SoftMark: Software Watermarking via a Binary Function Relocation

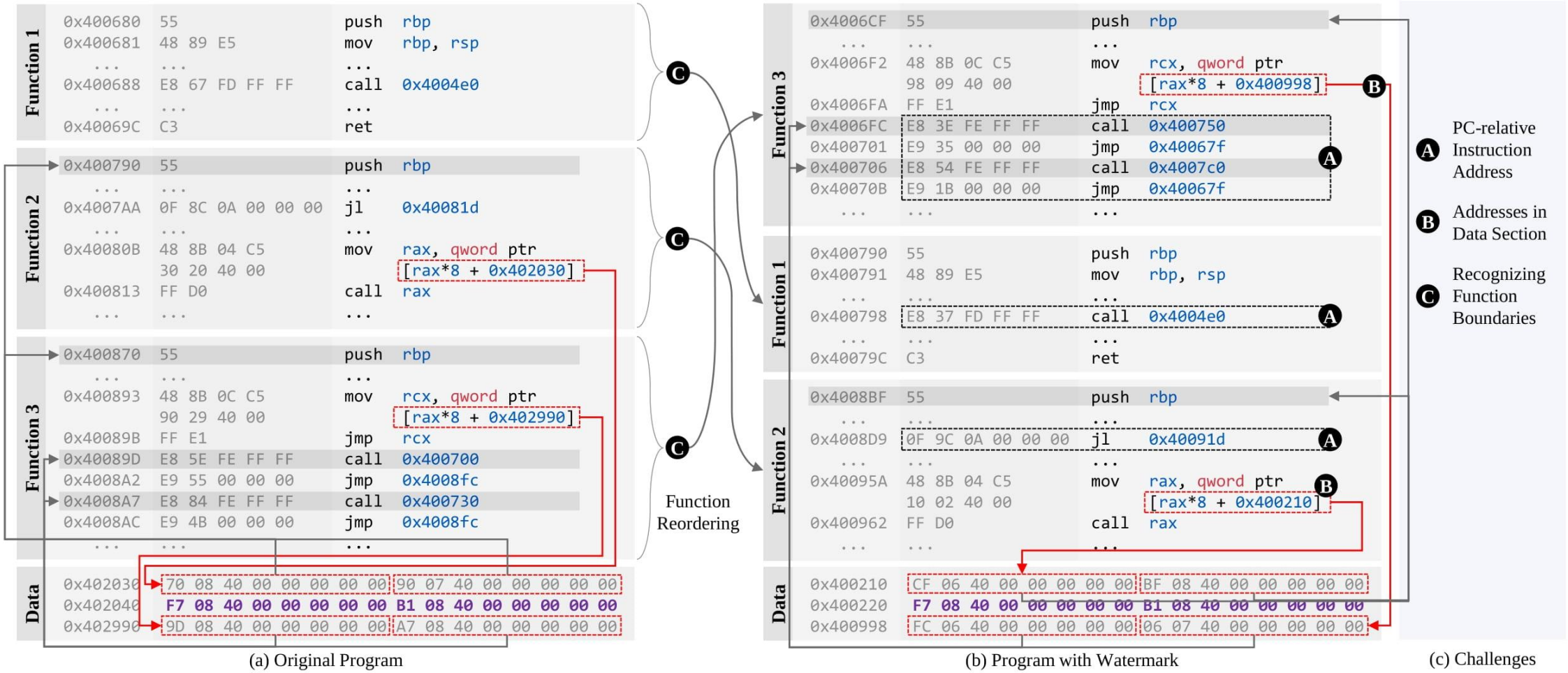
ACSAC 2021

Honggoo Kang, Yonghwi Kwon,
Sangjin Lee, and Hyungjoon Koo

Software Watermarking

- Providing a digital fingerprint
  - Inserts certain information that represents its owner or distributor
  - Offers the traceability
- Difference with code-signing?
  - Code-signing focuses on preventing unauthorized modification
  - Better to use *both* code-signing and software watermarking
- Impossible to prevent all viable attacks, but a watermarking scheme should sufficiently discourage attackers

Motivation

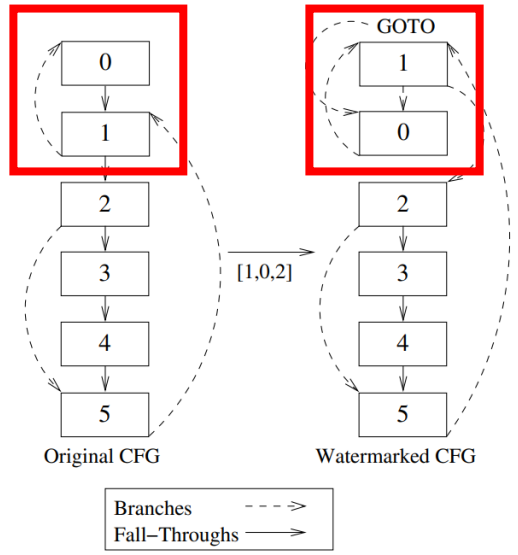


(a) Original Program (b) Program with Watermark (c) Challenges

Prior Attempts

- Reordering-based Approach
  - Reordering basic blocks, safe operands of mathematical equations

- Limitations
  - Perceptive
    - Inserting GOTO statements ➔ easily detectable
  - Forgeable
    - Rearranging a structure can be accomplished easily
  - Fragile
    - Not resilient to arbitrary modifications at the instruction level



[Figure Ref]
- Myles, G., Collberg, C., Heidepriem, Z., and Navabi, A. 2005. The Evaluation of Two Software Watermarking Algorithms. Software Practice and Experience

Advantages over Prior Attempts

- Introducing no supplementary structure
  - No size overhead, negligible performance overhead
  - Gives a lower chance for attackers to recognize the presence of a watermark with statistical analysis or inference

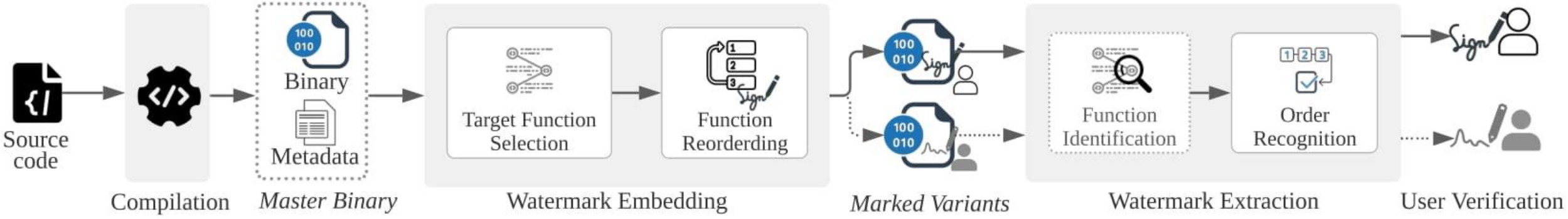
- Can reach up to an increasingly large number of encodings
  - Depends on the number of reorderable functions

Overview

- SoftMark
  - A software watermarking system that leverages a function relocation
  - The order of functions implicitly encodes a hidden identifier

- Process
  - Generating a master binary ➔ Embedding a watermark & Recording a ledger

  ➔ Extracting a watermark from variant with a ledger ➔ Verifying an associated user



Reliable Binary Instrumentation

- CCR(Compiler-assisted Code Randomization) (Koo, Hyungjoon et al., In 2018 IEEE S&P)
  - A reliable binary rewriter that leverages the metadata to generate variants
  - A modified compiler toolchain(LLVM and gold linker) produces transformation-assisting metadata including function boundaries, fixup locations, and jump table information
  - A set of metadata can be embedded into executables

➔ Leverage CCR for reliable watermark embedding & extracting at the binary level

Strategies on SoftMark

- One to one mapping (order, value)

Function Order	$F_1-F_3-F_2$	$F_2-F_1-F_3$	$F_2-F_3-F_1$	$F_3-F_1-F_2$
Watermark Value	00_2	01_2	10_2	11_2

- Stirling's formula:

$$\log_2(n!) \approx \log_2(\sqrt{2\pi n}(\frac{n}{e})^n) \text{ where } n > 0$$

Functions	Representable bits	Bits	Required functions
10	21	64	21
20	61	128	35
30	107	192	47
50	214	256	58
100	524	512	99
300	2041	1024	171
500	3767	2048	301
1000	8529	4096	537

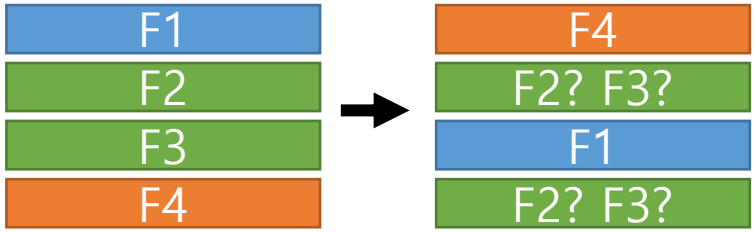
- Higher capacity

Program Name	Size (KB)	SoftMark		Davidson-Myhrvold	
		Functions	Bits	Basic Blocks	Bits
400.perlbench	1,423	895	**7,491**	683	**5,451**
403.gcc	3,728	2,206	**21,326**	534	**4,073**
433.milc	148	88	**446**	20	**61**
445.gobmk	3,923	857	**7,119**	135	**765**
456.hmmer	339	237	**1,532**	46	**191**
458.sjeng	156	78	**382**	74	**357**
464.h264ref	685	259	**1,708**	160	**945**
482.sphinx3	210	155	**909**	20	**61**
addr2line	1,180	704	**5,649**	142	**815**
ar	1,213	739	**5,982**	142	**815**
bfdtest1	1,165	686	**5,479**	142	**815**
cxxfilt	1,179	705	**5,659**	142	**815**
nm-new	1,195	721	**5,810**	142	**815**
objcopy	1,410	867	**7,217**	181	**1,101**
objdump	2,474	988	**8,409**	186	**1,139**
ranlib	1,213	739	**5,982**	142	**815**
size	1,180	708	**5,687**	142	**815**
strings	1,180	705	**5,659**	142	**815**
strip-new	1,410	867	**7,217**	181	**1,101**
ctags	1,495	1,150	**10,039**	178	**1,078**
lighttpd	195	136	**772**	85	**426**
vsftpd	118	143	**822**	87	**439**
pscp	713	664	**5,273**	99	**518**
psftp	722	671	**5,338**	99	**518**
puttygen	391	328	**2,273**	144	**829**
cgtest	405	335	**2,332**	141	**808**

Strategies on SoftMark (2)

- Unique function candidates
  - Do not use indistinguishable functions in watermark embedding

(To prevent ambiguous watermarks at extracting)



- Desirable function set
  - Prefer a function that has a trampoline containing code pointers

(To give more challenge to an attacker)

- Non-candidate functions
  - Randomly scatter all other functions

(To increase a resiliency against collusive attacks)

Strategies on SoftMark (3)

- Ledger (Bookkeeper)
  - Holds the list of functions and their locations, the property of each function (index, indirect call invocation), and the pattern of basic blocks
  - Basic block recognition with patterns:
    - Regular expression of byte values (Quick search)

      Index, Basic block patterns
      ex) FUNC#1 ['5bc30f1f4000', '5389fbb8[0-9a-f]{8}b9[0-9a-f]{8}']

      Fixups

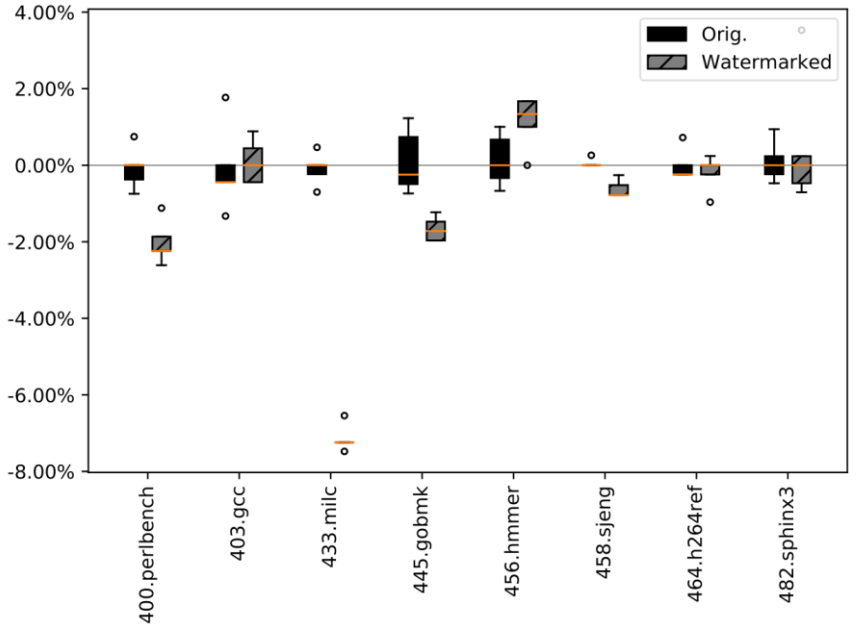
    - Sequence of opcode mnemonics and sizes after disassembly (Deep investigation)

      Index, Basic block patterns
      ex) FUNC#1 ['(push, 1)(push, 1)(push, 1)(sub, 7)(jnz, 6)', ...]

Evaluation

- Size overheads = None
  - No additional structure

- Performance overheads
  - Less than 1.1%



Program Name	O/H
400.perlbench	-1.9%
403.gcc	0.1%
433.milc	-7.2%
445.gobmk	-1.7%
456.hmmer	1.1%
458.sjeng	0.6%
464.h264ref	0.2%
482.sphinx3	0.5%

Evaluation (2)

- Pre-analysis time
  - Examines the property of a function
  - Longer than embedding, extracting time
    - One-time process for each program
- Deep investigation
  - Takes longer time than a quick search

Program Name	Quick search: Time with a regular expression (sec)						Deep investigation: Time with a disassembly (sec)					
	Pre-Anal.	Em.	Ex. W1	Ex. W2	Ex. W3	Avg.	Pre-Anal.	Em.	Ex. W1	Ex W2	Ex. W3	Avg.
400.perlbench	60	15	2.1	1.6	1.8	1.8	27,426	16	3,372	309	1,432	1,704
403.gcc	346	101	6.5	5.8	4.2	5.5	200,270	101	2,471	2,445	2,125	2,347
433.milc	2.2	0.8	0.3	-	-	0.3	12	0.8	6.8	-	-	6.8
445.gobmk	51	34	0.7	1.0	0.8	0.8	6,282	34	352	139	133	208
456.hmmer	5.9	1.7	0.6	0.5	0.9	0.7	437	1.7	38	57	46	47
458.sjeng	2.3	0.9	0.6	-	-	0.6	25	0.9	20	-	-	20
464.h264ref	13	3.5	1.2	1.2	1.1	1.2	1,358	3.5	94	75	64	78
482.sphinx3	3.4	1.5	0.3	0.4	-	0.3	102	1.5	15	16	-	15
addr2line	39	11	0.7	0.9	2.2	1.3	13,534	11	804	630	846	760
ar	40	11	1.2	0.9	0.6	0.9	14,982	12	797	812	782	797
bfdtest1	39	10	0.8	0.6	1.9	1.1	11,459	11	645	662	646	651
cxxfilt	39	12	0.7	0.9	2.1	1.2	13,045	12	735	653	865	751
nm-new	40	11	1.1	0.6	1.1	0.9	12,760	12	699	639	640	659
objcopy	58	14	3.6	4.0	1.1	2.9	28,679	15	822	1,044	987	951
objdump	87	23	4.3	0.9	4.5	3.2	37,856	23	901	1,521	1,243	1,222
ranlib	40	11	1.2	0.9	0.6	0.9	12,949	12	365	494	457	439
size	40	11	0.9	0.5	1.7	1.0	11,450	11	632	664	631	642
strings	39	11	0.8	0.6	2.2	1.2	13,972	11	701	656	669	675
strip-new	64	15	3.6	4.1	1.1	2.9	25,440	15	1,124	1,131	1,096	1,117
ctags	19	19	0.9	1.0	0.7	0.9	26,012	20	539	455	322	439
lighttpd	1.4	1.2	0.5	0.1	-	0.3	89	1.2	12	20	-	16
vsftpd	1.3	1.1	0.2	0.1	-	0.2	69	1.1	4.1	5.3	-	4.7
pscp	15	4.2	0.5	0.3	0.4	0.4	2,304	4.3	116	138	149	134
psftp	15	4.3	0.5	0.3	0.4	0.4	2,319	4.3	117	142	153	137
puttygen	2.4	2.6	0.5	0.3	0.2	0.3	555	2.8	41	50	29	40
cgtest	6.4	1.8	0.6	0.2	0.3	0.4	310	1.9	35	46	32	38

Evaluation (3)

- May be susceptible to a distortion attack (semantic preserving code transformation)
- Robust to operand distortion

- Function Relocation
  - Significant challenges to adversaries in practice
    (Function boundary detection on a stripped binary)

	stripped binary		
Program	Precision	Recall	F1 Score
400.perlbench	0.828	0.611	0.703
403.gcc	0.804	0.576	0.671
433.milc	0.859	0.653	0.742
445.gobmk	0.830	0.245	0.379
456.hmmer	0.880	0.505	0.642
458.sjeng	0.777	0.626	0.693
464.h264ref	0.856	0.693	0.755
482.sphinx3	0.843	0.594	0.697
addr2line	0.827	0.401	0.540
ar	0.832	0.414	0.553
bfdtest1	0.816	0.396	0.533
cxxfilt	0.836	0.406	0.547
nm-new	0.843	0.421	0.562
objcopy	0.826	0.464	0.594
objdump	0.818	0.475	0.601
ranlib	0.798	0.400	0.533
size	0.840	0.406	0.547
strings	0.803	0.393	0.527
strip-new	0.831	0.470	0.600
ctags	0.845	0.576	0.685
lighttpd	0.799	0.735	0.766
vsftpd	0.854	0.791	0.821
pscp	0.802	0.621	0.700
psftp	0.791	0.622	0.697
puttygen	0.780	0.515	0.620
cgtest	0.789	0.524	0.630

What we have not talked about

- Current limitations of SoftMark
  - Binary packing: unpack required
  - Semantic-preserving code transformation
    - Opcode distraction may lower a survival rate
- Collision with a function relocation
- Constraints on function relocation with CCR
- Capacity of other existing techniques

Please read our paper!

Conclusions

- Stealthier and robust watermark scheme
- Negligible performance overhead
- Reasonable capacity
- Practical and efficient in watermark embedding and extracting

KOREA UNIVERSITY

UNIVERSITY of VIRGINIA

SUNG KYUN KWAN UNIVERSITY

Thank you

Honggoo Kang

Ph.D student in Korea University

honggoonin@korea.ac.kr