A Mathematical Model of Exploitation and Mitigation Techniques Using Set Theory

STORM

STrategic Offensive Research & Mitigations

Intel Corporation

Rodrigo Branco, Kekai Hu, **Henrique Kawakami**, and Ke Sun.

Motivation

- One of the most challenging problems in computer security is formalization of vulnerabilities, exploits, mitigations and their relationship.
- In spite of various existing researches and theories, a mathematical model that can be used to quantitatively represent and analyze exploit complexity and mitigation effectiveness is still in absence.

Exploit Primitives (EP)

- An exploit primitive is an attack ability that can be achieved from a security vulnerability
- Each primitive is composed of two elements: type and property
  - Type: read, write, execute
  - Property: Associated with the type, to define location, timing, repeatability
  - We've used 5 major primitive properties (*):
    - Arbitrary Addresses (AA)
    - Arbitrary Content (AC)
    - Arbitrary Operation (AO)
    - Arbitrary Number of Times (AN)
    - and Arbitrary Time (AT)
    - **Limitation: We did not model partial.**

(*) Definition taken from: "RAP: RIP ROP" Presentation @H2HC 2015 by PaX Team

Exploit Objective (EO)

- Exploit Objective is the ultimate goal of an attacker against the analyzed/modeled system:
  - EO: Attacker wants administrative access in a remote system
  - EO: Attacker wants to get code execution once a spreadsheet is visualized on specific systems

- EO is used to represent 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)

- The exploit condition is the minimal required combination of EP in a system to achieve an exploit objective (EO)

- Exploit Condition is used to represent the grouping of multiple EP an attacker need to achieve her final objective (which can be limited to a given program, or can be used to analyze an entire system)
  - EC is always associated with an EO (for example, a simple arbitrary memory leak is *NOT* a security issue in a system that deals only with public information and in which the EO is to just access that information)
  - EC is the minimum requirement for an exploit to be successful. Thus, if any primitive of the EC is removed, the EC is not exploitable any more (would require new primitives to be acquired by the attacker)
  - One EO can have many different EC. If any one of the ECs is met, the EO can be achieved (take for example information leak in a process as EO: the attacker might not have a direct arbitrary read primitive, but if she has arbitrary execution, she can still achieve the EO)

Exploit Difficulty (complexity)

- We proposing the usage of Big O notation to define ED and measure mitigation quality
  - Big O has known limitations for performance usage
  - But deserves though on complexity definition for EC since it has a good way to show the difference between exponential increase (potentially useful) versus fixed increase (potentially un-useful)

- ED applies Big O notation to quantitatively describe the upper bound on the growth rate of the time and cost for an attacker to achieve an EC

- In a system with no mitigations, the complexity of an EP is O(1)

The Set Representation

- EP has two elements, type and property

  T = { All EP types } = { Read, Write, Execute }

  P = { All EP properties } = { AA, AC, AO, AN, AT }

- Thus, an EP can be represented as a combination of type t ϵ T and a property p ϵ P
  - ep = {t, p}
- EC is just a set of EP
  - ec = {ep1, ep2, ep3…, epn}

    = {{t1, p1}, {t2, p2}, {t3, p3}…,{tn,pn}}
- And the exploitability E of a given EO can be represented as a set of ECs:
  - E = {ec1, ec2, ec3…,ecn}

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**)
  - It adds O(∞) complexity

- Probabilistic mitigation (PM) is a mitigation that **significantly** increases ED *AND* reduces the successful rate of its target EP
  - It adds O(2^16) complexity for a 16-bit ASLR

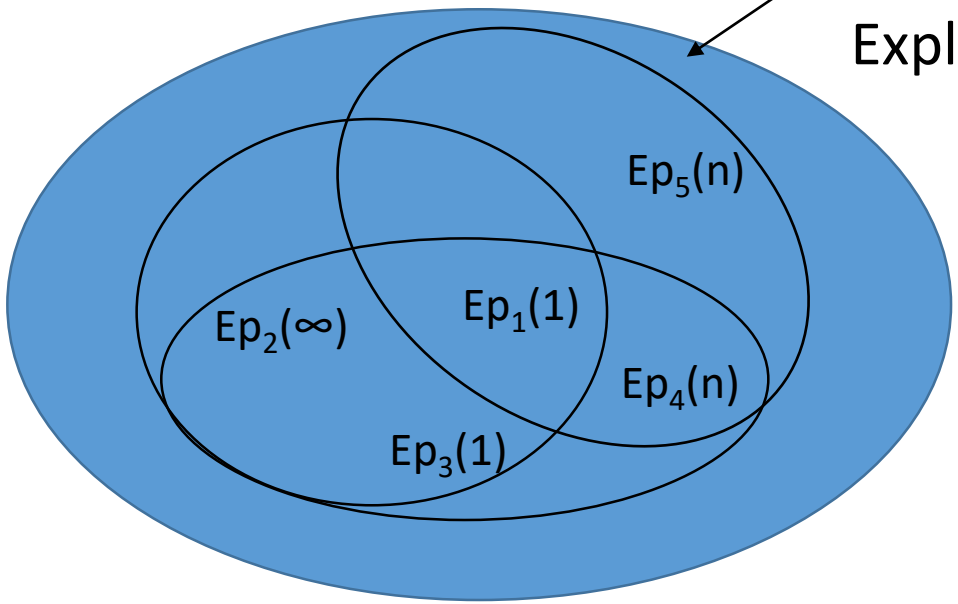
Adding the mitigations to the model

- We use the notation ep’ = ep(ed) to represent an exploit primitive with its exploit difficulty (without any mitigations, ep’ = ep(1) – as mentioned, O(1) is the complexity used for no mitigations)

- So a mitigation can be represented in a set of EPs with the added EDs
  - m = {ep’1(ed1), ep’2(ed2)…, ep’n(edn)}

  Where ed1 ϵ {O(1), O(n), O(∞)}

Graphical representation



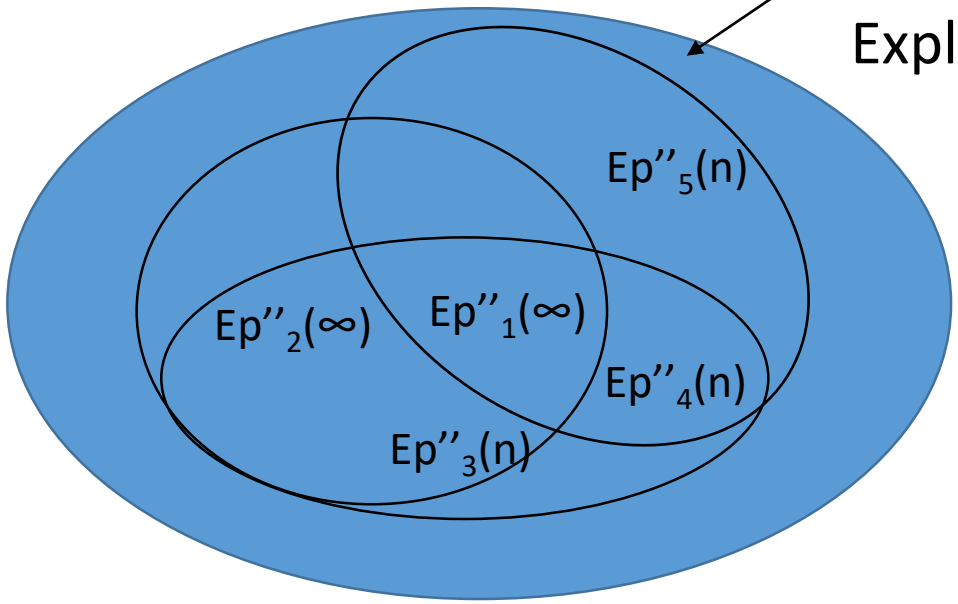
Exploit Primitives

Exploit Condition for a given Exploit Objective

Exploit Complexity -> $O(Ep_1) + O(Ep_4) + O(Ep_5)$

$= O(1) + O(n) + O(n)$

Graphical representation



Exploit Primitives after mitigations

Exploit Condition for a given Exploit Objective

Exploit Complexity -> $O(Ep_1) + O(Ep_4) + O(Ep_5)$

$= O(\infty) + O(n) + O(n)$

In this specific case, having a deterministic mitigation for Ep_1 mitigates all the exploit conditions. On the other hand, a mitigation for Ep_5 alone would not change overall Exploitability.

The end!

Henrique Kawakami
henrique.kawakami *nospam* intel.com