Evaluating Defensive Countermeasures for Software-Based Hardware Abstraction

James Carambat, J. Todd McDonald, Todd R. Andel, Samuel H. Russ

School of Computing, University of South Alabama

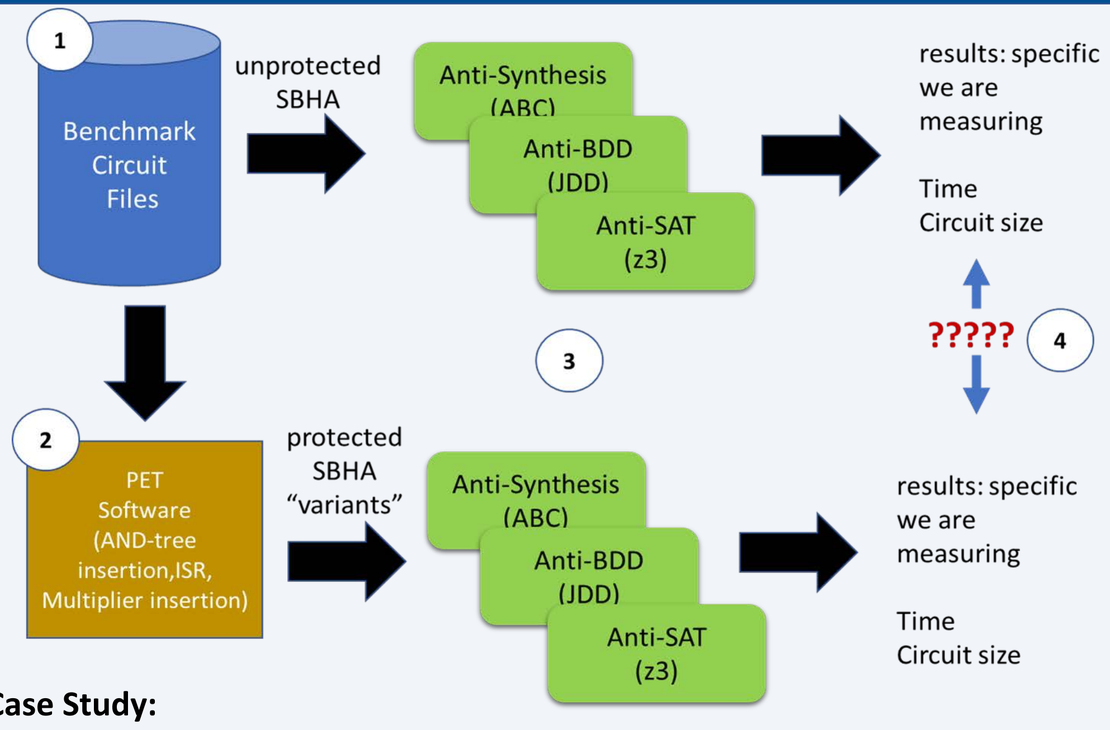
ABSTRACT

Software protection is one of the most important methods of preventing adversaries from obtaining a software owner's **intellectual property (IP)** within sold and distributed software programs. Statistics from 2021 indicate that software piracy increased 20% 30% during the pandemic and that 37% of software on tested machines worldwide was unlicensed. It was also estimated that 21% of all internet consumers download and use illegal games. Unlicensed software, in this case, represents cracked software that has been embezzled from the original owner and resold by an adversary. These types of attacks are labeled as **man at the end (MATE)** attacks, where an adversary is a legitimate end user who has full control and authority over a target software environment, giving free range to reverse engineering, tampering, and cloning of the program. A novel method of program protection called **software-based hardware abstraction (SHBA)** specifically targets adversarial reverse engineering, in the attempt to circumvent or completely defeat automated tools that are used by MATE attacks in finding **secret information hidden in programs**. Since many programs rely on checking secret information in the form of PIN codes, passwords, or keys for authentication, SBHA is focused initially on **protection of point-functions**, which are essentially IF statements that return true for only one input out of all possible others. SBHA takes such software constructs and transforms them into a **digital logic representation.** The circuit is then transformed back into software, which results in a protected variant of the original program. The feature of concentration in this study will be on **modifying the circuits in a way that an adversary could not easily return the value of the program point function**, given a **recovered netlist of the circuit from a software program**. These methods of circuit modification and protection, once implemented, will be analyzed by hardware specific tools, specifically **ABC synthesis tool**, **Java Binary Decision Diagrams (JDD)**, and the **z3 SAT solver.** The goal of the research is to characterize the strength of countermeasures against such state-of-the-art tools that essentially act in the role of an attacker.

RESEARCH QUESTIONS

- ❑ **What is the power of an adversary that recovers a (circuit) netlist from an SBHA-protected binary?**
- ❑ **Can we produce effective countermeasures to defeat MATE attacks that target netlist analysis from SBHA programs?**

METHODOLOGY



Case Study:

- ❑ **Step 1:** Create **benchmark circuits** which are the basis of password-query point functions (1,8,12,16,20 character passwords): these represent **unprotected SBHA circuit netlists**
- ❑ **Step 2:** Use **semantic-preserving circuit transformations** on the benchmark circuit set that are potentially able to deter or hinder SOTA circuit analysis tools and techniques
- ❑ **Step 3:** Evaluate both **unprotected** and **protected** versions of benchmark circuits against synthesis, BDD, and SAT-based analysis tools
- ❑ **Step 4:** Compare **effectiveness** of transformations against tools

WHAT IS SBHA?

- ❑ **Software-based hardware abstraction (SBHA)** replaces code/software-based structures with a virtual digital logic circuit, which is represented as code (software)
- ❑ Take a **program P** which has a point-function for checking a password (%), which is an embedded secret

P

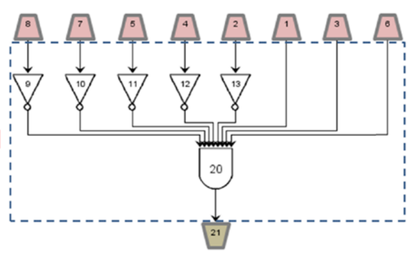
```
if (password == "%")
  printf ("1"); runprogram();
else
  printf ("0"); exit(-1);
```

- ❑ The function of program P can be represented as a **Boolean logic circuit C** that will output a 1 (true) only if the input matches the required password

C

87654321	20
00000000	0
00000001	0
00000010	0
00000011	0
00000100	0
...	
00100100	0
00100101	**1**
00100110	0
...	
11111100	0
11111101	0
11111110	0
11111111	0

ASCII '%'
= 0x25
= 00100101



20 = !8 * !7 * 6 * !5 * !4 * 3 * !2 * 1

- ❑ The **digital logic circuit C** can be converted back into traditional software, thus virtualizing the original program P into a **circuit-based version (P_{HW})**

P_{HW}

```
void runcircuit(bool input[], bool out[]) {
   bool v8 = input[0], v7 = input[1],
   ... v2 = input[6], v1 = input[7];
   bool v9 = not(v8), v10 = not(v7),
   v11 = not(v5), v12 = not(v4), v13 = not(v2);
   bool v20in[8];
   v20in[0] = v9; v20in[1] = v10;
   v20in[2] = v6; v20in[3] = v11;
   v20in[4] = v12; v20in[5] = v3;
   v20in[6] = v13; v20in[7] = v1;
   v20  = andmulti(v20in,8);
   out[0] = v20;
}
```

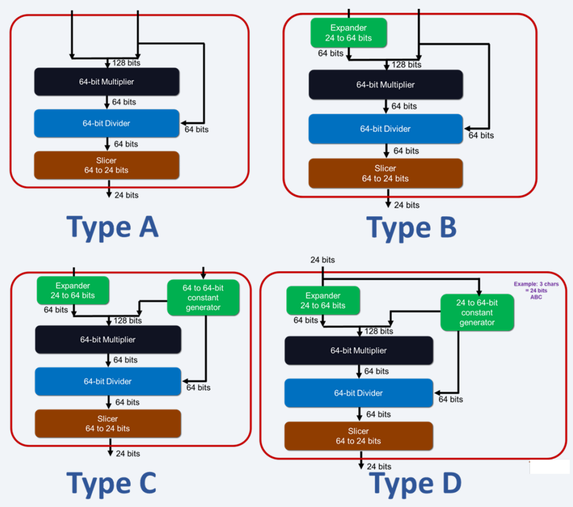
- ❑ The software-based abstraction can then be used to replace the original point function check and create a **new version of program P (referred to as P')** – thus the IF statement condition is replaced with a Boolean logic abstraction (SBHA)

P'

```
bool circuitinput[8];
bool circuitoutput[1];
convertString(argv[1],circuitinput,1);
runcircuit (circuitinput, circuitoutput);
if (circuitoutput[0]) {
  printf ("0"); exit(-1); }
else {
  printf ("1"); runprogram(); }
```

INITIAL RESULTS

- ❑ Four different types of multiplier/divider configurations (Type A, B, C, D) are considered for insertion BEFORE the point function circuit as an anti-SAT and anti-BDD approach



- ❑ Initial results of 1-char password circuit against 3 different forms of attackers (SAT, BDD, synthesis) show Type A/B completely successful

Type	Component	Size	JDD (Y/N)	JDD Time	z3	z3 Time
A	pwd1-A	36489	N	N/A	incorrect model	1m:17s
A	pwd1-A.isr	34485	N	N/A	no model	1m:19s
A	pwd1-A.isrfull	36447	N	N/A	no model	45s
B	pwd1-B	36553	N	N/A	incorrect model	53s
B	pwd1-B.isr	35722	N	N/A	no model	50s
B	pwd1-B.isrfull	37763	N	N/A	no model	45s
C	pwd1-C	36617	Y	1m:48s	recovered	52s
C	pwd1-C.isr	36930	Y	1m:45s	no model	55s
C	pwd1-C.isrfull	39053	Y	2m:51s	no model	1m:2s
D	pwd1-D	36617	Y	1m:47s	recovered	50s
D	pwd1-D.isr	36793	Y	2m:12s	no model	1m:0s
D	pwd1-D.isrfull	38676	Y	2m:16s	no model	55s

ACKNOWLDGEMENTS

This research was supported in part by the National Science Foundation under grant DGE-1564518.

QR Code