a.length - 1 - i; j++) {
            if (a[j] > a[j+1]) {
                int temp = a[j];
                a[j] = a[j+1];
                a[j+1] = temp;
                s = true;
            }
        }
    }
}
Disassembled Binary Code

• Machine generated, adhering to hardware specifications & limitations
• Few simple control flow instructions:
  • jump & call (+interrupt)
• Optimization causes: entangled computation flows, context dependent

Long unstructured sequence of low-level operations

...
Disassembled Binary Code

- Machine generated, adhering to hardware specifications & limitations
- Few simple control flow instructions: `jump & call (+interrupt)`
- Optimization causes: entangled computation flows, context dependent

These put a lot of pressure on the model

Long unstructured sequence of low-level operations
Key Idea: Use Binary Program Analysis to Create a Compact & Rich Representation

...\nmov rsi, rdi
mov rdx, 16
mov rax, [rbp-58h]
mov rdi, rax
mov edx, eax
mov rax, [rbp-4h]
mov rax, [rbp-58h]
mov rdi, rax
mov r8, 4
mov rdx, 10
mov esi, 0
lea rcx, [rbp-88h]
mov rdi, rax
...
Key Idea: Use Binary Program Analysis to Create a Compact & Rich Representation

Distill many computations into a single representation component

Recover structure

Recover flow
Key Idea: Use Binary Program Analysis to Create a Compact & Rich Representation

... 
mov rsi, rdi 
mov rdx, 16 
mov rax, [rbp-58h] 
mov rdi, rax 
mov edx, eax 
mov rax, [rbp-4h] 
mov rax, [rbp-58h] 
mov rdi, rax 
mov r8, 4 
mov rdx, 10 
mov esi, 0 
lea rcx, [rbp-88h] 
mov rdi, rax 
...
Key Idea: Use Binary Program Analysis to Create a Compact & Rich Representation

...  
mov rsi, rdi  
mov rdx, 16  
mov rax, [rbp-58h]  
mov rdi, rax  
mov edx, eax  
mov rax, [rbp-4h]  
mov rax, [rbp-58h]  
mov rdi, rax  
mov r8, 4  
mov rdx, 10  
mov esi, 0  
lea rcx, [rbp-88h]  
mov rdi, rax  
...
Motivating Task

Naming Procedures in Binaries
Helping Reverse Engineers
Helping Reverse Engineers

Where to start?
## Un-Stripping Procedure Names

<table>
<thead>
<tr>
<th>Function name</th>
<th>Function name</th>
</tr>
</thead>
<tbody>
<tr>
<td>sub_42FEA7</td>
<td>limit_bandwidth</td>
</tr>
<tr>
<td>sub_42F08D</td>
<td>write_data</td>
</tr>
<tr>
<td>sub_42FC0C0</td>
<td>fd_read_body</td>
</tr>
<tr>
<td>sub_42F5EA</td>
<td>fd_read_hunk</td>
</tr>
<tr>
<td>sub_42F680</td>
<td>line_terminator</td>
</tr>
<tr>
<td>sub_42F717</td>
<td>fd_read_line</td>
</tr>
<tr>
<td>sub_42F7F8</td>
<td>retr_rate</td>
</tr>
<tr>
<td>sub_42F809</td>
<td>calc_rate</td>
</tr>
<tr>
<td>sub_42F863</td>
<td>retrieve_url</td>
</tr>
<tr>
<td>sub_42F878</td>
<td>retrieve_from_file</td>
</tr>
<tr>
<td>sub_42F8CC</td>
<td>printfat</td>
</tr>
<tr>
<td>sub_42F901</td>
<td>sleep_between_retrievals</td>
</tr>
<tr>
<td>sub_42FAC8</td>
<td>free_urlpos</td>
</tr>
<tr>
<td>sub_42FDE3</td>
<td>rotate_backups</td>
</tr>
<tr>
<td>sub_42FE25</td>
<td>getproxy</td>
</tr>
<tr>
<td>sub_42FF77</td>
<td>urlUses_proxy</td>
</tr>
<tr>
<td>sub_43094</td>
<td>no_proxy_match</td>
</tr>
<tr>
<td>sub_430BE</td>
<td>set_local_file</td>
</tr>
<tr>
<td>sub_430C1</td>
<td>input_file_url</td>
</tr>
<tr>
<td>sub_430C71</td>
<td>spider_cleanup</td>
</tr>
<tr>
<td>sub_430E73</td>
<td>nonexistent_url</td>
</tr>
<tr>
<td>sub_430A13</td>
<td>print_broken_links</td>
</tr>
<tr>
<td>sub_430A88</td>
<td>url_unescape_1</td>
</tr>
</tbody>
</table>

### Diagram

```
; Attributes: bp-based frame

public free_urlpos

ptr= qword ptr -10h
var_8= qword ptr -8

push rbp
 mov rbp, rsp
 sub rsp, 20h
 mov [rbp+ptr], rdi
 jmp short loc_436C0A

loc_436C0A:
 cmp [rbp+ptr], 0
 jnz short loc_436B0B
 nop
 leave
 ret
 free_urlpos endp

loc_436B0B:
 mov rax, [rbp+ptr]
 mov rax, [rax+28h]
 mov [rbp+var_8], rax
 mov rax, [rbp+ptr]
 mov rax, [rax]
 test rax, rax
 js short loc_436BD2

loc_436BD2:
 nop
 leave
 retn
 free_urlpos endp

mov rax, [rbp+ptr]
mov rax, [rax]
call url_free
```
Un-Stripping Procedure Names

Start at the right place

Focus on the right procedure
Our Approach
Extract Paths From the CFG

Extract simple paths (no loops)
Using API Calls

API calls

... 
call socket 
... 
call setsockopt 
... 
call connect 
... 

Reconstructed API Call Sites

connect(rdi, rsi, rdx)

Library information + Calling Conventions
Augmenting Call Sites

call socket(...)
mov [rbp-58h], rax
mov rax, [rbp-58h]
mov rdi, rax

connect(rdi,rsi,rdx)

mov [rbp-50],rdi
mov rdi, [rbp-50]
mov rsi, rdi

mov rdx, 16

In the C code:
connect(sock,addr,16)
Using Concrete or Abstracted Values

1. Concrete value (Integer, Enum, String)
2. ARG – procedure argument
3. GLOBAL - pointer to a global variable
4. RET – a return value from a call
5. STACK – pointer to stack memory
6. Ø – no information
Pointer-Aware Slicing of Call Site Arguments

```
connect(rdi, rsi, rdx)
```

```
mov rdi, [rbp-50h]
```

```
mov rsi, rdi
```

```
mov [rbp-50h], rdi
```

```
rbp
```

```
rdi
```

```
∅
```

```
V
```

```
P
```

```
V(rbp)
```

```
P(rbp-50h)
```

```
V
```

```
P
```

```
V
```

```
P
```

```
V
```

```
P
```

```
V
```

```
P
```

```
V
```

```
P
```
Augmenting Call Site Arguments

```
connect(rdi, rsi, rdx)

mov rsi, rdi
mov rdi, [rbp - 50h]
mov rsi, rdi
```

Diagram:

- STK
- ARG
- ∅
Augmenting Call Site Arguments

Using concrete or abstracted values:
1. Concrete value (Integer, Enum, String)
2. ARG – procedure argument
3. GLOBAL - pointer to a global variable
4. RET – a return value from a call
5. STACK – pointer to stack memory
Augmenting Call Site Arguments

```
connect(rdi, rsi, rdx)
```

```
connect(rdi, rsi, rdx)
```

```
mov rdi, [rbp - 50h]
```

```
mov [rbp - 50h], rdi
```

```
mov rsi, rdi
```

```
mov rdi, [rbp - 50h]
```

```
rdb ARG | \emptyset |]
```

```
mov ARG | \emptyset |]
```

```
mov ARG | \emptyset |]
```

```
ARG
```

```
STK
```

```
∅
```

```
ARG
```

```
ARG
```

```
ARG
```

```
STK
```

```
ARG
```

```
∅
```

```
∅
```

```
ARG
```

```
STK
```

```
ARG
```

```
ARG
```

```
STK
```

```
ARG
```

```
ARG
```

```
ARG
```

```
ARG
```

```
STK
```

```
ARG
```

```
ARG
```

```
ARG
```

```
ARG
```

```
ARG
```

```
ARG
```

```
ARG
```

```
ARG
```

```
ARG
```

```
ARG
```

```
ARG
```

```
ARG
```

```
ARG
```

```
ARG
```

```
ARG
```

```
ARG
```

```
ARG
```

```
ARG
```

```
ARG
```
Augmenting Call Site Arguments

\[ \text{connect}(\text{rdi}, \text{ARG}, \text{rdx}) \]
Augmented Control Flow Graph

Place augmented call sites back in the CFG

LSTM & Transformer

GNN

setsockopt(RET,0,10,STK,4)
socket(2,1,0)
connect(RET,ARG,16)
printf(GLOBAL, …)
close(…)
printf(GLOBAL, …)
socket(2,1,0)
setsockopt(RET,0,10,STK,4)
connect(RET,ARG,16)
close(…)
printf(GLOBAL, …)
Evaluation
Implementation: Nero

https://github.com/tech-srl/Nero
Evaluation Corpus

GNU software repository

Cleanup

67,246 Labeled Procedures

Strip & Obfuscate APIs

Strip

8:1:1 Package-Based Split

https://doi.org/10.5281/zenodo.4081641
GNU software repository

Cleanup

setsockopt(RET, 0, 10, STK, 4)

UnknownExternal(RET, 0, 10, STK, 4)

Strip & Obfuscate APIs

67,246 Labeled Procedures

8:1:1 Package-Based Split

https://doi.org/10.5281/zenodo.4081641
Evaluation Results

Nero-GNN

Nero-Transformer

Nero-LSTM

DIRE [Lacomis et al. 2019]

Debin [He et al. 2018]

LSTM-text

Transformer-text
Evaluation Results

Nero-GNN

Nero-Transformer

Nero-LSTM

DIRE [Lacomis et al. 2019]

Debin [He et al. 2018]

LSTM-text

Transformer-text

“Debin: Predicting Debug Information in Stripped Binaries”, CCS’ 18

“DIRE: A Neural Approach to Decompiled Identifier Naming”, ASE ’19
Evaluation Results

F1 Score

Nero-GNN
Nero-Transformer
Nero-LSTM
DIRE [Lacomis et al. 2019]
Debin [He et al. 2018]
LSTM-text
Transformer-text

"Debin: Predicting Debug Information in Stripped Binaries", CCS’ 18
"DIRE: A Neural Approach to Decompiled Identifier Naming", ASE ’19
Evaluation Results

![Bar chart showing evaluation results for different models.]

- **Nero-GNN**: 45.53
- **Nero-Transformer**: 40.04
- **Nero-LSTM**: 39.4
- **DIRE [Lacomis et al. 2019]**: 35.52
- **Debin [He et al. 2018]**: 38.83
- **LSTM-text**: 21.72
- **Transformer-text**: 19.64

**Legend**:
- Stripped
- Stripped & Obfuscated API calls

"Debin: Predicting Debug Information in Stripped Binaries“, CCS’ 18
"DIRE: A Neural Approach to Decompiled Identifier Naming“, ASE ’19
Ablation Study

Paths → Augmented Call Sites → LSTM

Paths → Only Calls → LSTM

Augmented Call Sites → LSTM

Only Calls → LSTM

F1 Score

39.4

33.77

26.65

24.04
Ablation Study

Paths → Augmented Call Sites → LSTM

Paths → Only Calls → LSTM

Augmented Call Sites → LSTM

Only Calls → LSTM

F1 Score

0 5 10 15 20 25 30 35 40 45

24.04

39.4

call getaddrinfo...

call strerror...

call setsockopt...
Ablation Study

- Paths → Augmented Call Sites → LSTM: F1 Score = 39.4
- Paths → Only Calls → LSTM: F1 Score = 26.65
- Augmented Call Sites → LSTM: F1 Score = 33.77
- Only Calls → LSTM: F1 Score = 24.04
## Qualitative Evaluation

<table>
<thead>
<tr>
<th>Error Type</th>
<th>Ground Truth</th>
<th>Predicted Name</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Programmers Vs English Language</strong></td>
<td>i18n_initialize</td>
<td>i18n_init</td>
</tr>
<tr>
<td></td>
<td>split_cfg_path</td>
<td>split_config_path</td>
</tr>
<tr>
<td></td>
<td>add_env_opt</td>
<td>add_option</td>
</tr>
<tr>
<td><strong>Date Structure Name Missing</strong></td>
<td>get_best_speed</td>
<td>get_list_item</td>
</tr>
<tr>
<td></td>
<td>ftp_parse_winnt_ls</td>
<td>parse_tree</td>
</tr>
<tr>
<td></td>
<td>abort_gzip_signal</td>
<td>fatal_signal_handler</td>
</tr>
<tr>
<td><strong>Verb Replaced</strong></td>
<td>read_units</td>
<td>parse</td>
</tr>
<tr>
<td></td>
<td>retrieve_from_file</td>
<td>get_from_file</td>
</tr>
<tr>
<td></td>
<td>display_help</td>
<td>show_help</td>
</tr>
</tbody>
</table>
## Qualitative Evaluation

<table>
<thead>
<tr>
<th>Error Type</th>
<th>Ground Truth</th>
<th>Predicted Name</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Programmers Vs English Language</strong></td>
<td>i18n_initialize</td>
<td>i18n_init</td>
</tr>
<tr>
<td></td>
<td>split_cfg_path</td>
<td>split_cfg_path</td>
</tr>
<tr>
<td></td>
<td>add_env_opt</td>
<td>add_env_opt</td>
</tr>
<tr>
<td><strong>Date Structure Name Missing</strong></td>
<td>get_best_speed</td>
<td>get_list_item</td>
</tr>
<tr>
<td></td>
<td>ftp_parse_winnt_ls</td>
<td>parse_tree</td>
</tr>
<tr>
<td></td>
<td>abort_gzip_signal</td>
<td>fatal_signal_handler</td>
</tr>
<tr>
<td><strong>Verb Replaced</strong></td>
<td>read_units</td>
<td>parse</td>
</tr>
<tr>
<td></td>
<td>retrieve_from_file</td>
<td>get_from_file</td>
</tr>
<tr>
<td></td>
<td>display_help</td>
<td>show_help</td>
</tr>
</tbody>
</table>

**Measured F1 is actually a lower-bound**
Takeaway Messages

Augmented call sites serve as a strong basis for procedure representations

call socket(...)
mov [rbp-58h], rax
mov rax, [rbp-58h]
mov rdi, rax

connect(rdi,rsi,rdx)

mov rdx, 16

In the C code:
connect(sock,addr,16)

Reconstructing the CFG enables the use of seq2seq and GNN models

Place Augmented Call sites back on the CFG

LSTM & Transformer

GNN

https://github.com/tech-srl/Nero