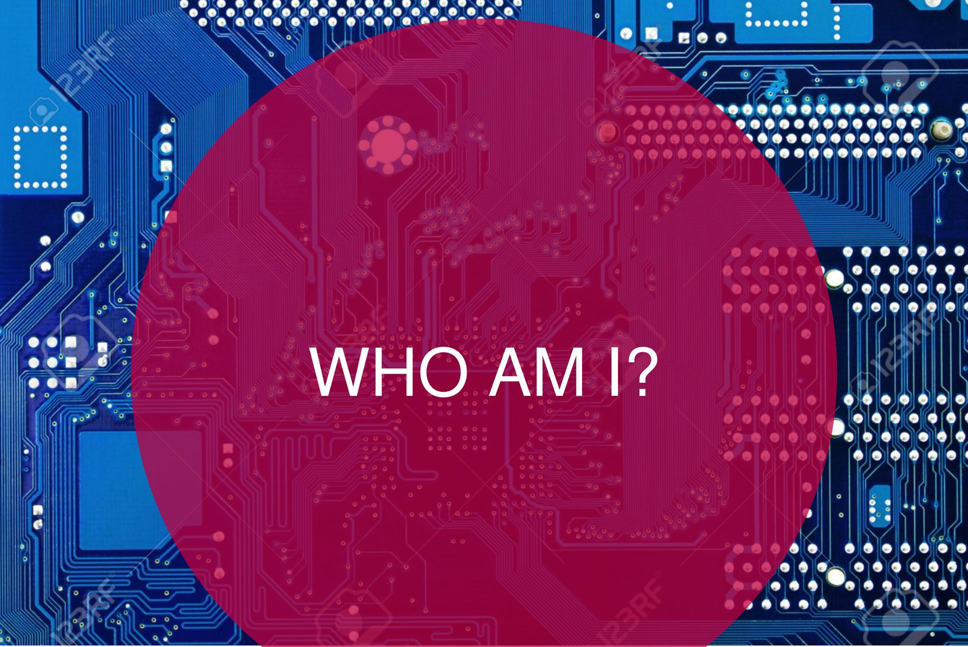
SMT Solvers in IT Security - Deobfuscating binary code with logic

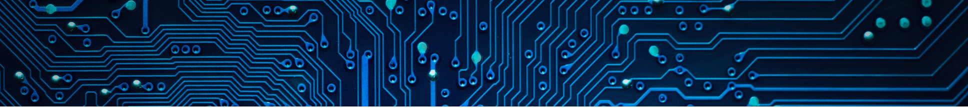
barbieauglend @ BlackHoodie17 - Luxembourg
✉ barbieauglend@chaosdorf.de • 🐦 barbieauglend

DISCLAIMER

This research was accomplished by me in my personal capacity. The opinions and views expressed in this talk and article are my own and do not necessarily reflect the official policy or view of my employer.

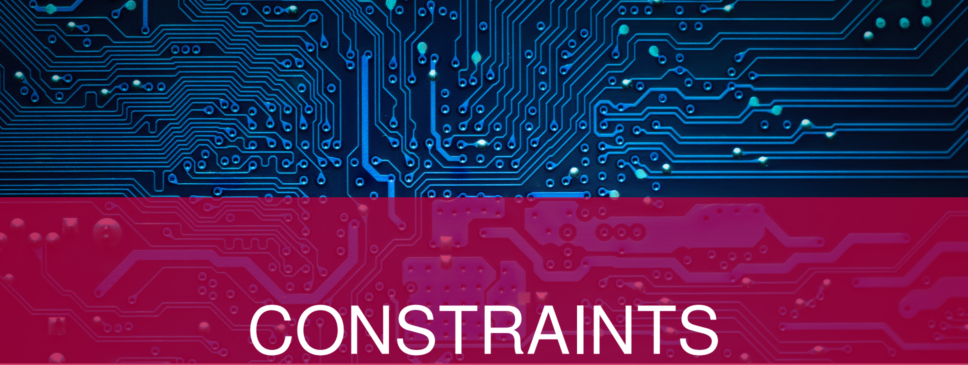


WHO AM I?

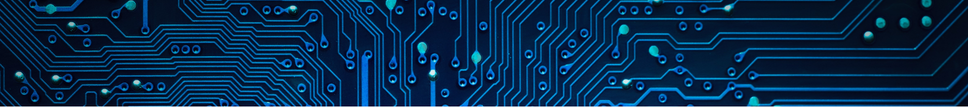


Overview:

- Introduction to Constraint Logic Programming
- Applications of CLP in IT Security
- Binary Obfuscation
- Malware deobfuscation using CLP

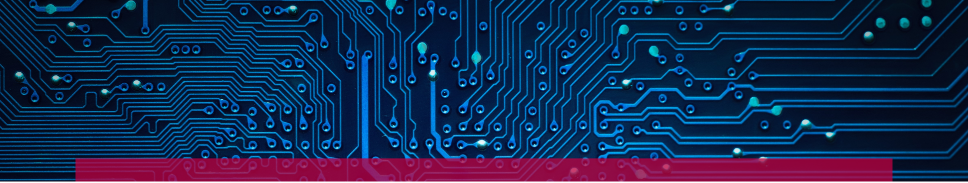


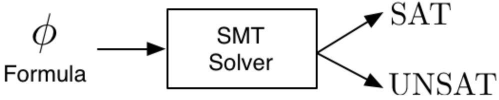
CONSTRAINTS



> ”Constraint programming represents one of the closest approaches computer science has yet made to the Holy Grail of programming: the user states the problem, the computer solves it.”

Eugene C. Freuder, Constraints, April 1997





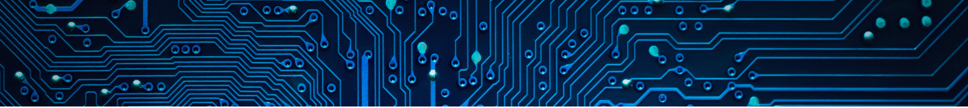
ϕ

Formula

SMT Solver

SAT

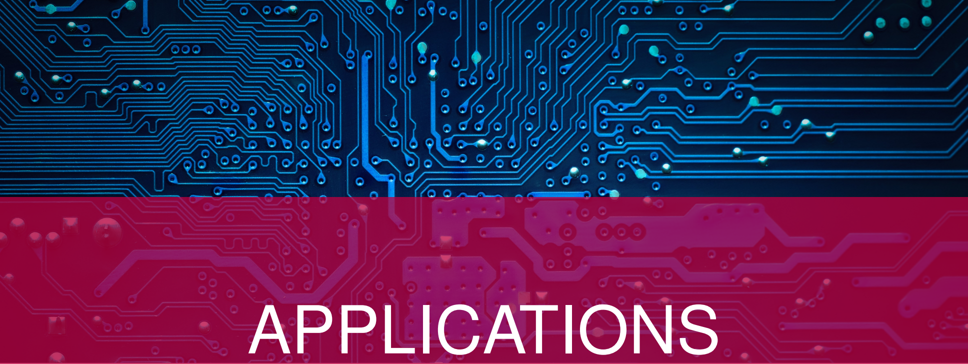
UNSAT



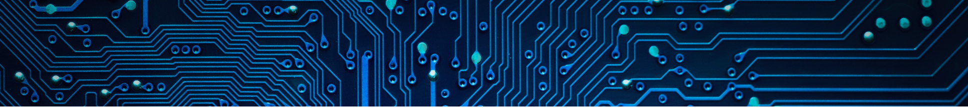
Automated Theorem Proving

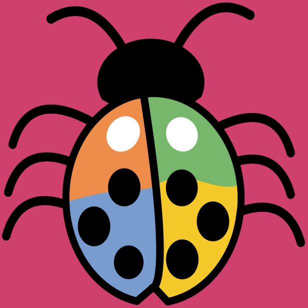
- Hardware and Software → Large-scale verification
- Languages specification and Computing proof obligations

SYMBOLIC EXECUTION



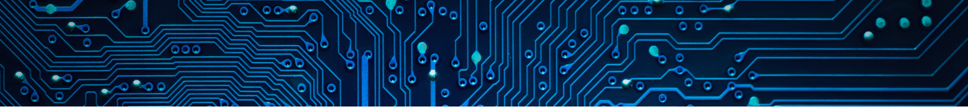
APPLICATIONS





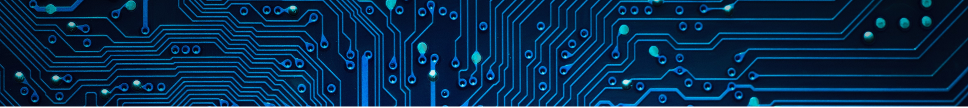
Bug Hunting

- Fuzzing
- Verification
- Analysis



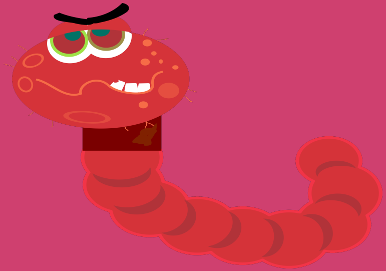
Exploit Generation

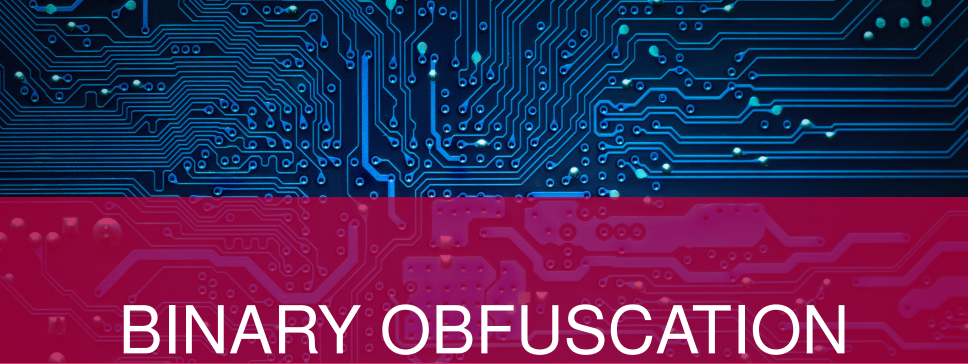
- Automatic Exploit Generation
- Proof of Concept
- Automatic Payload Generation



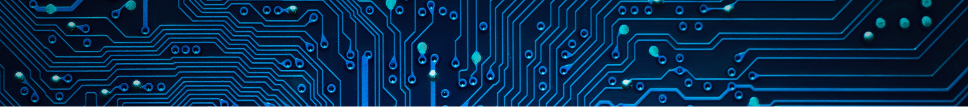
Malware Analysis

- Obfuscation
- Garbage-code elimination
- Compilation
- Packing
- Anti-debugging
- Crypto analysis



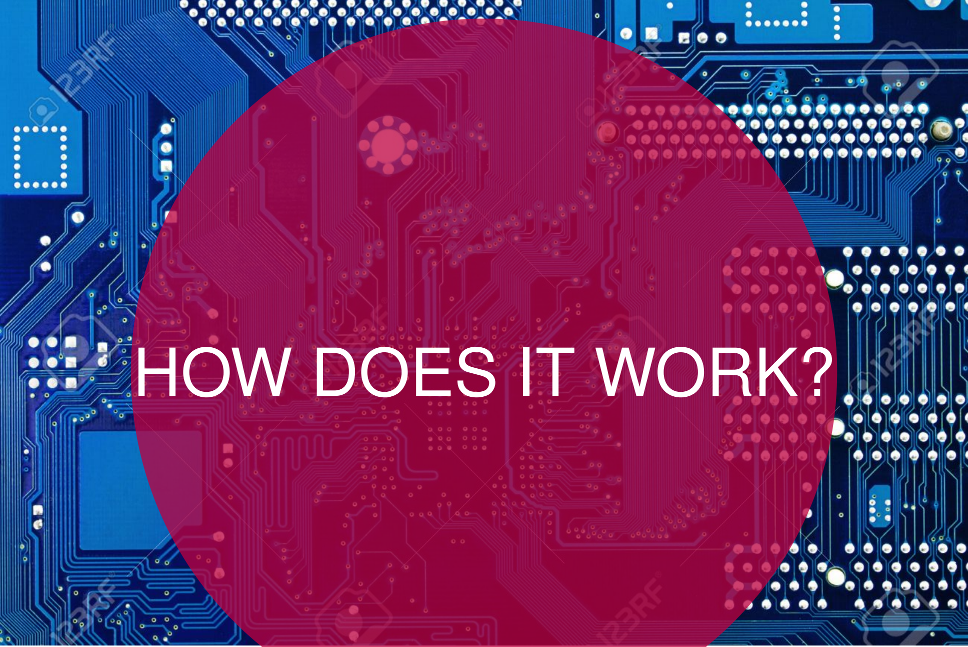


BINARY OBFUSCATION

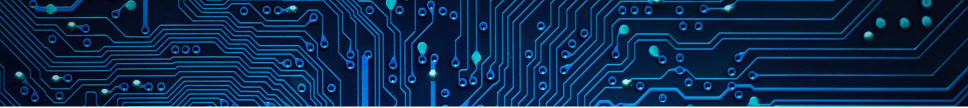


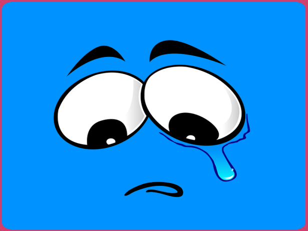
Malware Obfuscation

SW Property Protection

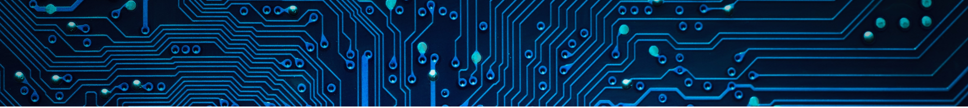


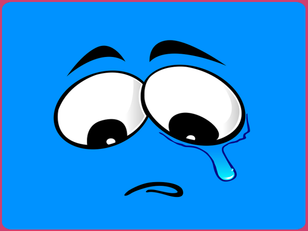
HOW DOES IT WORK?





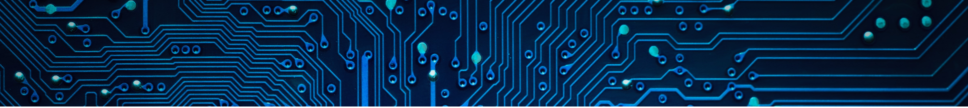
- Compiled
- Packed
- Obfuscated
- Anti-debugging

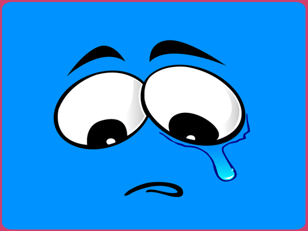




Garbage Code

- Unnecessary instructions
- Jumps that are never taken

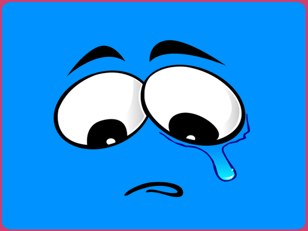


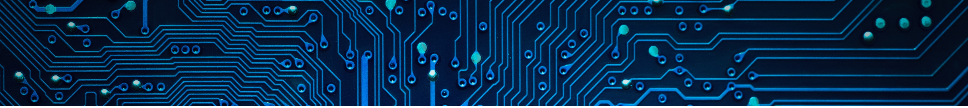


The exclusive or operation

Packers

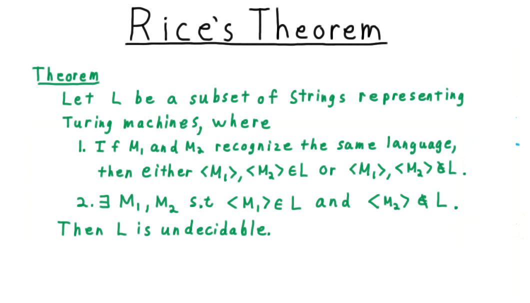
- UPX, NSIS
- self implemented





Malware Analysis

- Practical: Techniques to thwart analysis
- Theoretical: Rice's Theorem



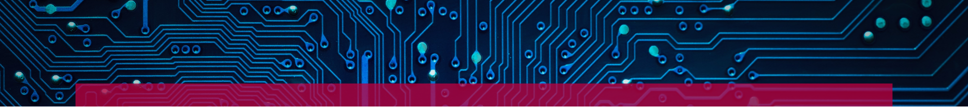
Rice's Theorem

Theorem

Let L be a subset of Strings representing Turing machines, where

1. If M_1 and M_2 recognize the same language, then either $\langle M_1 \rangle, \langle M_2 \rangle \in L$ or $\langle M_1 \rangle, \langle M_2 \rangle \notin L$.
2. $\exists\, M_1, M_2$ s.t $\langle M_1 \rangle \in L$ and $\langle M_2 \rangle \notin L$.

Then L is undecidable.



```
x = input();
x = x + 7;

if (x > 0)
y = input();
else
y = 11;

if (x > 2)
if (y == 42)
throw
  Exception()
```

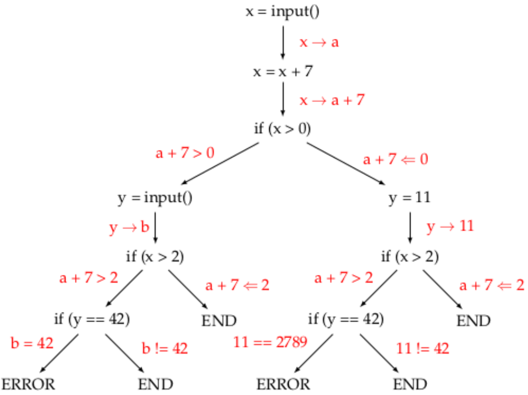
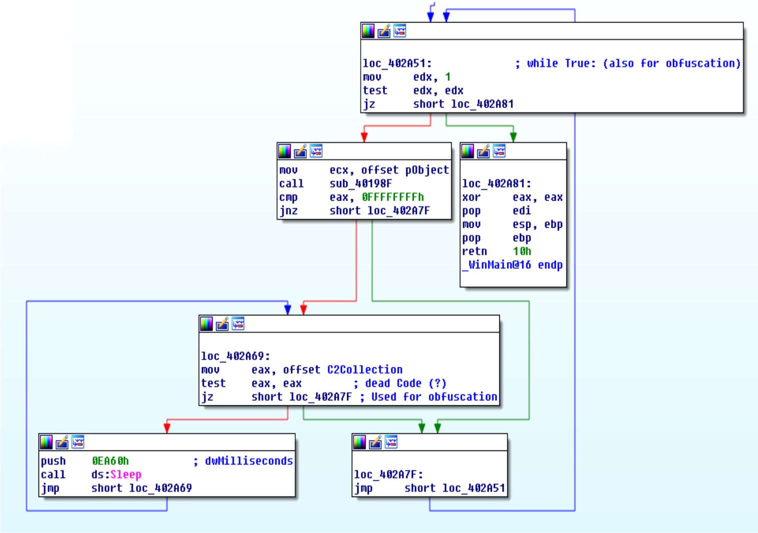


Figure 8: *Example of symbolic execution for simple program*

- Symbols as arguments

  ⇒ **any** feasible path
- Program states
  - Symbolic values for memory locations
  - Path conditions



```
loc_402A51:              ; while True: (also for obfuscation)
mov     edx, 1
test    edx, edx
jz      short loc_402A81
```

```
mov     ecx, offset pObject
call    sub_40198F
cmp     eax, 0FFFFFFFFh
jnz     short loc_402A7F
```

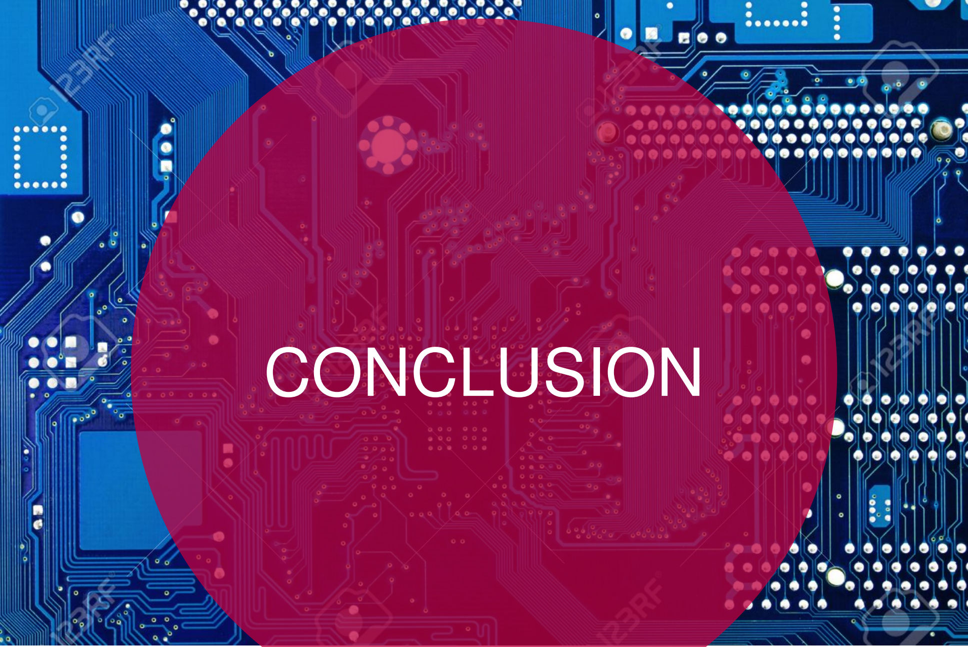
```
loc_402A81:
xor     eax, eax
pop     edi
mov     esp, ebp
pop     ebp
retn    10h
_WinMain@16 endp
```

```
loc_402A69:
mov     eax, offset C2Collection
test    eax, eax         ; dead Code (?)
jz      short loc_402A7F ; Used for obfuscation
```

```
push    0EA60h            ; dwMilliseconds
call    ds:Sleep
jmp     short loc_402A69
```

```
loc_402A7F:
jmp     short loc_402A51
```

CONCLUSION



THANK YOU!