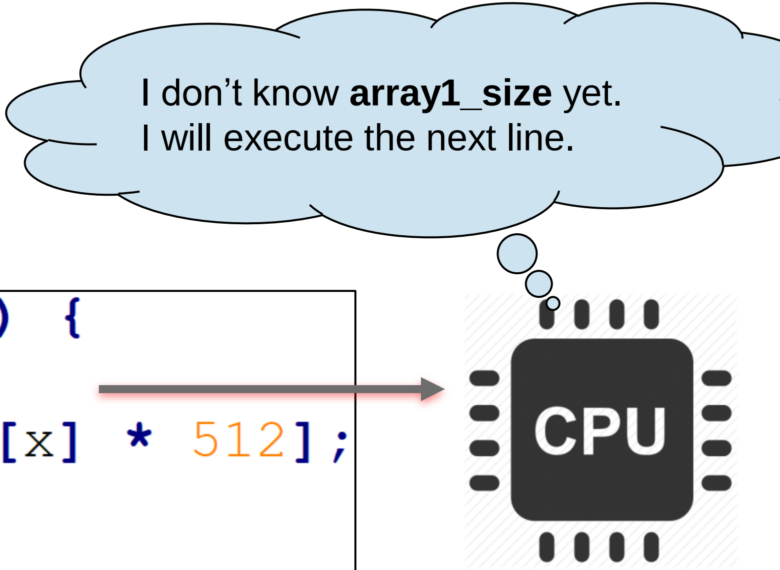
FastSpec: Scalable Generation and Detection of Spectre Gadgets Using Neural Embeddings

M. Caner Tol[1], Berk Gulmezoglu[2], Koray Yurtseven[1], and Berk Sunar[1]

[1] Worcester Polytechnic Institute
[2] Iowa State University

Speculative Execution

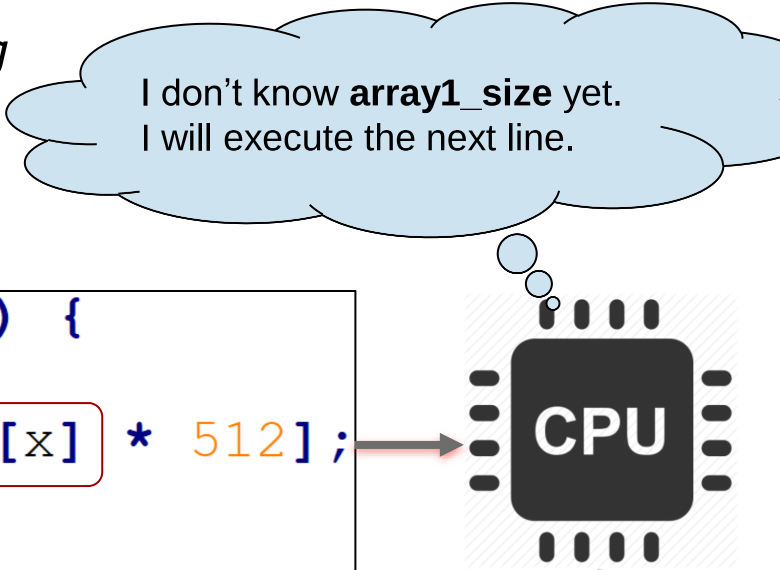
- array1_size not cached




```
void user_function_v01(size_t x) {
    if (x < array1_size) {
        temp &= array2[array1[x] * 512];
    }
}
```

Spectre Attack (v1)

- Kocher et al, *Spectre Attacks: Exploiting Speculative Execution*, S&P ’19
- Mitigation: Software Updates




```
void user_function_v01(size_t x) {
    if (x < array1_size) {
        temp &= array2[array1[x] * 512];
    }
}
```

Problem 1: What do they look like?

```c
__declspec(noinline) void leakByteNoinlineFunction(uint8_t k) { temp &= array2[(k)* 512]; }
void victim_function_v03(size_t x) {
    if (x < array1_size)
        leakByteNoinlineFunction(array1[x]);
}
```

```c
__inline int is_x_safe(size_t x) { if (x < array1_size) return 1; return 0; }
void victim_function_v13(size_t x) {
    if (is_x_safe(x))
        temp &= array2[array1[x] * 512];
}
```

```c
void victim_function_v01(size_t x) {
    if (x < array1_size) {
        temp &= array2[array1[x] * 512];
    }
}
```

```c
void victim_function_v04(size_t x) {
    if (x < array1_size)
        temp &= array2[array1[x << 1] * 512];
}
```

```c
void victim_function_v15(size_t *x) {
    if (*x < array1_size)
        temp &= array2[array1[*x] * 512];
}
```

```c
void victim_function_v07(size_t x) {
    static size_t last_x = 0;
    if (x == last_x)
        temp &= array2[array1[x] * 512];
    if (x < array1_size)
        last_x = x;
}
```

```c
void victim_function_v11(size_t x) {
    if (x < array1_size)
        temp = memcmp(&temp, array2 + (array1[x] * 512), 1);
}
```

```c
void leakByteLocalFunction_v02(uint8_t k) { temp &= array2[(k)* 512]; }
void victim_function_v02(size_t x) {
    if (x < array1_size) {
        leakByteLocalFunction(array1[x]);
    }
}
```

```c
void victim_function_v14(size_t x) {
    if (x < array1_size)
        temp &= array2[array1[x ^ 255] * 512];
}
```

```c
void victim_function_v08(size_t x) {
    temp &= array2[array1[x < array1_size ? (x + 1) : 0] * 512];
}
```

```c
void victim_function_v12(size_t x, size_t y) {
    if ((x + y) < array1_size)
        temp &= array2[array1[x + y] * 512];
}
```

```c
void victim_function_v06(size_t x) {
    if ((x & array_size_mask) == x)
        temp &= array2[array1[x] * 512];
}
```

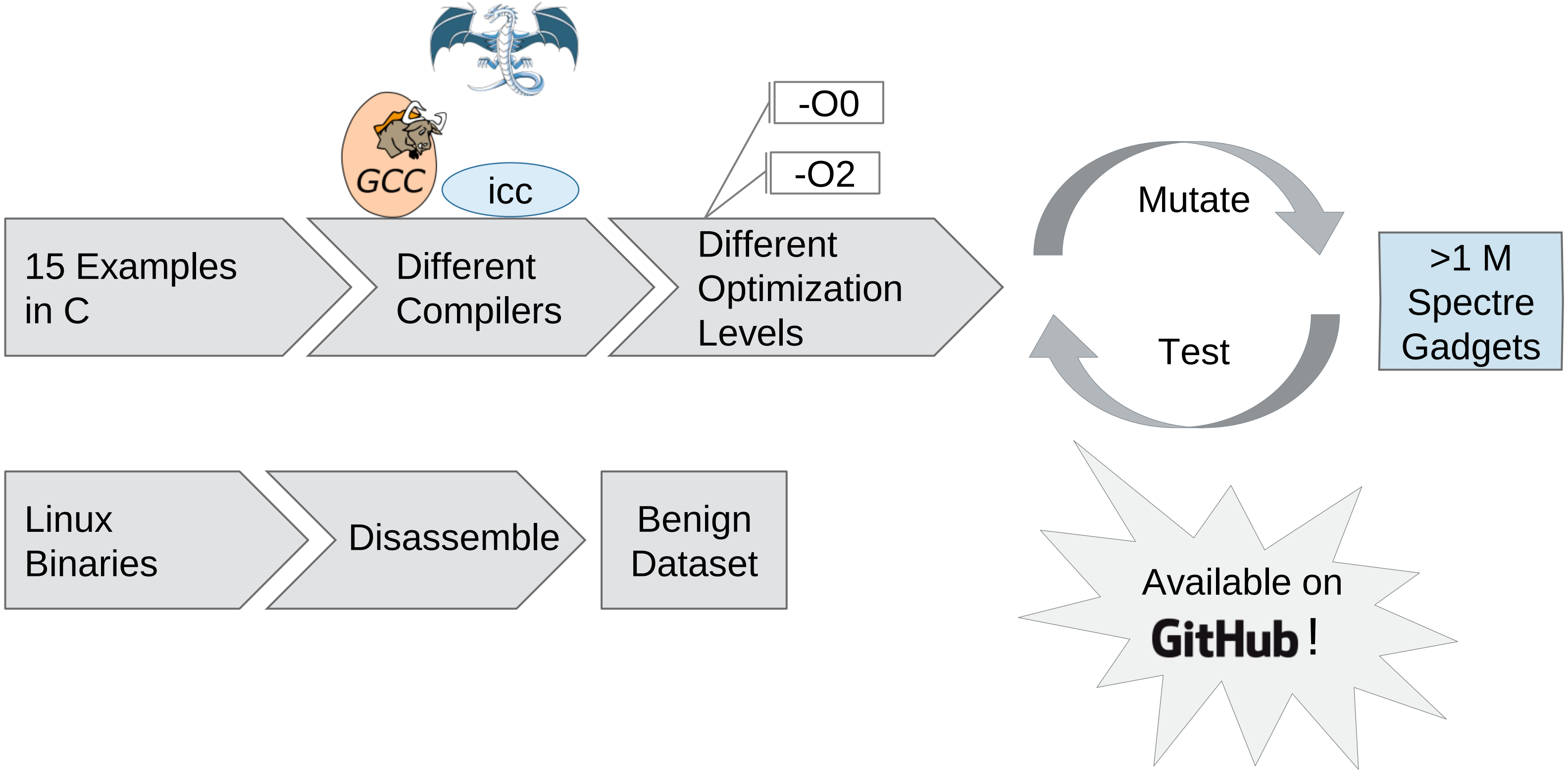
```c
void victim_function_v05(size_t x) {
    size_t i;
    if (x < array1_size) {
        for (i = x - 1; i >= 0; i--)
            temp &= array2[array1[i] * 512];
    }
}
```

```c
void victim_function_v10(size_t x, uint8_t k) {
    if (x < array1_size) {
        if (array1[x] == k)
            temp &= array2[0];
    }
}
```

```c
void victim_function_v09(size_t x, int *x_is_safe) {
    if (*x_is_safe)
        temp &= array2[array1[x] * 512];
}
```

Paul Kocher, *Spectre Mitigations in Microsoft's C/C++ Compiler*, 2018

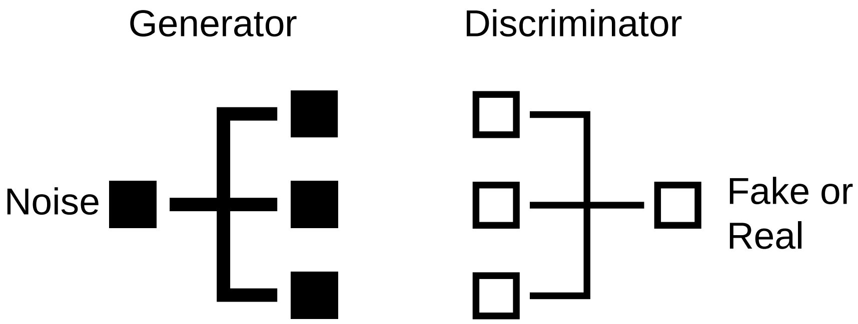
Creating Spectre Gadget Dataset in Assembly



SpectreGAN

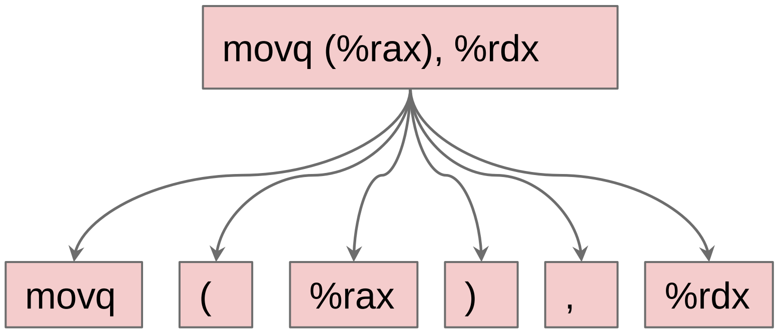
- Spectre gadget generator using Generative Adversarial Networks (Goodfellow et al, NIPS14)

- MaskGAN (Fedus et al, ICLR18)

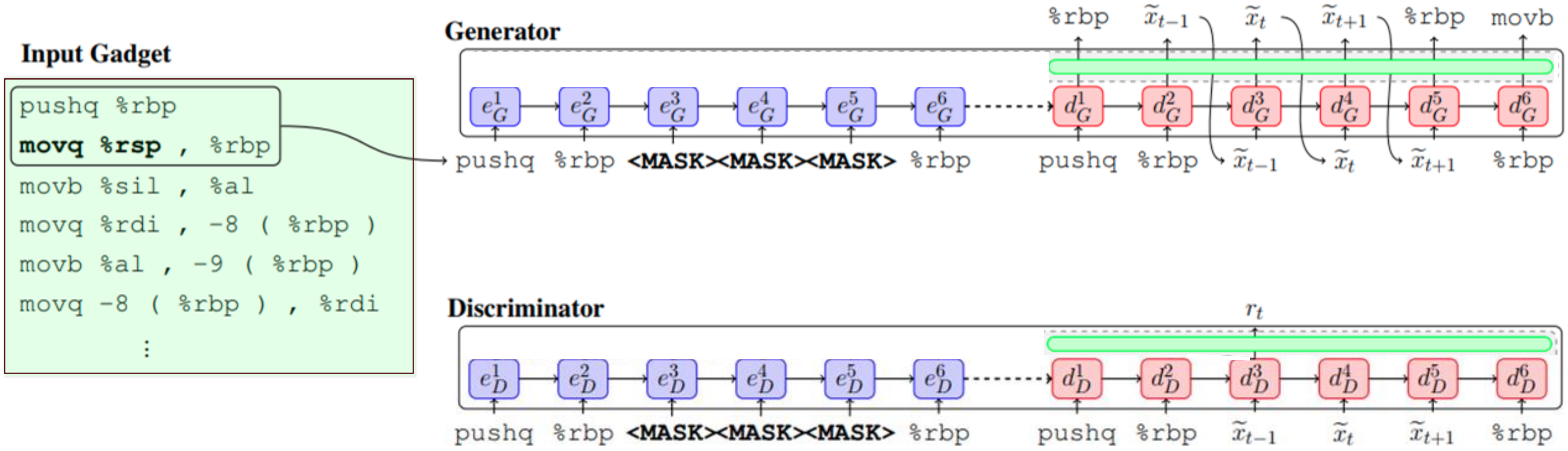


Tokenization

- <imm> Immediate
- <label> Label
- <UNK> Unknown label



SpectreGAN



Input Gadget
pushq %rbp
movq %rsp , %rbp
movb %sil , %al
movq %rdi , -8 (%rbp)
movb %al , -9 (%rbp)
movq -8 (%rbp) , %rdi
⋮
Generator
%rbp $\widetilde{x}_{t-1}$ $\widetilde{x}_t$ $\widetilde{x}_{t+1}$ %rbp movb
e_G^1 e_G^2 e_G^3 e_G^4 e_G^5 e_G^6 d_G^1 d_G^2 d_G^3 d_G^4 d_G^5 d_G^6
pushq %rbp <MASK><MASK><MASK> %rbp pushq %rbp $\widetilde{x}_{t-1}$ $\widetilde{x}_t$ $\widetilde{x}_{t+1}$ %rbp
Discriminator
r_t
e_D^1 e_D^2 e_D^3 e_D^4 e_D^5 e_D^6 d_D^1 d_D^2 d_D^3 d_D^4 d_D^5 d_D^6
pushq %rbp <MASK><MASK><MASK> %rbp pushq %rbp $\widetilde{x}_{t-1}$ $\widetilde{x}_t$ $\widetilde{x}_{t+1}$ %rbp

SpectreGAN

- ~3 days of training
- Assembly function syntax without any supervision
- 70% success rate in the compiled samples compare to 5% in fuzzing.

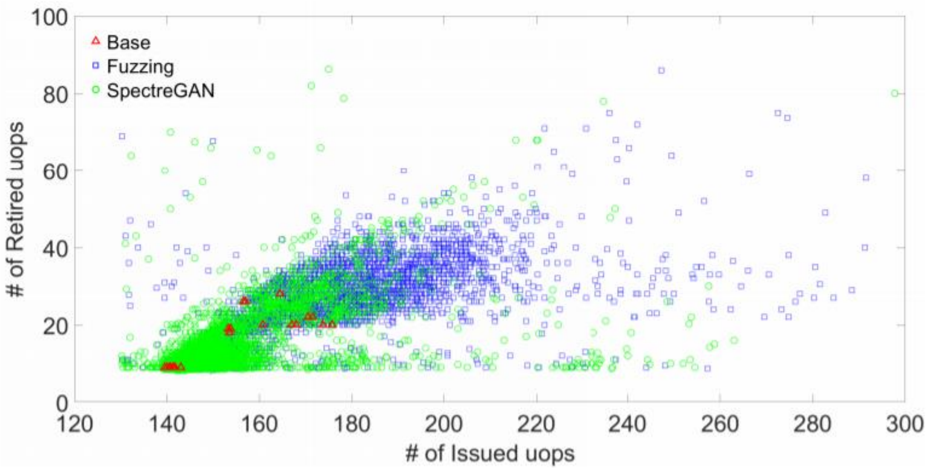
```
victim_function:
.cfi_startproc
movl  size(%rip),%eax
cmpq  %rdi,%rax
jbe  .L0
leaq  array1(%rip),%rax
movzbl  (%rdi,%rax),%eax
ror  $1,%rsi
shlq  $9,%rax
leaq  array2(%rip),%rcx
movss  %xmm8,%xmm4
movb  (%rax,%rcx),%al
andb  %al,temp(%rip)
movd  %xmm1,%r14d
test  %r15,%rcx
sbbl  %r13d,%r9d
.L0:
retq
cmovll  %r8d,%r10d
.cfi_endproc
```

→

```
victim_function:
.cfi_startproc
movl  size(%rip),%eax
cmpq  %rdi,%rax
jbe  .L0
leaq  array1(%rip),%rax
movzbl  (%rdi,%rax),%eax
ror  $1,%rsi
shlq  $9,%rax
movb  array2(%rdi),%al
andb  %al,temp(%rip)
.L1:
andb  %r13b,%al
movb  array2(%rax),%al
andb  %al,temp(%rip)
sbbl  %r13d,%r9d
.L0:
retq
cmovll  %r8d,%r10d
.cfi_endproc
```

SpectreGAN

- Unique n-gram analysis (n=5)
  - Base 4.7K
  - Fuzzing ~1M
  - SpectreGAN ~1M
  - ~2M in total
- Microarchitectural analysis
  - uops_issued vs uops_retired
- Detection analysis
  - oo7 and Spectector tools



Problem 2: Where are they?

- We can blindly put lfence after every branch.
  - With 62-74.8% performance overhead (Carruth, 2018)

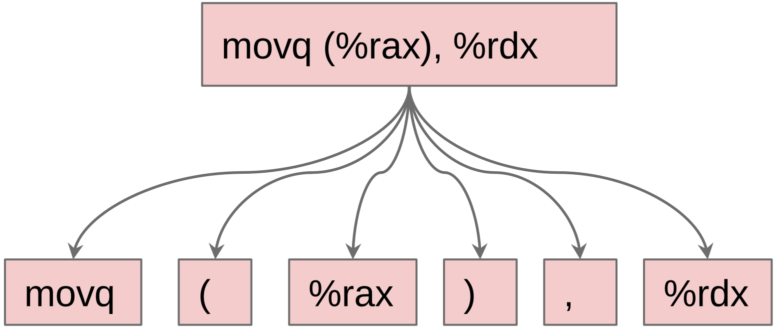
Or…

- We can build a ___________ tool to find the Spectre gadgets.
  - automated
  - scalable
  - accurate

FastSpec

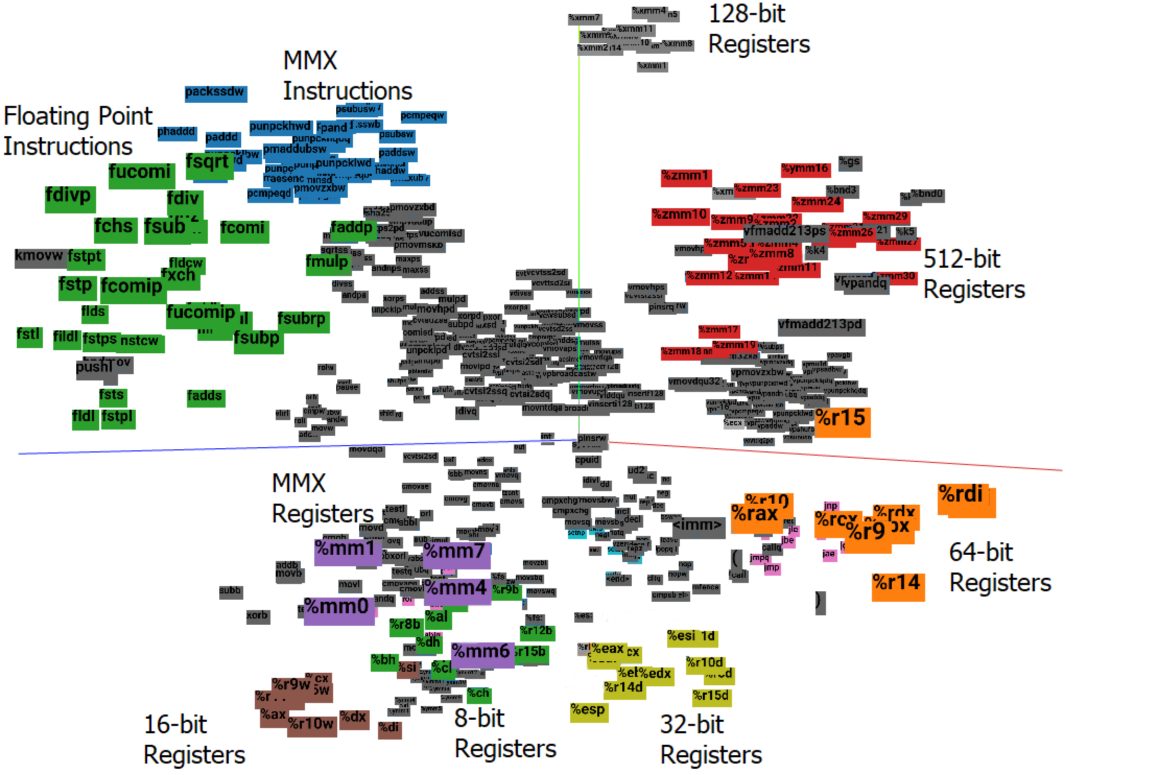
- BERT-based vulnerability detector

  (Devlin et al, NAACL18)
- Scans binaries with linear complexity

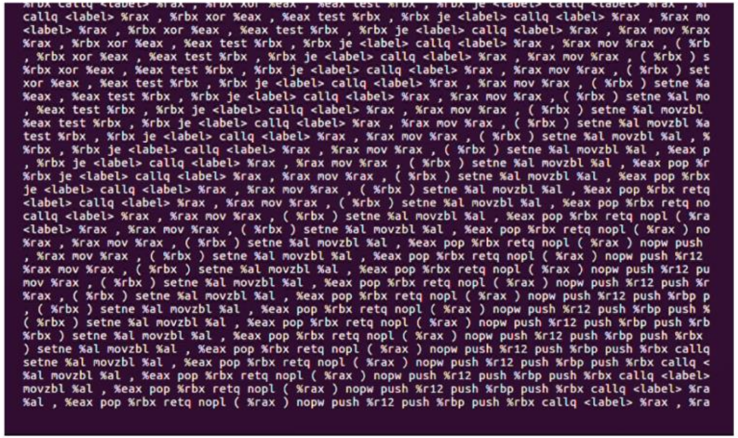


FastSpec

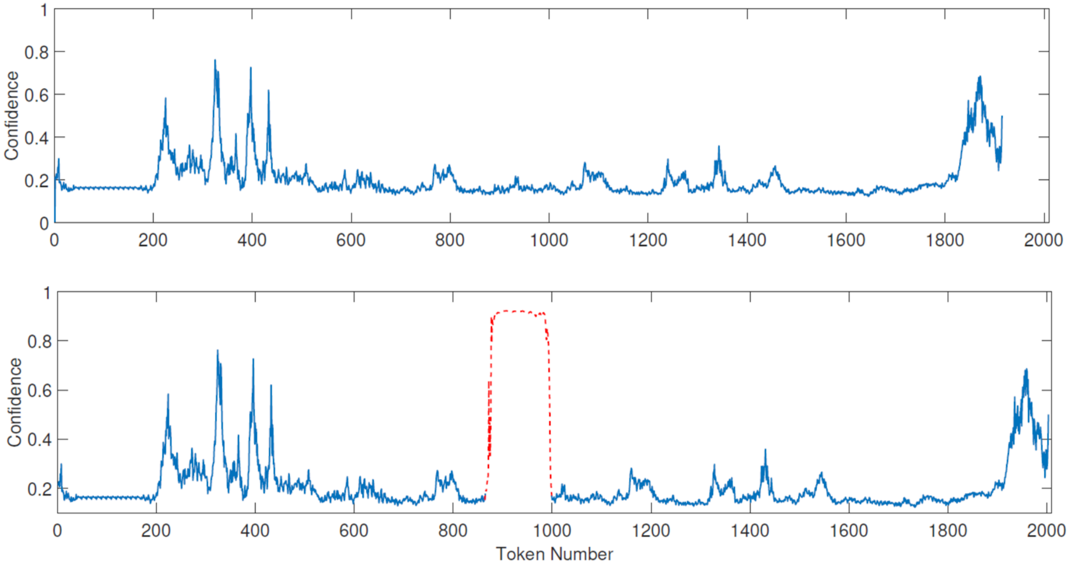
- Vector representations
- t-SNE visualization



FastSpec



callq <label> %rax , %rbx xor %eax , %eax test %rbx , %rbx je <label> callq <label> %rax , %rax mo
<label> %rax , %rbx xor %eax , %eax test %rbx , %rbx je <label> callq <label> %rax , %rax mov %rax
%rax , %rbx xor %eax , %eax test %rbx , %rbx je <label> callq <label> %rax , %rax mov %rax , (%rb
, %rbx xor %eax , %eax test %rbx , %rbx je <label> callq <label> %rax , %rax mov %rax , (%rbx) s
%rbx xor %eax , %eax test %rbx , %rbx je <label> callq <label> %rax , %rax mov %rax , (%rbx) set
xor %eax , %eax test %rbx , %rbx je <label> callq <label> %rax , %rax mov %rax , (%rbx) setne %a
%eax , %eax test %rbx , %rbx je <label> callq <label> %rax , %rax mov %rax , (%rbx) setne %al mo
, %eax test %rbx , %rbx je <label> callq <label> %rax , %rax mov %rax , (%rbx) setne %al movzbl
%eax test %rbx , %rbx je <label> callq <label> %rax , %rax mov %rax , (%rbx) setne %al movzbl %a
test %rbx , %rbx je <label> callq <label> %rax , %rax mov %rax , (%rbx) setne %al movzbl %al , %
%rbx , %rbx je <label> callq <label> %rax , %rax mov %rax , (%rbx) setne %al movzbl %al , %eax p
, %rbx je <label> callq <label> %rax , %rax mov %rax , (%rbx) setne %al movzbl %al , %eax pop %r
%rbx je <label> callq <label> %rax , %rax mov %rax , (%rbx) setne %al movzbl %al , %eax pop %rbx
je <label> callq <label> %rax , %rax mov %rax , (%rbx) setne %al movzbl %al , %eax pop %rbx retq
<label> callq <label> %rax , %rax mov %rax , (%rbx) setne %al movzbl %al , %eax pop %rbx retq no
callq <label> %rax , %rax mov %rax , (%rbx) setne %al movzbl %al , %eax pop %rbx retq nopl (%ra
<label> %rax , %rax mov %rax , (%rbx) setne %al movzbl %al , %eax pop %rbx retq nopl (%rax) no
%rax , %rax mov %rax , (%rbx) setne %al movzbl %al , %eax pop %rbx retq nopl (%rax) nopw push
, %rax mov %rax , (%rbx) setne %al movzbl %al , %eax pop %rbx retq nopl (%rax) nopw push %r12
%rax mov %rax , (%rbx) setne %al movzbl %al , %eax pop %rbx retq nopl (%rax) nopw push %r12 pu
mov %rax , (%rbx) setne %al movzbl %al , %eax pop %rbx retq nopl (%rax) nopw push %r12 push %r
%rax , (%rbx) setne %al movzbl %al , %eax pop %rbx retq nopl (%rax) nopw push %r12 push %rbp p
, (%rbx) setne %al movzbl %al , %eax pop %rbx retq nopl (%rax) nopw push %r12 push %rbp push %
(%rbx) setne %al movzbl %al , %eax pop %rbx retq nopl (%rax) nopw push %r12 push %rbp push %rb
%rbx) setne %al movzbl %al , %eax pop %rbx retq nopl (%rax) nopw push %r12 push %rbp push %rbx
) setne %al movzbl %al , %eax pop %rbx retq nopl (%rax) nopw push %r12 push %rbp push %rbx callq
setne %al movzbl %al , %eax pop %rbx retq nopl (%rax) nopw push %r12 push %rbp push %rbx callq <
%al movzbl %al , %eax pop %rbx retq nopl (%rax) nopw push %r12 push %rbp push %rbx callq <label>
movzbl %al , %eax pop %rbx retq nopl (%rax) nopw push %r12 push %rbp push %rbx callq <label> %ra
%al , %eax pop %rbx retq nopl (%rax) nopw push %r12 push %rbp push %rbx callq <label> %rax , %ra



Confidence
0
0.2
0.4
0.6
0.8
1
0
200
400
600
800
1000
1200
1400
1600
1800
2000
Token Number

Case Study 1: OpenSSL

- OpenSSL v3 "*speed*" benchmark
- SpecFuzz (Oleksenko et al, USENIX '20)

- Sliding window of size 80 tokens
- AUC=0.998
- FP=0.04%, FN=2%

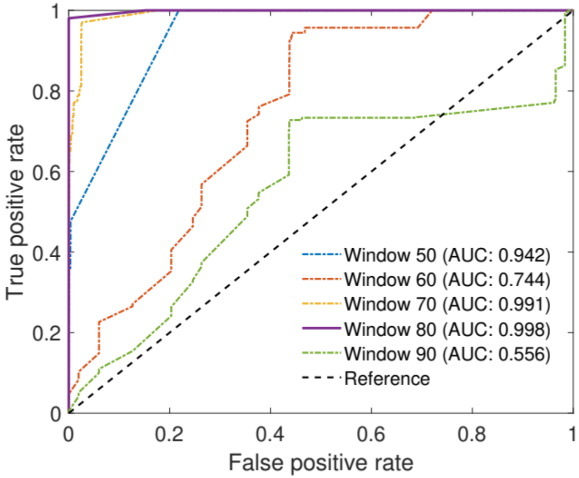
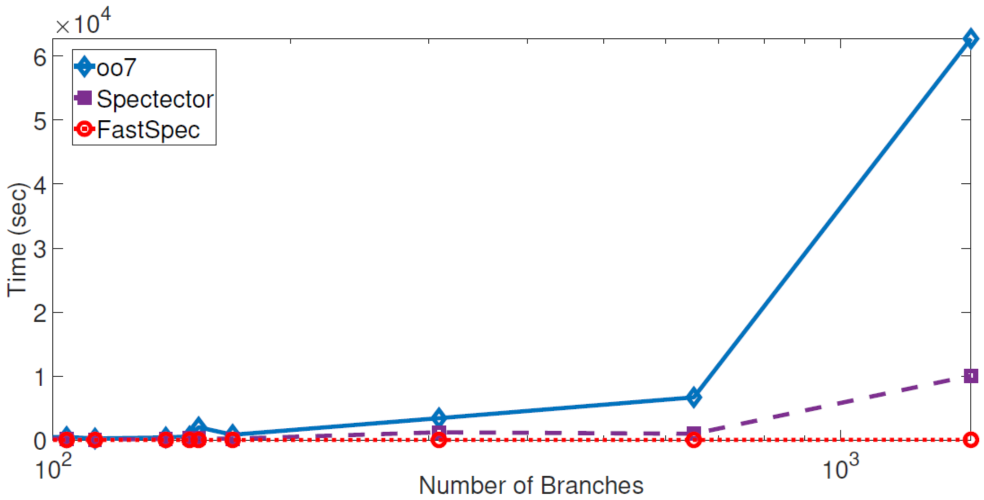


Figure 5: Solid line stands for the ROC curve of FastSpec for Spectre gadget class. Dashed line represents the reference line.

Case Study 2: Phoronix Test Suite

- State of the art tools are not scalable.
- Crafty benchmark
  - 10K branches
  - 0.6 MB
- Spectector: 2 days
- oo7: 10+ days
- FastSpec: **<6 mins**



Case Study 2: Phoronix Test Suite

TABLE 2: Comparison of *oo7* [6], Spectector [8], and FastSpec on the Phoronix Test Suite. The last column shows that FastSpec is on average 455 times faster than *oo7* and 75 times faster than *Spectector*. (#CB: Number of conditional branches, #Fc: Number of functions, #DFc: Number of detected functions)

Benchmark	Size (KB)	#CB	#Fc	SpecFuzz	oo7			Spectector			FastSpec		
				#DFc	Precision	Recall	Time (sec)	Precision	Recall	Time (sec)	Precision	Recall	Time (sec)
Byte	183.5	363	83	7	0.70	**0.90**	400	1.00	0.43	115	**1.00**	0.86	**14**
Clomp	79.4	1464	45	1	0	0	17.5 hr	0.05	0.9	2.8 hr	**1.00**	**1.00**	**35**
Crafty	594.8	10796	207	44	**1.00**	0.54	>10 day	0.60	**0.91**	48 hr	0.23	0.80	**315**
C-ray	27.2	139	11	1	**1.00**	1.00	395	0.2	0.9	153	0.50	**1.00**	**8**
Ebizzy	18.5	104	6	3	0	0	467	0.60	1.00	206	**1.00**	0.33	**3**
Mbw	13.2	70	5	1	0	0	145	**0.50**	1.00	34	0.33	**1.00**	**2**
M-queens	13.4	51	4	1	1.00	1.00	136	0.50	1.00	24	**1.00**	**1.00**	**2**
Postmark	38.0	309	49	6	1.00	0.83	3409	0.43	0.95	1202	**1.00**	**1.00**	**10**
Stream	22.0	113	4	3	0	0	231	0	0	63	**1.00**	**0.66**	**4**
Tiobench	36.1	169	19	1	0	0	813	0.25	0.8	201	**0.33**	**1.00**	**9**
Tscp	40.8	651	38	13	0	0	6667	1.00	0.15	972	**1.00**	**0.92**	**12**
Xsbench	27.9	153	32	1	**1.00**	**1.00**	1985	0	0	249	0.50	0.90	**7**
Average					0.47	0.44		0.43	0.67		**0.74**	**0.87**	

Conclusion

- New Spectre gadget dataset with 1+ million samples
- Task specific assembly code generation
- New DL-based Spectre v1 detection tool

Reproduce results: **vernamlab/FastSpec**

Contact me: **mtol@wpi.edu**

canertol