Hardware-Assisted Malware Analysis

Marcus Botacin[1], André Grégio[3], Paulo Lício de Geus[2]

[1]Msc. Computer Science
Institute of Computing - UNICAMP
marcus@lasca.ic.unicamp.br

[2]Advisor
Institute of Computing - UNICAMP
paulo@lasca.ic.unicamp.br

[3]Co-Advisor
Federal University of Paraná (UFPR)
gregio@inf.ufpr.br

CTD - SBSEG

Topics

Topics

Malware Threats.


The Washington Post
Democracy Dies in Darkness

Europe

Malware, described in leaked NSA documents, cripples computers worldwide

Figure: Washington Post: `https://tinyurl.com/ljo7ekm`

BBC | BBC iD | Menu | Search

NEWS | Sections

Massive ransomware infection hits computers in 99 countries

13 May 2017 | Technology

Figure: BBC: `https://tinyurl.com/mfogjhe`

Topics

Malware Analysis.

Figure: Imperva: `https://tinyurl.com/zkbsnl2`

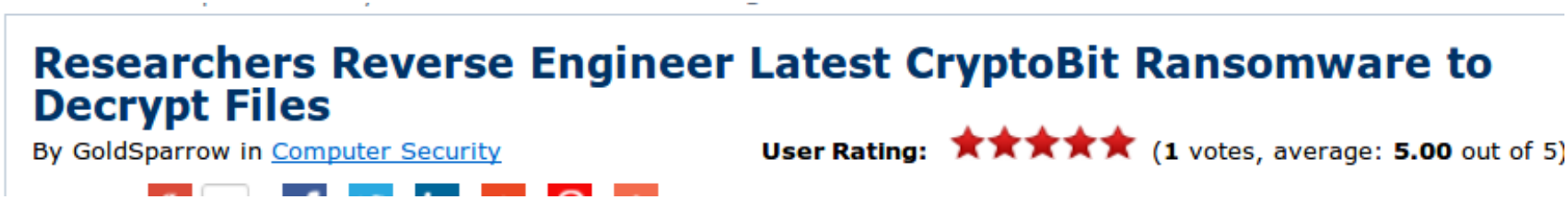

Figure: Enigma: `https://tinyurl.com/kydgwve`

Topics

The Challenges.

PetrWrap Crypto Ransomware Blocks Security Researchers From Reverse Engineering Code Samples

JP Buntinx March 16, 2017 News, Security

Figure: Themerkle: `https://tinyurl.com/kasuxcr`

MAY 13, 2017 @ 04:01 AM 95,046 The Little Black

How One Simple Trick Just Put Out That Huge Ransomware Fire

Figure: Forbes: `https://tinyurl.com/l7ecrex`

How to stop analysis?

Table: **Anti-Analysis**: Tricks summary. Malware samples may employ multiple techniques to evade distinct analysis procedures.

Technique	Description	Reason	Implementation
Anti Debug	Check if running inside a debugger	Blocks reverse engineering attempts	Fingerprinting
Anti VM	Check if running inside a VM	Analysts use VMs for scalability	Execution Side-effect
Anti Disassembly	Fool disassemblers to generate wrong opcodes	AV signatures may be based on opcodes	Undecidable Constructions

And then...

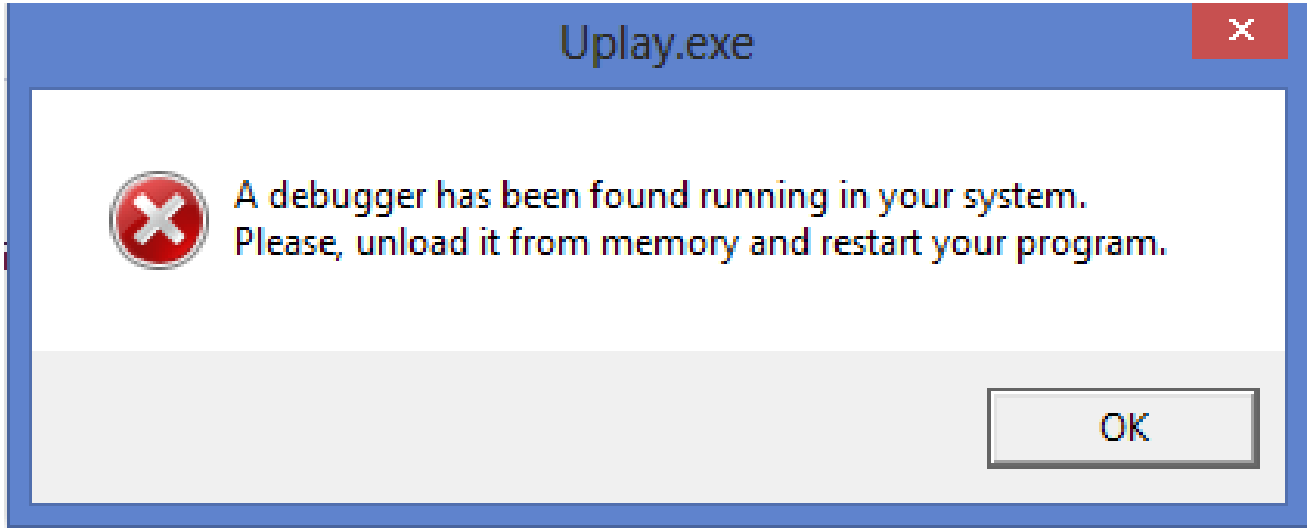

Figure: Commercial solution armored with anti-debug technique.

And then...

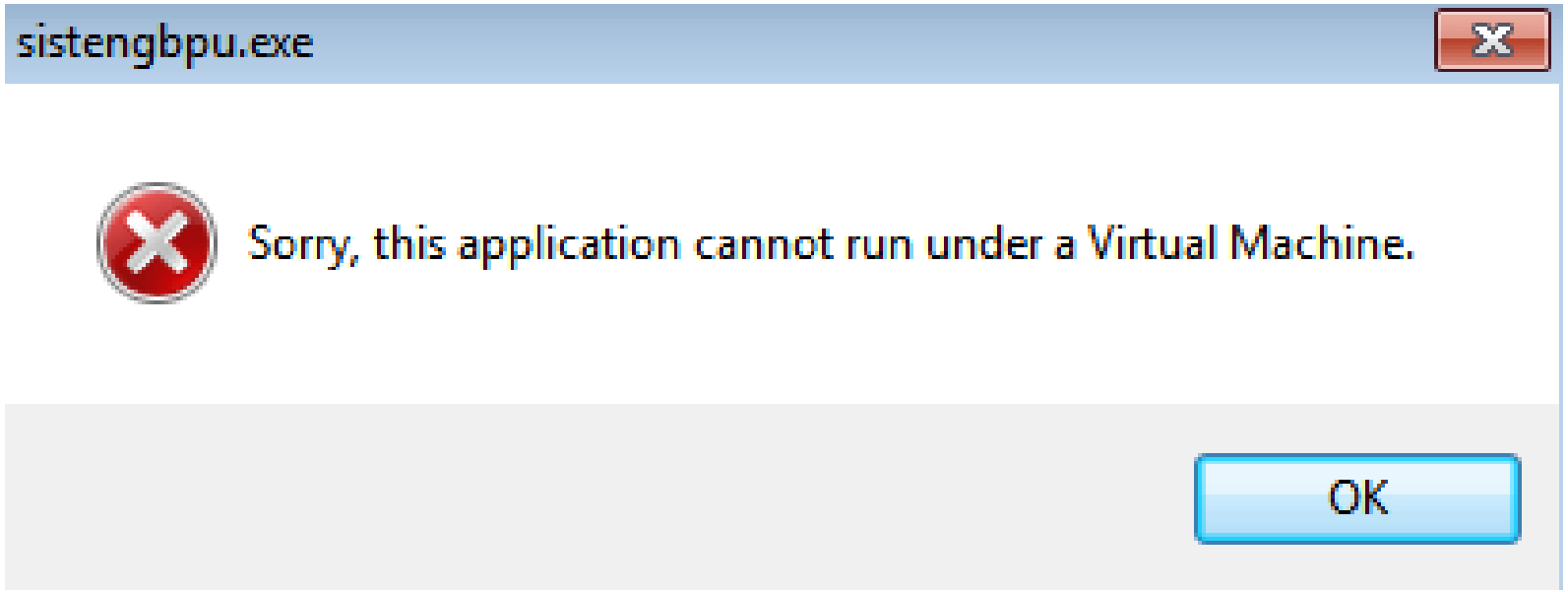

Figure: Real malware impersonating a secure solution which cannot run under an hypervisor.

Topics

Transparency

1. **Higher privileged.**
2. **No non-privileged side-effects.**
3. **Identical Basic Instruction Semantics.**
4. **Transparent Exception Handling.**
5. **Identical Measurement of Time.**

Topics

A Survey

Table: **Hardware features**.

Technique	PROS	CONS	Gaps
HVM	Ring -1	Hypervisor development	High overhead
SMM	Ring -2	BIOS development	High implementation cost
AMT	Ring -3	Chipset code change	No malware analysis solution
HPCs	Lightweight	Context-limited information	No malware analysis solution
GPU	Easy to program	No register data	No introspection procedures
TSX	Commit-based	Store only few KB	Overcome the KB barrier
SGX	Isolates goodware	Also isolates malware	No enclave inspection
SOCs	Tamper-proof	Passive components	Raise alarms

Topics

Branch Monitors

FROM	TO
0x0FFFF418	0x0FFF8F36
0x0FFF1510	0x0FFFCF2E
0x0FFF8014	0x0FFF0523
0x0FFF81b3	0x0FFFE057

Figure: Branch Stack.

Topics

Proposed Framework

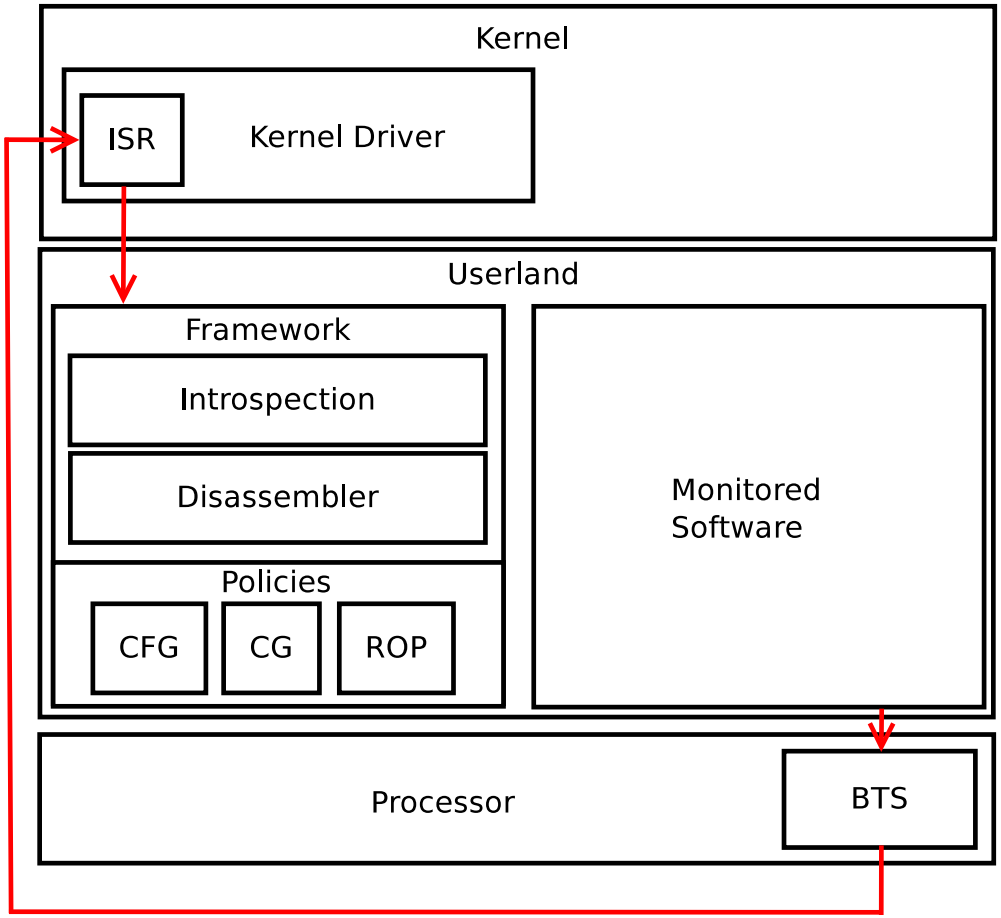

Figure: Proposed framework architecture.

Could I develop a performance-counter-based malware analyzer?

Could I isolate processes' actions?

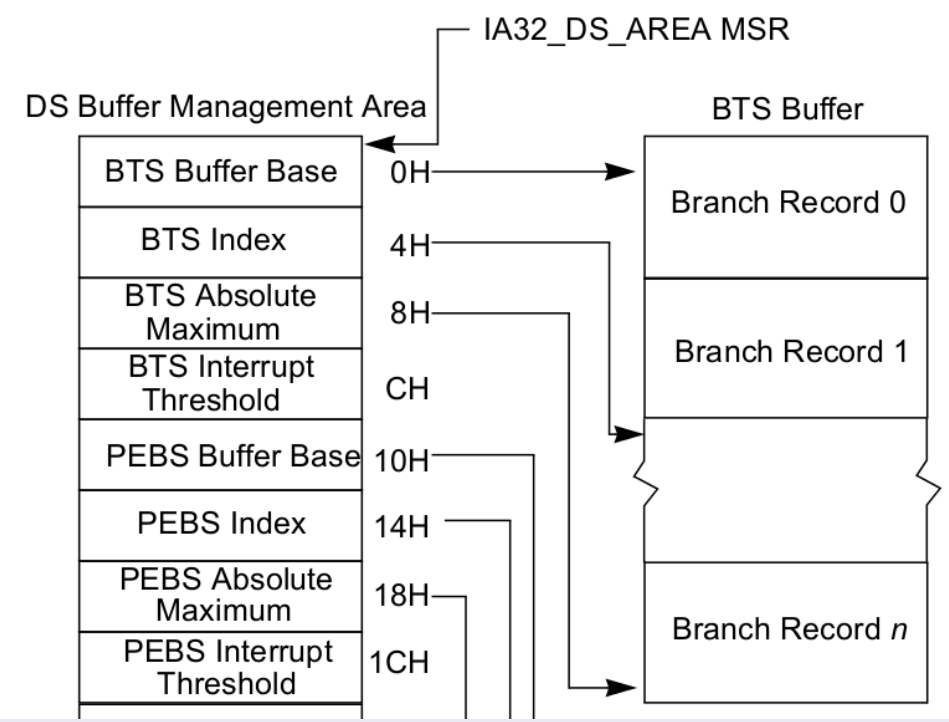

Figure: Data Storage (DS) AREA.

Could I develop a performance-counter-based malware analyzer?

Is CG reconstruction possible?

Table: ASLR - Library placement after two consecutive reboots.

Library	NTDLL	KERNEL32	KERNELBASE
Address 1	0xBAF80000	0xB9610000	0xB8190000
Address 2	0x987B0000	0x98670000	0x958C0000

Could I develop a performance-counter-based malware analyzer?

Is CG reconstruction possible?

Table: Function Offsets from `ntdll.dll` library.

Function	***Offset***
NtCreateProcess	0x3691
NtCreateProcessEx	0x30B0
NtCreateProfile	0x36A1
NtCreateResourceManager	0x36C1
NtCreateSemaphore	0x36D1
NtCreateSymbolicLinkObject	0x36E1
NtCreateThread	0x30C0
NtCreateThreadEx	0x36F1

Could I develop a performance-counter-based malware analyzer?

Is CG reconstruction possible?

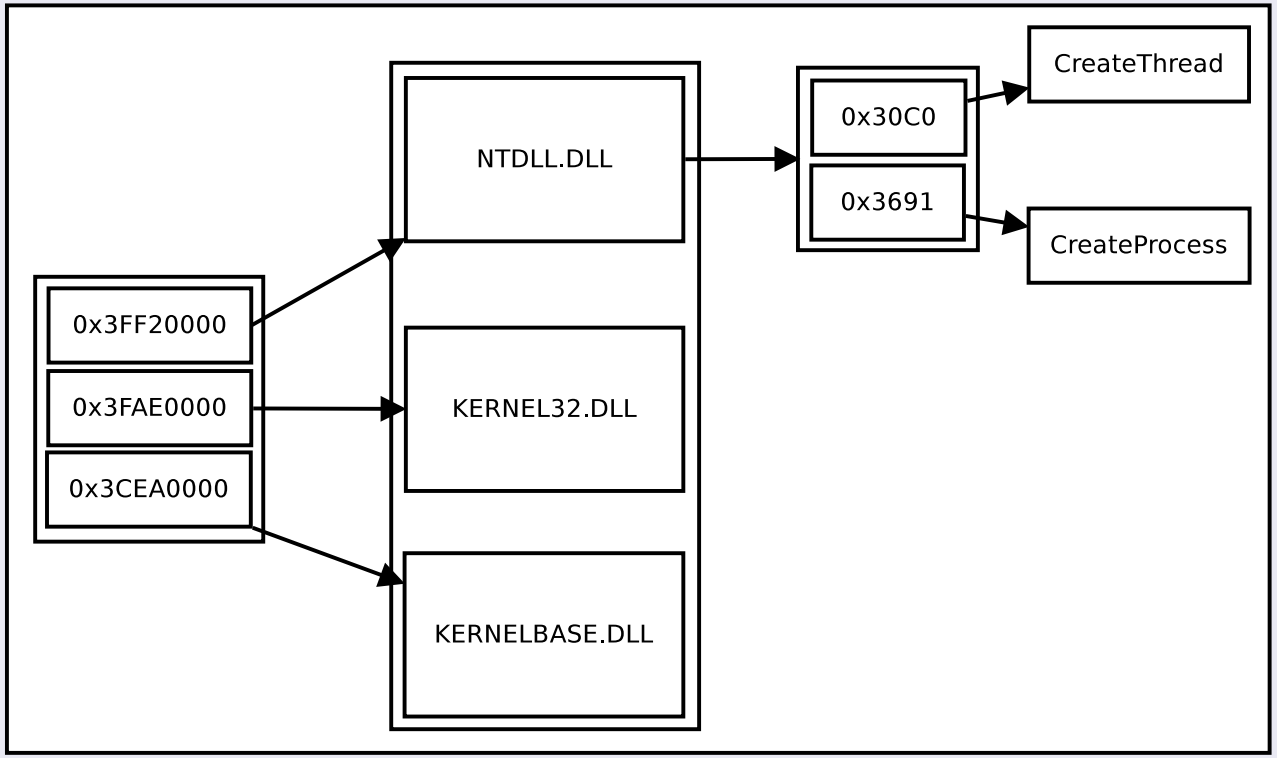

Figure: Introspection Mechanism.

Could I develop a performance-counter-based malware analyzer?

Is CG reconstruction possible?

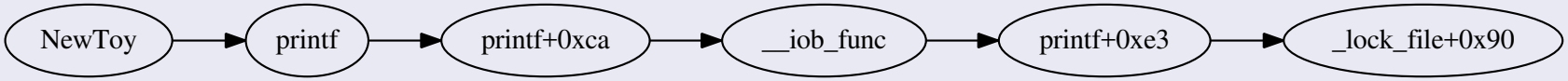

Figure: Step Into.

Figure: Step Over.

Could I retrieve all executed functions ?

Is CFG reconstruction possible?

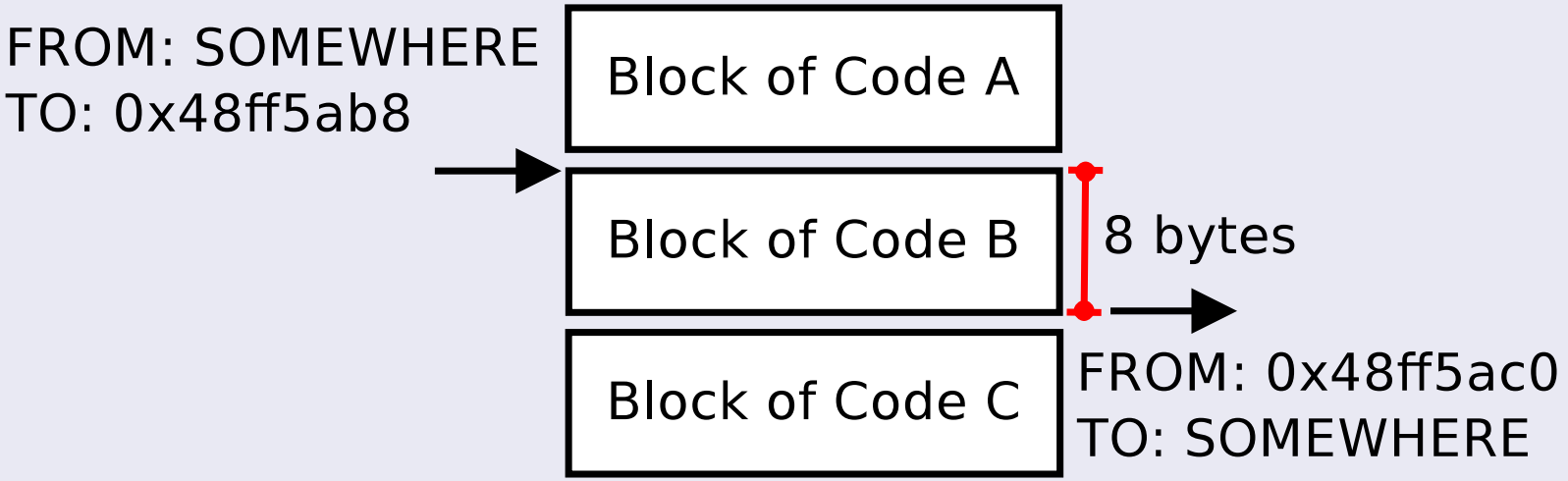

Figure: Code block identification.

Could I retrieve all executed functions ?

Is CFG reconstruction possible?

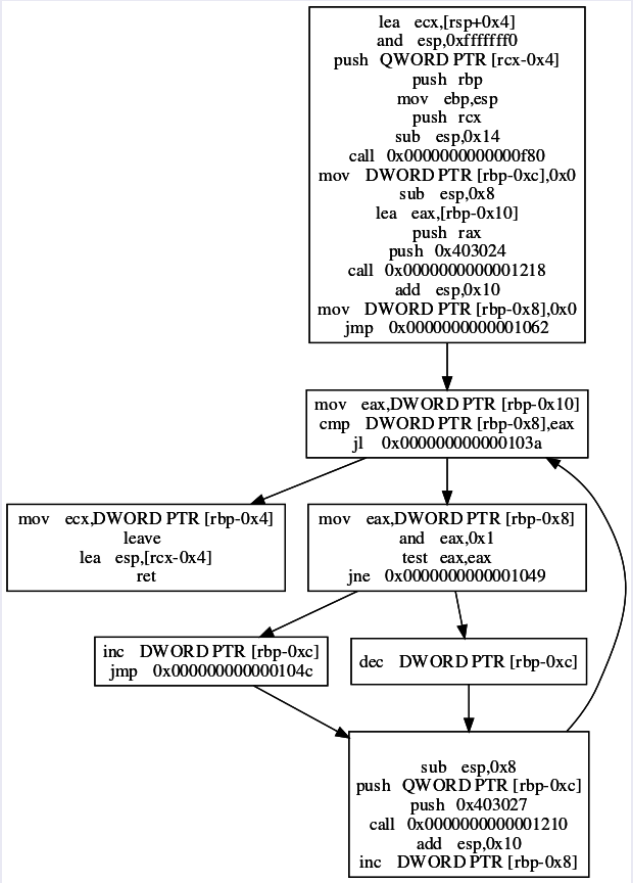

Figure: **it is possible to reconstruct the whole execution flow.**

Is the final solution transparent?

Deviating Behavior

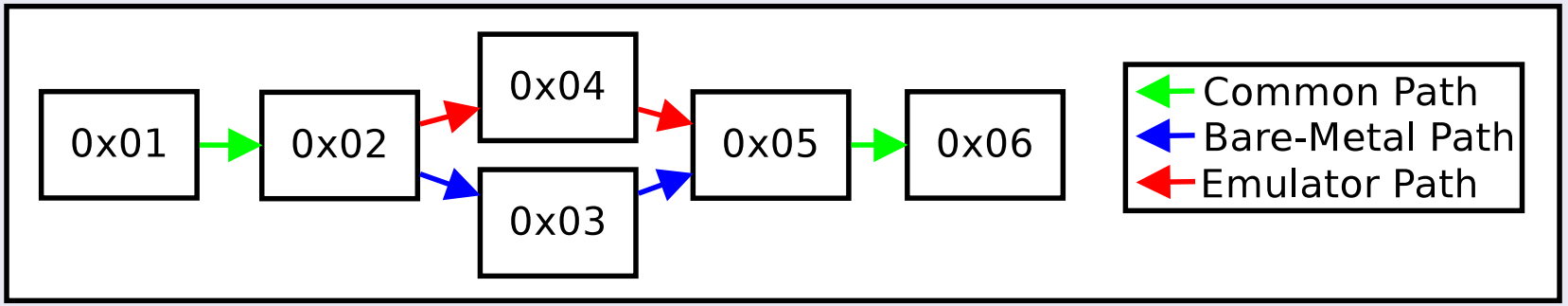

Figure: Deviating behavior identification.

Is the final solution transparent?

Deviating Behavior

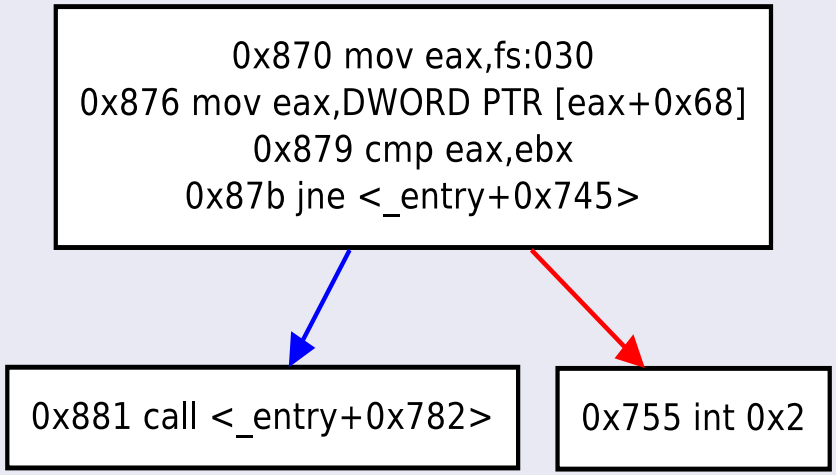

Figure: Divergence: True Positive.

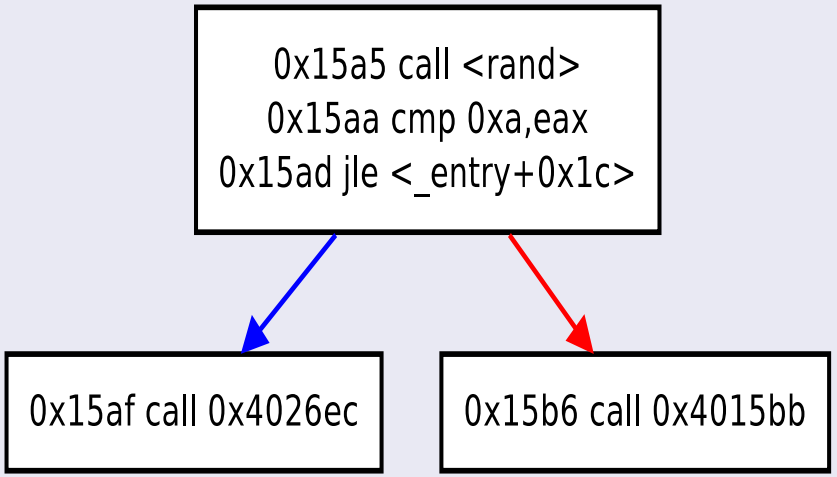

Figure: Divergence: False Positive.

Could I develop a Debugger?

Inverted I/O

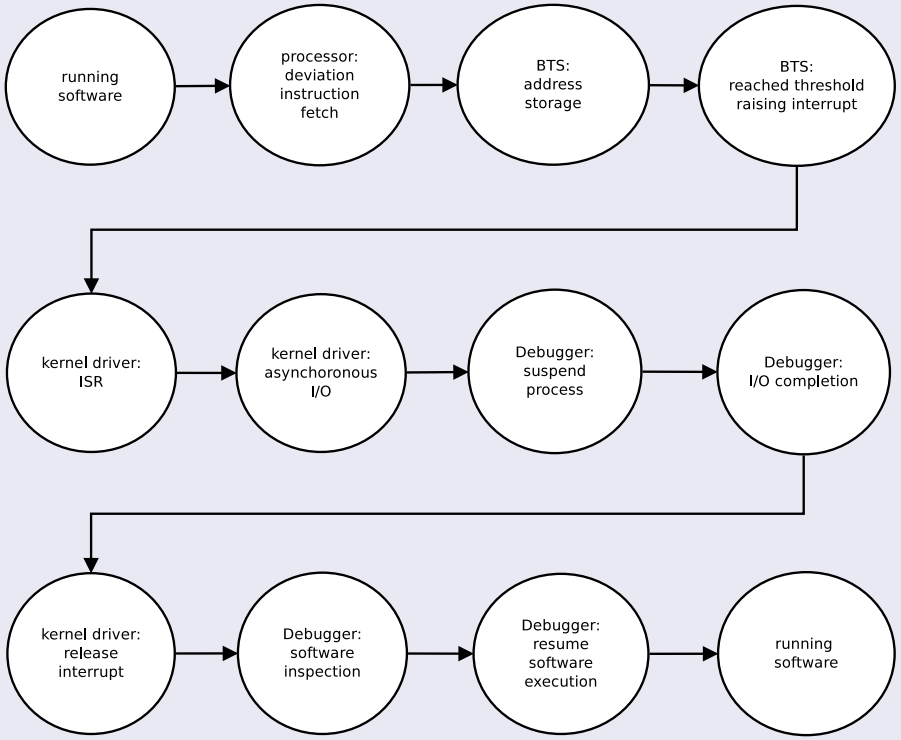

Figure: Debugger's working mechanism.

Could I develop a Debugger?

Integration

```
ubuntu@ubuntu-VirtualBox: ~
(gdb) target remote 192.168.1.106:5000
Remote debugging using 192.168.1.106:5000
0x0007f712 in ?? ()
(gdb) info registers
eax            0x1      1
ecx            0x2      2
edx            0x3      3
ebx            0x4      4
```

Figure: GDB integration.

Does the solution handle ROP attacks?

ROP Attacks

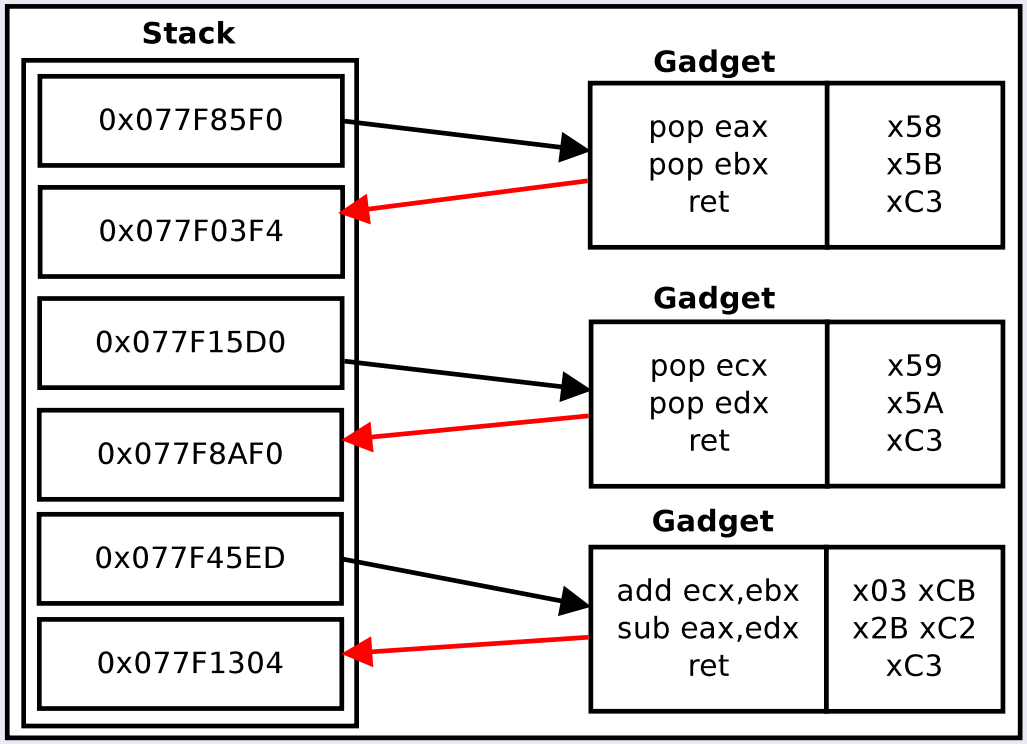

Figure: ROP chain example.

Does the solution handle ROP attacks?

CALL-RET Policy

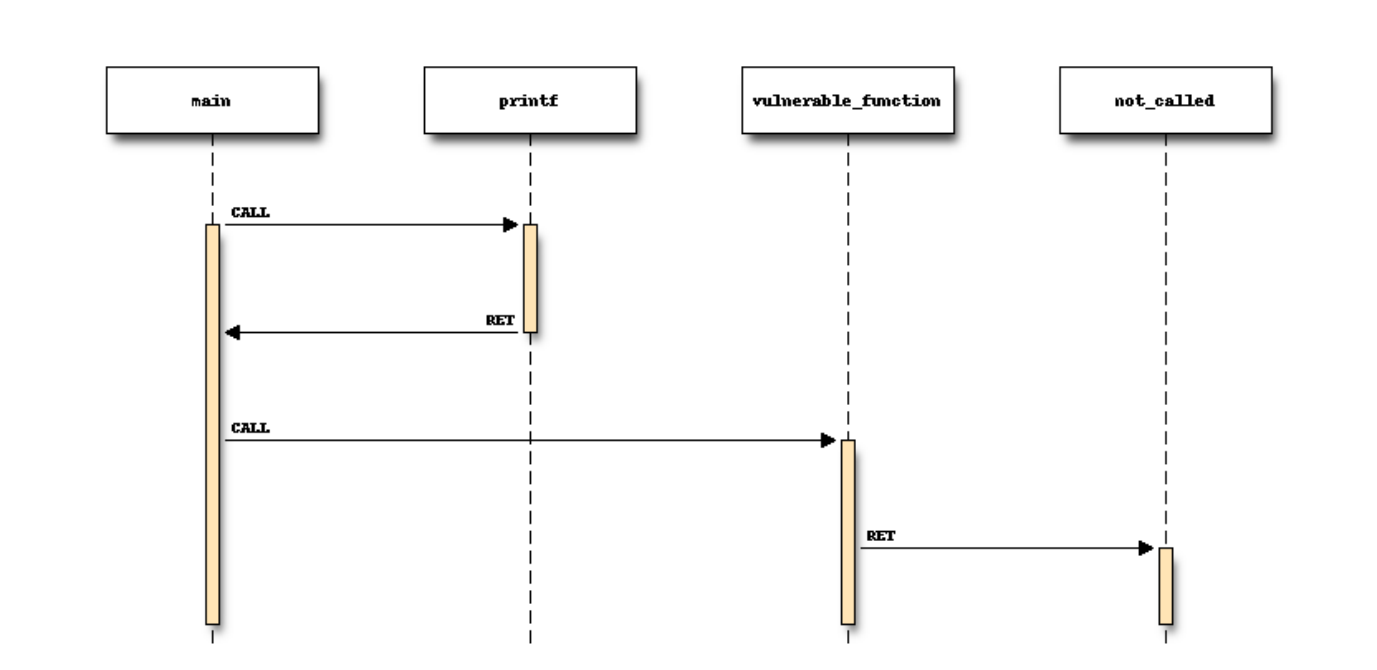

Figure: CALL-RET CFI policy.

Does the solution handle ROP attacks?

Exploit Analysis

Table: Excerpt of the branch window of the ROP payload.

FROM	TO
—	0x7c346c0a
0x7c346c0b	0x7c37a140
0x7c37a141	—

Does the solution handle ROP attacks?

Exploit Analysis

Listing 1: Static disassembly of the `MSVCR71.dll` library.

```
7c346c08: f2 0f 58 c3          addsd   %xmm3,%xmm0
7c346c0c: 66 0f 13 44 24 04    movlpd  %xmm0,0x4(%esp)
```

Listing 2: Dynamic disassembly of the `MSVC71.dll` executed code.

```
0x1000 (size=1)  pop    rax
0x1001 (size=1)  ret
```

Topics

Lessons learned

- Transparency is essential.
- Hardware-assisted approaches may fulfill transparency requirements.
- There are open problems on hardware monitoring.
- Security, performance, and development efforts as trade-offs (really?).
- Performance monitors as lightweight alternatives.

Topics

And now?

- Multi-core monitor.
- Linux Monitor.
- Malware clustering.

Topics

Main Papers

- *Who watches the watchmen: A security-focused review on current state-of-the-art techniques, tools and methods for systems and binary analysis on modern platforms*—ACM Computing Surveys (**A1**).
- *Enhancing Branch Monitoring for Security Purposes: From Control Flow Integrity to Malware Analysis and Debugging*—ACM Transactions on Privacy and Security (**A2**).
- *The other guys: automated analysis of marginalized malware*—Journal of Computer Virology and Hacking techniques (**B1**).

Solution's Availability

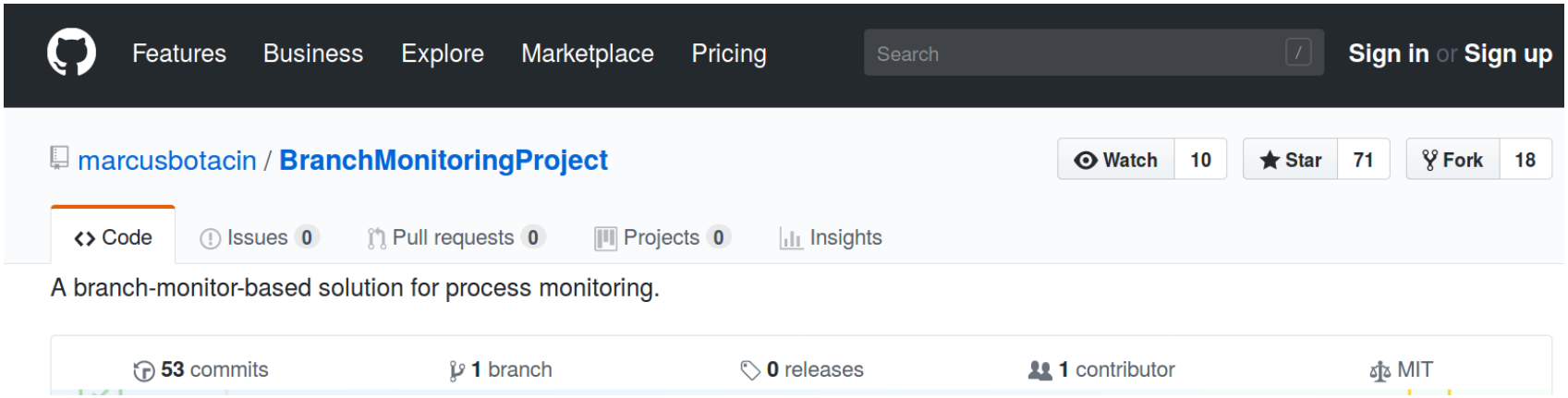

Figure: Solution's Availability. Solution is public on github.
`https://github.com/marcusbotacin/BranchMonitoringProject`

Questions ?

Contact

marcus@lasca.ic.unicamp.br
mfbotacin@inf.ufpr.br