

Inside the Machine

How Offensive Security is Defining the Way We Compute

Rodrigo Rubira Branco (BSDaemon)

rodrigo *noSPAM* kernelhacking.com

<https://twitter.com/bsddaemon>



**“There is no comfort in
the growth zone and no
growth in the comfort
zone”**

Unknown

HEAVEN is where the police
are British, the cooks are Italian,
the mechanics German, the lovers
Brazilian and it is all organized
by the Swiss.

HELL is where the chefs are
British, the mechanics Brazilian,
the lovers Swiss, the police
German, and it is all organized
by the Italians.

DISCLAIMER

I don't speak for my employer. All the opinions and information here are of my responsibility.

Personal:

"If hacking is an art, please excuse me, but I have poetic license"

- BSDaemon

I'm sorry in advance for the excessive usage of generalizations!

FYI: my wife is a lawyer ☺



Objectives

- I'm not here to try to teach the priests how to pray
 - Instead, I'll try to give sermon arguments (PoC || GTFO inspiration)
 - Propose some baseline for certain discussions (on mitigations and exploitability)
 - In this audience, I expect that most of the points themselves are well known, but hope that neglected ones will be called out and the most important ones will be re-enforced

Objectives

- I'm not here to try to teach the priests how to pray
 - Instead, I'll try to give sermon arguments (PoC || GTFO inspiration)
 - Propose some baseline for certain discussions (on mitigations and exploitability)
 - In this audience, I expect that most of the points themselves are well known, but hope that neglected ones will be called out and the most important ones will be re-enforced

Defining the objectives in one image?

Objectives

The attacker when I found a bug



The attacker when I propose a mitigation



Objectives – Halvar's MitiGator [1]



MitiGator: Raising the bar until no more exploits can be seen.

[1] Halvar Flake. "Weird Machines, Exploitability, and provable Non-Exploitability". Oxford Presentation 2018.

Three Points to Take Out (1/3)

- Threat Modeling is a relatively mature element of secure system engineering (where system can be hardware and/or software)
 - We have security objectives, attackers in scope (i.e.: physical presence, physical possession, remote, unprivileged, privileged), threats and features (relevant to demonstrate that the identified threats are addressed)
 - **But: We do not apply the same 'mature' process in defining mitigations (against unknown vulnerabilities)**
 - **Notice: Mitigations should be used/defined on top of usual quality in development, not to 'compensate' for the lack of it**

Three Points to Take Out (1/3)

As a mathematician once **joked** with me
“The Engineer’s Proof by Induction is slightly different:
If it works for $n=1$, $n=2$, $n=3$ it works for all cases”

So here is something to remember:

A new vulnerability mitigation intended to address classes of vulnerabilities should fully mitigate at least 3 real past vulnerabilities to be seriously considered.

Three Points to Take Out (2/3)

- There are not a lot of formalization of exploits (more on this later), which makes exploit writing a mix between deep engineering and art [1]
 - **Understanding a presentation and overall aspects of an exploit instance, is not the same as understanding how vulnerabilities **CAN BE** and **ARE** exploited** (oftentimes, other possible avenues, investigations that did not pan out, other engineering problems that needed to be solved – be it because of the tools that already proportioned it to the researcher, or because the researcher already had the knowledge in his mind - as the real dimensions of time spent are all missed)

[1] Excellent discussion in: “Are Reverse Engineering and Exploit Writing an Art or Science?” at NYUPoly THREADS Conference Panel (Cyber Security Awareness Week), Nov, 2013.

Three Points to Take Out (2/3)

So here is something else to remember:

Those proposing mitigations must understand the problem. Such understanding can be gained through experience writing real exploits (with real limitations and complexities) or by teaming up with someone with that experience.

Disclaimer: Being able to write exploits do not imply liking to do it (I do know developers that work in very good mitigations that do not like to write exploits, but have the full understanding and ability to do so)

The ideas expressed herein represent those of the presenter in his individual capacity.

Three Points to Take Out (3/3)

- When (we) researchers share the knowledge (be it in a presentation, paper, blog, informal conversations), oftentimes (we/they) simplify the problem
 - **Be it because of lack of space/time, lack of perception (sometimes we do not even see the bigger picture), Impostor Syndrome (they know so much about the problem that they believe they do not know enough and end up not discussing many aspects as they feel unfit to do so) – re-enforced by Dunning-Kruger on many in the industry (they know so little that they believe they really understand things) or many others**
- Unfortunately, as a result, there is a misperception of abilities in the industry (as it is believed there are many more people capable of designing proper mitigations than actually there are)
 - That essentially means additional broken mitigations, additional complexity, and additional challenges to actually adopt/propagate the mitigations that do work

Three Points to Take Out (3/3)

- To better describe the relation between Dunning-Kruger and Impostor Syndrome:

**“As the area of our knowledge grows, so too does the
perimeter of our ignorance”**

Neil deGrasse Tyson

Three Points to Take Out (3/3)

So here is the last thing to remember:

Investments in defense should benefit from the investments in offense (be guided by it) [1]. In the mitigations space, implementation details make all the difference and proper architectural definitions are not enough.

[1] Ben Hawkes. “Project Zero – Make 0day Hard”. CanSecWest 2015.

The ideas expressed herein represent those of the presenter in his individual capacity.

What do I miss? (1/2)

- Proper mitigation architecture and design is hard, and it is specially hard when data is not available (or can't be obtained without huge investments)
 - That is why I like so much presentations and posts on Pwn2Own achievements and security research performed (even if they are not releasing something super new, it is more data on what was done)
 - An additional benefit: making sure others can get into the state-of-the art on the field (instead of it become a dyeing art)
 - I would love if additional information on time spent, what was known before (like specific primitives, techniques) and others were also made available (Microsoft more recently makes some of it available, which help understand the rationale behind some of their decisions)

What do I miss? (2/2)

- Example of data that is not easily available: what is the % of memory corruption exploits in Metasploit (or overall, publicly available exploits) that bypass ASLR? DEP? CFG? Canary? Etc...
 - From the ones that do not, which ones would need additional bugs? (versus ones that the exploit is just a PoC but could already have bypassed the mitigation with the obtained primitives?)
- While it looks like only a data collection problem (from what is already available), it does require a lot of knowledge to dig deeper to properly answer the 'was it possible anyway' question

The Importance of Formalization

- In the past years we've been witnessing an exceptionally important trend in computer security towards formalizing offensive work
 - That is essential to finally migrate from a mix art/engineering form to purely engineering (with all the needed talent that engineering requires)
 - It is also a key element in bringing the academia towards the state-of-the-art and with that, scale offense (currently a big challenge)
- From that trend, terms like weird machine appeared, LangSec gained much more traction/support
 - Opening doors for everyone to understand that data is what drives code, that parsers are recognizers and creating new concepts on secure development
 - But it also demonstrated that even some basic assumptions are in conflict with what was taught by defense

LangSec x Secure Development

- In current secure development guidance, it is a 'rule' that input is checked as close as possible to the use of the input, which makes 'shotgun' parsing acceptable – while LangSec proposes the recognition to happen at the entry point
- In current development guidance, the rule of intrinsically secure functions seem to make sense (after all, many vulnerabilities arise from copy+paste code from one project to another – without copying the input handling of the parameters – or by adding new functionality – and as so, new paths to vulnerable functions). But once again, due to real life restrictions (development time and performance of having duplicated checks), recognizing everything *and* adding specific checks is hard in practice. Somehow more guidance/discussions are still needed even in basic areas

More formalizations

- In recent years, we've also seen the work by Julien Vanegue on heap modeling for exploit writing [1]
 - Which helps rationalizing between different mitigations
 - And move us from art to science state in the subject (improving toolset, enabling real/better automation)
- And more recently, Halvar's paper [2] providing a theoretical (mathematical) framework to define exploitation (of memory corruptions)

[1] Julien Vanegue; "Heap Models for Exploit Systems"; IEEE S&P LangSec Workshop; 2015

[2] Halvar Flake; "Weird machines, exploitability, and provable unexploitability"; 2018

Not a comprehensive list

- Complexity has been identified in many areas, an example that I like
"a modern compressor is a bit like a compiler. The compressed data is a kind of program in bytecode, and the decompressor is just an interpreter that runs that bytecode"
Link: <http://cbloomrants.blogspot.com>
- Obviously, my intent is not to be comprehensive, but to demonstrate an important trend
 - There were many earlier tries on defining exploit writing, like the work by Gerardo Richarte [1]
 - And the conscious effort by Microsoft on using the understanding on exploits to drive mitigation strategies, shared in multiple presentations along the years - See [2] for the latest, which includes HW in the threat scope
 - The ROP definition for example, was very well known by practitioners (exploit writers), but only got widely understood once properly defined (in an academic paper) [3]

[1] Gerardo Richarte; "About exploit writing"; 2002

[2] David Weston; "Hardening with Hardware"; BlueHat IL 2018

[3] H. Shacham; "The geometry of innocent flesh on the bone: return-into-libc without function calls (on the x86)". CCS 2007

Once upon a time...

- ... A professor of mathematics (Nei Soma)
- In a computer engineering grad school (Airforce Technology Institute – ITA in Brazil)
- Had a student that had as job title: “Exploit writer”
- So, the professor had the idea: **Define your work using set theory**

What now?

- The work itself got lost, but never fully forgotten
 - I kept using the ideas in discussions
 - Until I finally met some **motivated individuals** that really incentivized (and worked with me) to better elaborate and put on paper the idea
 - A full paper is available “A Mathematical Modeling of Exploitations and Mitigation Techniques Using Set Theory”: <http://spw18.langsec.org/papers/Kawakami-Exploit-modeling-using-set-theory.pdf>
 - For anyone interested in the actual formulas and details
- Here are their names:
 - Kekai Hu
 - Ke Sun
 - Henrique Kawakami

Set Theory Mitigation and Exploit Modeling

A quick glance

- Exploit Primitives (EP) - attack ability that can be achieved from a security vulnerability
 - Each primitive is composed of two elements: type and property
 - Type: read, write, execute
 - 5 major primitive properties (as defined by PaX Team [1]):
 - Arbitrary Addresses (AA), Arbitrary Content (AC), Arbitrary Operation (AO), Arbitrary Number of Times (AN) and Arbitrary Time (AT)
 - **Limitation: We did not model partial.**
- Exploit Objective (EO) is the ultimate goal of an attacker against the analyzed/modeled system (E.g: Attacker wants administrative access in a remote system).
 - Represents the concept of chained vulnerabilities used by one exploit (un-ting the relation of one exploit to one vulnerability, that is the classical view by non exploit writers)
- Exploit Condition (EC) is the minimal required combination of EP in a system to achieve an exploit objective (EO)
- Exploit Difficulty (ED): we propose the usage of Big O notation to define ED and measure mitigation quality

[1] PaX Team; “RAP: RIP ROP”; Hackers 2 Hackers Conference (H2HC); 2015

Probabilistic x Deterministic Mitigation

- Deterministic mitigation (DM) fully eliminate (or with an un-computable order of chances probabilistically eliminate) at least one EP (**emphasis: there is no 1:1 relation between vulnerability and primitives, and vulnerability and exploits**)
 - Properly applied encryption for at rest data [1], for example, can offer DM (if we assume the cipher indeed requires un-existent amounts of computation to be broken). It adds $O(\infty)$ complexity
 - * if the data is not at rest (like in the DRAM encryption case), encryption (without integrity) adds non-determinism for the attacker that can be bypassed [1]
- Probabilistic mitigation (PM) is a mitigation that **significantly** increases ED
 - *AND* reduces the successful rate of its target EP
 - It adds $O(n)$ complexity

[1] Gueron, Shay; Branco, Rodrigo. “Blinded Random Corruption Attacks”. IEEE HOST 2016

Three Points to Take Out (My conclusions!)

- A new vulnerability mitigation intended to address classes of vulnerabilities should fully mitigate at least 3 real past vulnerabilities to be seriously considered
- Those proposing mitigations must understand the problem. Such understanding can be gained through experience writing real exploits (with real limitations and complexities) or by teaming up with someone with that experience
- Investments in defense should benefit from the investments in offense (be guided by it). In the mitigation's space, implementation details make all the difference and proper architectural definitions are not enough

The end!! Really is !?

Rodrigo Rubira Branco (BSDaemon)

rodrigo *noSPAM* kernelhacking.com

<https://twitter.com/bsddaemon>



**“There is no comfort in
the growth zone and no
growth in the comfort
zone”**

Unknown