# On Exploit Quality Metrics – and How to Use Them for Automated Pentesting

Carlos Sarraute

CoreLabs & ITBA PhD program
Buenos Aires, Argentina

8.8 Security Conference – November 18, 2011

# Introduction

## My company: Core Security Technologies

- Boston (USA)
  - marketing, sales, engineering
- Buenos Aires (Argentina)
  - research and development

# Introduction

## My company: Core Security Technologies

- Boston (USA)
  - marketing, sales, engineering
- Buenos Aires (Argentina)
  - research and development

## CoreLabs: the research team

Some areas of interest:

- Vulnerability research
  - Bugweek
  - Publication of advisories
- Cyber-attack planning and simulation
- Improving OS detection using neural networks

# Agenda outline

# Agenda

# Penetration Testing?

## Penetration testing

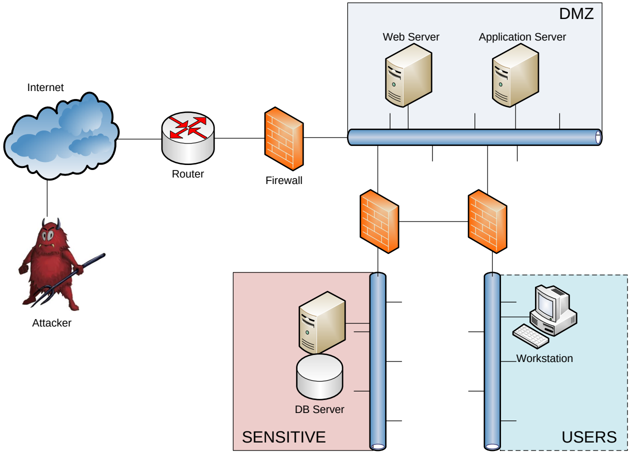
Actively verifying network defenses by conducting an intrusion in the same way an attacker would.

- Penetration testing tools have the ability to launch real exploits for vulnerabilities.
  - different from vulnerability scanners (Nessus, Retina, ...)
  - **no false positives!**
- Main tools available:
  - Core Impact (since 2001)
  - Immunity Canvas (since 2002)
  - Metasploit (since 2003)
    - open source, owned by Rapid7 since 2009

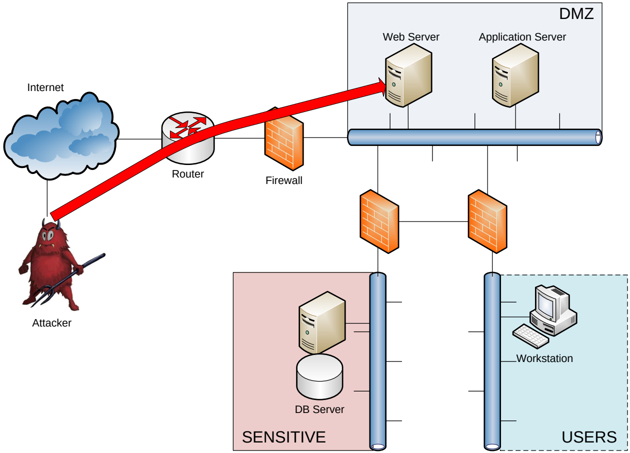
# Need for Automation

- Reduce human labor
- Increase testing coverage
  - Higher testing frequency
  - Broader tests trying more possibilities
- Complexity of penetration testing tools
  - More exploits
  - New attack vectors (Client-Side, WiFi, WebApps, . . . )
- Equip penetration testing tool with “expert knowledge”
- Construct attack plans that pivot.

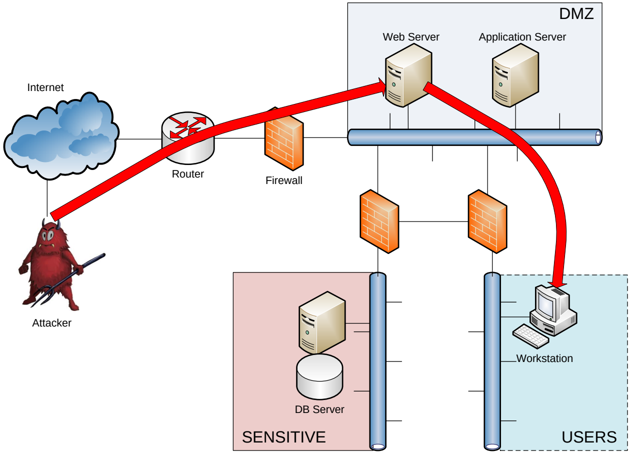
# Anatomy of a Real-World Attack



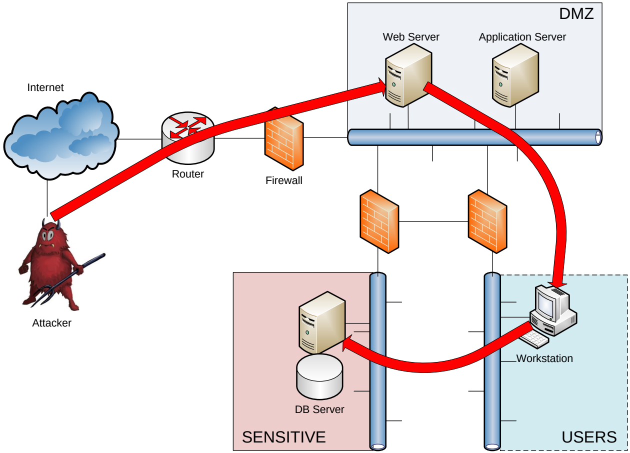
# Anatomy of a Real-World Attack



# Anatomy of a Real-World Attack



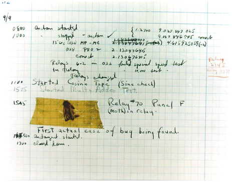
# Anatomy of a Real-World Attack



# Agenda

1. Motivation
2. On Exploit Quality Metrics
   - For different stakeholders
   - What can we measure?
3. An Efficient Planning Solution
   - Planning for dummies
   - Two Primitives
   - Using the Primitives in a Network Graph
   - Integrated with a Pentesting Tool
4. Demo
5. Summary

# Basic definitions



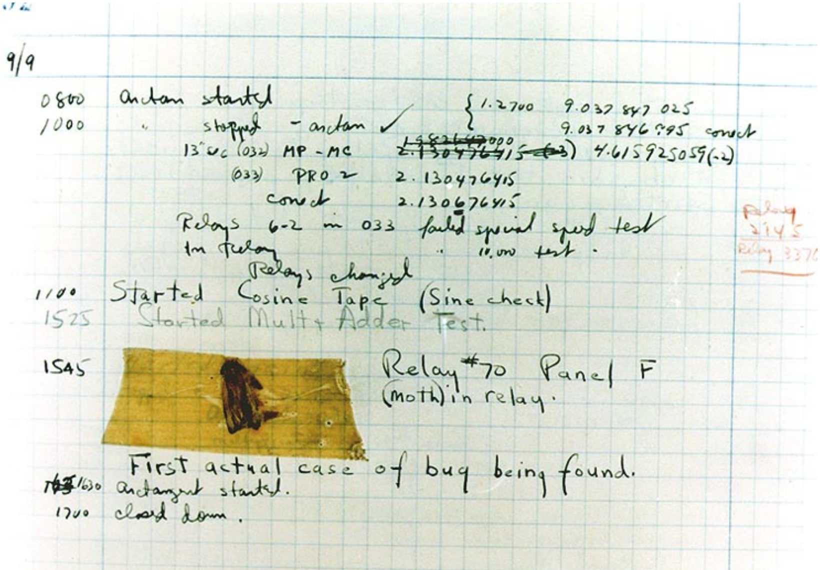

**Vulnerability (noun)** A flaw in a system that, if leveraged by an attacker, can potentially impact the security of said system

- Also: security bug, security flaw, security hole

**Exploit (verb)** To use or manipulate to one's advantage (Webster)

**Exploit (noun)** A security hole or an instance of taking advantage of a security hole

# Basic definitions



# Exploit Code

**Proof of Concept exploit - PoC (noun)** A software program or tool that exploits a vulnerability with the sole purpose of proving its existence.

**Exploit Code (noun)** A software program or tool developed to exploit a vulnerability in order to accomplish a specific goal.

- Possible goals: denial of service, arbitrary execution of code, etc

Reference: [Arc05]

# Agenda

# Users' profiles

- **Bad Guy** (Botnet Master)
  - Needs the exploit to be fast.
  - Will likely be running multiple instances.
  - Will run against multiple platforms in an automatic and massive fashion.
- **Penetration Tester** (or Bad Guy seeking a specific objective)
  - Someone trying to manually break into specific systems.
  - Maximize reliability in exploits for specific targets.
  - Exploit must survive real-world conditions
    $\longrightarrow$ unreliable or congested networks,
    $\longrightarrow$ high workload on the target computer.
  - Exploit should resist changes in application configurations.

# Engineering profiles I

- **Framework Developer** (Kernel, User Interface, etc.)
  - Interested in quality from a "software engineering" approach.
  - Quality also means including the documentation needed by that system.
- **Quality Assurance Analyst**
  - Documentation leading to a better assessment of the real capabilities of an exploit:
    $\longrightarrow$ Set of platforms and software versions targeted.
    $\longrightarrow$ Important configuration changes that must be made for the exploit to work.
  - Documentation will be used to design and/or execute test suites.
  - Regression testing: make sure those exploits for Windows 95 continue to work!

# Engineering profiles II

- **Exploit Writer**
  1. Support as many platforms as possible:
     $\longrightarrow$ platform = combination of OS versions and application versions.
     $\longrightarrow$ optimal = support all vulnerable platforms.
  2. The exploit as a piece of software easy to maintain and improve:
     $\longrightarrow$ code easy to understand $\Rightarrow$ less effort to add a platform or change the shellcode.
  3. Information about protections bypassed on each platform
  4. Well documented from a technical standpoint, especially when obscure techniques are used.

# Agenda

# Simple measurements

- Average running time
  - Straightforward to measure.
  - Some exploits require brute forcing
    $\longrightarrow$ sometimes that can be upgraded to more clever techniques
- Success rate or Probability of success
  - Success rate of testing an exploit repeatedly against a given platform.
  - Approximate different capacities, such as resilience to machine load, network load, or different configurations.
- Network traffic generated
- User required interaction
  - Determining if the exploitation of a bug will be “interactive” or unattended is an important piece of documentation.

# More complex measurements I

- Targets exploited / known vulnerable targets
  - A vulnerability affects a set of platforms, for example, Windows XP SP2 and SP3 can be affected.
  - Variations in libraries in intra-service-pack patches or when different languages are supported may affect the exploit.
- Resilience to changes in configuration and machine load
  - Exploit for a vuln may only work with the default configuration.
  - Exploit use methods (such as hardcoded address) that are sensitive to minor changes in memory layout.
  - Exploits are more reliable when non-default configurations are used during development, and when they are tested in real-life use conditions.

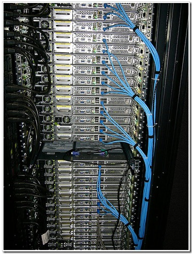
# More complex measurements II

- Number of bypassed protections
  - It's useful to know which security measures were bypassed.
  - Indication of how much knowledge was put into that exploit.
  - Results in better maintainability.
- Resilience to network traffic
  - Network traffic can affect a remote vulnerability due to timing issues,
  - . . . or when complex interactions are required to trigger the vuln.
  - Building proper testing environments is challenging.

# More complex measurements III

- Payload mutability
  - Some exploits will only work with a proof of concept payload.
  - The more versatile an exploitation technique is, the more adaptability the exploit will have.
- Used libraries in OS
  - Which specific part of the whole runtime and OS affect the exploit?
  - Helps back porting vulnerabilities to target more platforms.
  - Helps in gaining a better understanding of exploitability of vulnerabilities in a given OS.

# How do we measure those values?



1. Use the Exploit Testing team infrastructure.
   - 748 virtual machines with different OS and applications.
   - Automated execution of all the exploits against vulnerable images... every night!
   - Statistics are extracted from the database of executions.
2. Get feedback from users.
   - Anonymized feedback program in Core Impact.

# Agenda

# Agenda

1. Motivation
2. On Exploit Quality Metrics
   - For different stakeholders
   - What can we measure?
3. An Efficient Planning Solution
   - Planning for dummies
   - Two Primitives
   - Using the Primitives in a Network Graph
   - Integrated with a Pentesting Tool
4. Demo
5. Summary

# Simple brain teaser

In which order would you execute these exploits?

## An obvious problem

| *Action* | *Time* | *Probability* |
|---|---|---|
| $Exploit_1$ | $8s$ | $0,85$ |
| $Exploit_2$ | $100s$ | $0,05$ |

# Simple brain teaser

In which order would you execute these exploits?

## An obvious problem

| *Action* | *Time* | *Probability* |
|---|---|---|
| $Exploit_1$ | $8s$ | $0,85$ |
| $Exploit_2$ | $100s$ | $0,05$ |

## And maybe not so obvious

| *Action* | *Time* | *Probability* |
|---|---|---|
| $Exploit_1$ | $8s$ | $0,05$ |
| $Exploit_2$ | $100s$ | $0,85$ |

# Solution

We suppose the actions are independent, so the expected total running times are:

$$t_1 + (1 - p_1) \cdot t_2 <^? t_2 + (1 - p_2) \cdot t_1$$

# Solution

We suppose the actions are independent, so the expected total running times are:

$$t_1 + (1 - p_1) \cdot t_2 <^? t_2 + (1 - p_2) \cdot t_1$$

$$t_1 + t_2 - p_1 \cdot t_2 <^? t_2 + t_1 - p_2 \cdot t_1$$

# Solution

We suppose the actions are independent, so the expected total running times are:

$$t_1 + (1 - p_1) \cdot t_2 <^? t_2 + (1 - p_2) \cdot t_1$$

$$t_1 + t_2 - p_1 \cdot t_2 <^? t_2 + t_1 - p_2 \cdot t_1$$

$$p_2 \cdot t_1 <^? p_1 \cdot t_2$$

# Solution

We suppose the actions are independent, so the expected total running times are:

$$t_1 + (1 - p_1) \cdot t_2 <^? t_2 + (1 - p_2) \cdot t_1$$

$$t_1 + t_2 - p_1 \cdot t_2 <^? t_2 + t_1 - p_2 \cdot t_1$$

$$p_2 \cdot t_1 <^? p_1 \cdot t_2$$

$$\frac{t_1}{p_1} <^? \frac{t_2}{p_2}$$

# Solution and second brain teaser

## Best order

| *Action* | *Time* | *Probability* | $t/p$ |
|---|---|---|---|
| $Exploit_1$ | $8s$ | $0,05$ | $160$ |
| $Exploit_2$ | $100s$ | $0,85$ | $117,6$ |

# Solution and second brain teaser

## Best order

| *Action* | *Time* | *Probability* | $t/p$ |
|---|---|---|---|
| $Exploit_1$ | $8s$ | $0,05$ | 160 |
| $Exploit_2$ | $100s$ | $0,85$ | $117,6$ |

## What happens with more?

| *Action* | *Time* | *Probability* |
|---|---|---|
| $Exploit_1$ | $8s$ | $0,05$ |
| $Exploit_2$ | $100s$ | $0,85$ |
| $Exploit_3$ | $40s$ | $0,50$ |
| $Exploit_4$ | $2s$ | $0,01$ |

# Solution and second brain teaser

## Best order

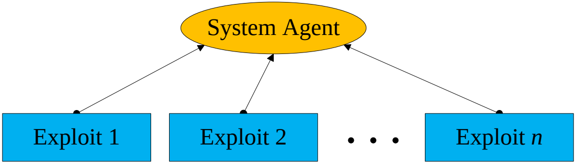
| *Action* | *Time* | *Probability* | *t/p* |
|---|---|---|---|
| $Exploit_1$ | $8s$ | 0, 05 | 160 |
| $Exploit_2$ | $100s$ | 0, 85 | 117, 6 |

## What happens with more?

| *Action* | *Time* | *Probability* | *t/p* | *Order* |
|---|---|---|---|---|
| $Exploit_1$ | $8s$ | 0, 05 | 160 | 3 |
| $Exploit_2$ | $100s$ | 0, 85 | 117, 6 | 2 |
| $Exploit_3$ | $40s$ | 0, 50 | 80 | 1 |
| $Exploit_4$ | $2s$ | 0, 01 | 200 | 4 |

# Agenda

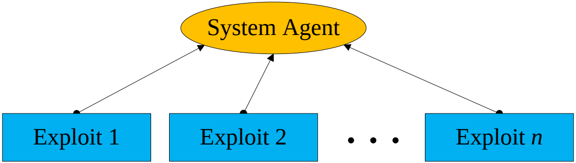
# The **Choose** primitive




## Problem

$\{A_1, \ldots, A_n\}$ independent actions that result in a goal $\mathfrak{g}$.
Each $A_k$ has probability of success $p_k$ and running time $t_k$.
**Task:** Find order of execution to minimize total running time.

# The **Choose** primitive



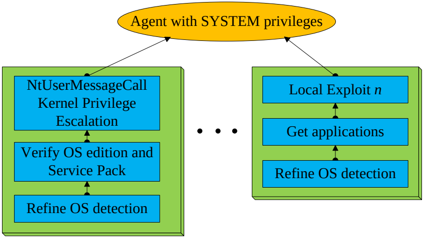

## Problem

$\{A_1, \ldots, A_n\}$ independent actions that result in a goal $\mathfrak{g}$.
Each $A_k$ has probability of success $p_k$ and running time $t_k$.
**Task:** Find order of execution to minimize total running time.

## Solution

Order actions according to $t_k/p_k$ (in increasing order).

# The **Combine** primitive




## Definition

We call *strategy* a group of actions that are executed in a fixed order.

## Problem

$\{G_1, \ldots, G_n\}$ are strategies that result in a goal $\mathfrak{g}$.
**Task:** Minimize total time.

# Expected probability and time

If the actions of $G$ are $\{A_1, \ldots, A_n\}$ then:
The expected running time of $G$ is

$$T_G = t_1 + p_1\, t_2 + p_1\, p_2\, t_3 + \ldots + p_1\, p_2 \ldots p_{n-1}\, t_n$$

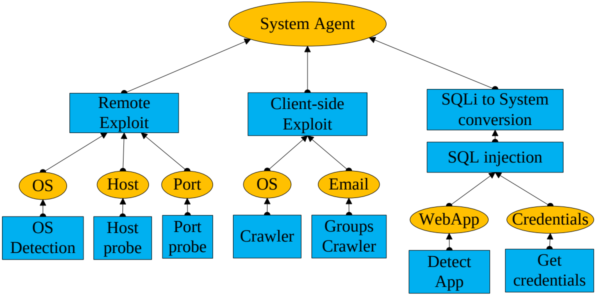
The probability of success is simply

$$P_G = p_1\, p_2 \ldots p_n$$

## Solution

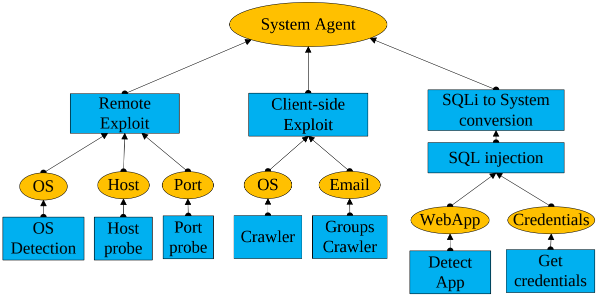
Sort the strategies according to $T_G/P_G$.
In each group, execute actions until one fails or all the actions are successful.
Complexity of planning: $\mathcal{O}(n \log n)$

# The **Combine** primitive (cont)




Groups of actions with an AND relation (order is not specified).

# The **Combine** primitive (cont)




Groups of actions with an AND relation (order is not specified).

**Idea**

In each group, order actions according to $t_k/(1 - p_k)$.

Intuitively, actions with higher probability of failure have priority.

# References (for this section)

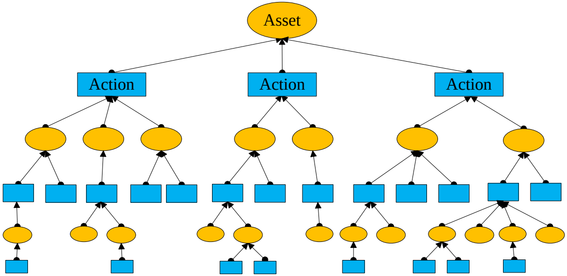
- [Sar09a] New Algorithms for Attack Planning
  - *FRHACK Conference, France. Sept 7/8, 2009.*
- [Sar09b] Probabilistic Attack Planning in Network + WebApps Scenarios
  - *H2HC Conference, São Paulo, Brazil. Nov 28/29, 2009.*

# Agenda

1. Motivation
2. On Exploit Quality Metrics
    - For different stakeholders
    - What can we measure?
3. An Efficient Planning Solution
    - Planning for dummies
    - Two Primitives
    - Using the Primitives in a Network Graph
    - Integrated with a Pentesting Tool
4. Demo
5. Summary

# **First level:** fixed source and target

Given a source machine and a target machine, the problem is to find a path in an Attack Tree:




1. *Action node:* connected by AND relation with its requirements $\longrightarrow$ use *Combine* primitive.
2. *Asset node:* connected by OR relation with the actions that provide that asset $\longrightarrow$ use *Choose* primitive.

# **Second level:** graph of machines

Use First level procedure to compute $Time(u,v)$ and $Prob(u,v)$ for all $u,v \in \mathcal{V}$ and then ...

**Algorithm 1** Modified Dijkstra's algorithm

$T[s] = 0,\ P[s] = 1$
$T[v] = +\infty,\ P[v] = 0 \quad \forall v \in \mathcal{V}, v \neq s$
$S \leftarrow \emptyset$
$Q \leftarrow \mathcal{V}$ (where $Q$ is a priority queue)
**while** $Q \neq \emptyset$ **do**
  $u \leftarrow \arg\min_{x \in Q} T[x]/P[x]$
  $Q \leftarrow Q \backslash \{u\},\ S \leftarrow S \cup \{u\}$
  **for all** $v \in \mathcal{V} \backslash S$ adjacent to $u$ **do**
    $T' = T[u] + P[u] \times Time(u,v)$
    $P' = P[u] \times Prob(u,v)$
    **if** $T'/P' < T[v]/P[v]$ **then**
      $T[v] \leftarrow T'$
      $P[v] \leftarrow P'$
**return** $\langle T, P \rangle$

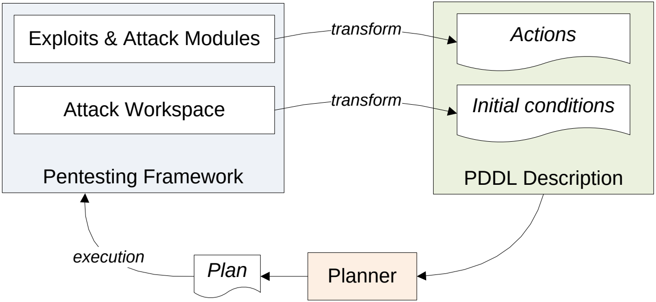
# Agenda

1. Motivation
2. On Exploit Quality Metrics
   - For different stakeholders
   - What can we measure?
3. An Efficient Planning Solution
   - Planning for dummies
   - Two Primitives
   - Using the Primitives in a Network Graph
   - Integrated with a Pentesting Tool
4. Demo
5. Summary

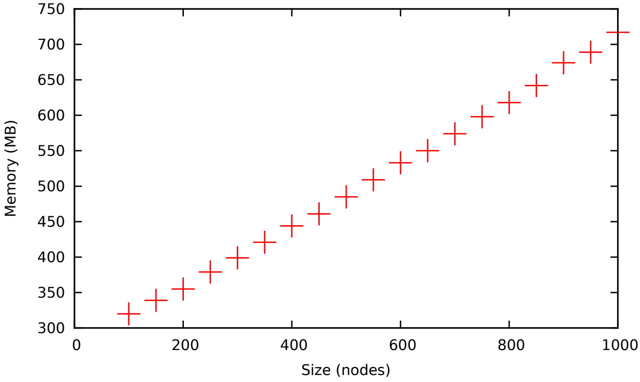
# Anatomy of a Planning-Based Attack

**Attack Planning, as used in Core Impact (and in Core Insight Enterprise):**

[LSR10]; a.k.a. “Cyber Security Domain” [BGHH05]

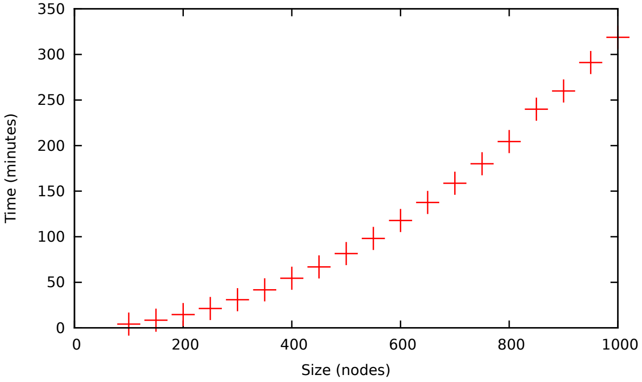


# Experimental results I



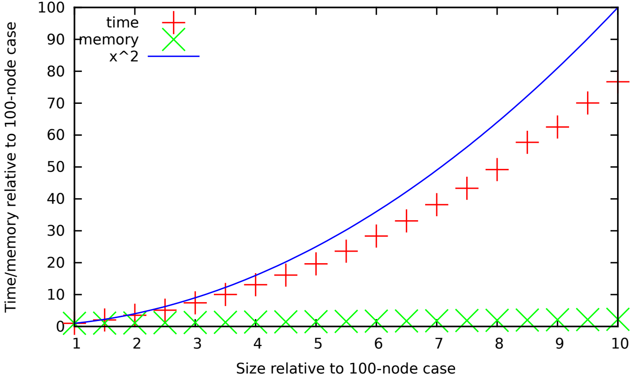

- Memory consumption of the planner (in MB) versus number of machines in the target network.
- Memory consumption is linear.

# Experimental results II




- Planner running time versus number of machines in the target network.
- Planner running time is cuadratic.

# Experimental results III




- Relative plot, clearly showing the cuadratic growth of planning time.
- Scales up to 1000 machines.

# References (for this section)

- [SRL11] An Algorithm to find Optimal Attack Paths in Nondeterministic Scenarios
  - C. Sarraute, G. Richarte, J. Lucangeli
  - *AISec workshop, ACM CCS, Chicago. October 21, 2011.*
- See also [LI05, NJ04, Sch99, Sch00]

# Agenda

1. Motivation
2. On Exploit Quality Metrics
   - For different stakeholders
   - What can we measure?
3. An Efficient Planning Solution
   - Planning for dummies
   - Two Primitives
   - Using the Primitives in a Network Graph
   - Integrated with a Pentesting Tool
4. **Demo**
5. Summary

# Demo time!

# Agenda

# Summary

We have presented:

- An analysis of the factors that affect **exploits quality.**
- An **attack model** based on a selection of factors:
  - Average running time
  - Probability of success
  - Details of the vulnerable platform (OS and application versions)
  - Connectivity requirements.
- An efficient planning solution, **integrated** to a penetration testing framework**.**
- An **evaluation of our implementation** that shows the feasability of planning and verifying attacks in **real-life scenarios**.

# That's all folks!

Thanks for your attention!
Questions?

carlos @ coresecurity . com
http://corelabs.coresecurity.com/

Thanks to Gerardo Richarte, Pedro Varangot and Ariel Waissbein for their ideas and contributions.

# References I


Ivan Arce.
On the quality of exploit code: An evaluation of publicly available exploit code.
In *RSA Security Conference*, San Francisco, CA, 2005.

Mark S. Boddy, Johnathan Gohde, Thomas Haigh, and Steven A. Harp.
Course of action generation for cyber security using classical planning.
In *Proc. of ICAPS'05*, 2005.

Richard Lippmann and Kyle Ingols.
An annotated review of past papers on attack graphs.
Technical Report ESC-TR-2005-054, Lincoln Laboratory, MIT, 2005.

Jorge Lucangeli, Carlos Sarraute, and Gerardo Richarte.
Attack Planning in the Real World.
In *Workshop on Intelligent Security (SecArt 2010)*, 2010.

# References II

S. Noel and S. Jajodia.
Managing attack graph complexity through visual hierarchical aggregation.
In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 109–118. ACM New York, NY, USA, 2004.

Carlos Sarraute.
New algorithms for attack planning.
In *FRHACK Conference, Besançon, France*, 2009.

Carlos Sarraute.
Probabilistic Attack Planning in Network + WebApps Scenarios.
In *H2HC Conference, Sao Paulo, Brazil*, 2009.

Bruce Schneier.
Attack trees.
*Dr. Dobb's journal*, 24(12):21–29, 1999.

# References III


Bruce Schneier.
*Secrets & lies: digital security in a networked world*, chapter 21.
John Wiley & Sons, Inc. New York, NY, USA, 2000.

Carlos Sarraute, Gerardo Richarte, and Jorge Lucangeli.
An algorithm to find optimal attack paths in nondeterministic scenarios.
In *ACM Workshop on Artificial Intelligence and Security (AISec'11), at ACM CCS Conference*, 2011.