Some Research Directions in Automated Pentesting

Carlos Sarraute

CoreLabs & ITBA PhD program
Buenos Aires, Argentina

H2HC – October 29/30, 2011
Agenda outline

1. Motivation

2. The Search for an Efficient Solution
   - Two primitives
   - Using the primitives in a Network Graph
   - Integration with a Pentesting Tool

3. The Search for a Better Model
   - POMDPs
   - Penetration Testing as POMDPs
   - Experiments

4. Discussion
Agenda

1. **Motivation**
2. The Search for an Efficient Solution
   - Two primitives
   - Using the primitives in a Network Graph
   - Integration with a Pentesting Tool
3. The Search for a Better Model
   - POMDPs
   - Penetration Testing as POMDPs
   - Experiments
4. Discussion
What is 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, ...)

- Main tools available:
  - Core Impact (since 2001)
  - Immunity Canvas (since 2002)
  - Metasploit (since 2003)
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”
Anatomy of a real-world attack

internet

Router

Firewall

WEB SERVER

APPLICATION SERVER

DB SERVER

SENSITIVE

WORKSTATION

DMZ

ATTACKER

Carlos Sarraute

Research Directions in Automated Pentesting

6/50
Anatomy of a real-world attack
Anatomy of a real-world attack
Anatomy of a real-world attack
Agenda

1. Motivation

2. The Search for an Efficient Solution
   - Two primitives
   - Using the primitives in a Network Graph
   - Integration with a Pentesting Tool

3. The Search for a Better Model
   - POMDPs
   - Penetration Testing as POMDPs
   - Experiments

4. Discussion
Agenda

1. Motivation
2. The Search for an Efficient Solution
   - Two primitives
   - Using the primitives in a Network Graph
   - Integration with a Pentesting Tool
3. The Search for a Better Model
   - POMDPs
   - Penetration Testing as POMDPs
   - Experiments
4. Discussion
The **Choose** primitive

**Problem**

\{A_1, \ldots, A_n\} independent actions that result in a goal \(g\).
Each \(A_k\) has probability of success \(p_k\) and running time \(t_k\).

**Task:** Find order of execution to minimize total running time.
Problem

\{A_1, \ldots, A_n\} independent actions that result in a goal \( g \).
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 \( g \).
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: $O(n \log n)$
Groups of actions with an AND relation (order is not specified).
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
- [Sar09b] Probabilistic Attack Planning in Network + WebApps Scenarios
Agenda

1. Motivation

2. The Search for an Efficient Solution
   - Two primitives
   - Using the primitives in a Network Graph
   - Integration with a Pentesting Tool

3. The Search for a Better Model
   - POMDPs
   - Penetration Testing as POMDPs
   - Experiments

4. Discussion
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 → use *Combine* primitive.
2. **Asset node**: connected by OR relation with the actions that provide that asset → 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

\[
T[s] = 0, \ P[s] = 1 \\
T[v] = +\infty, \ P[v] = 0 \quad \forall v \in \mathcal{V}, v \neq s \\
S \leftarrow \emptyset \\
Q \leftarrow \mathcal{V} \text{ (where $Q$ is a priority queue)}
\]

while $Q \neq \emptyset$ do

\[
u \leftarrow \arg \min_{x \in Q} \frac{T[x]}{P[x]}
\]

$Q \leftarrow Q \setminus \{u\}, \ S \leftarrow S \cup \{u\}$

for all $v \in \mathcal{V} \setminus S$ 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. The Search for an Efficient Solution
   - Two primitives
   - Using the primitives in a Network Graph
   - Integration with a Pentesting Tool

3. The Search for a Better Model
   - POMDPs
   - Penetration Testing as POMDPs
   - Experiments

4. Discussion
Attack Planning, as used in Core Insight Enterprise

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

- Scales up to 1000 machines.
- Planner running time is quadratic.
- Memory consumption is linear.
[SRL11] An Algorithm to find Optimal Attack Paths in Nondeterministic Scenarios
  - C. Sarraute, G. Richarte, J. Lucangeli
Motivation

The Search for an Efficient Solution
- Two primitives
- Using the primitives in a Network Graph
- Integration with a Pentesting Tool

The Search for a Better Model
- POMDPs
- Penetration Testing as POMDPs
- Experiments

Discussion
Anatomy of a real-world attack w/o uncertainty

What’s the problem?
Anatomy of a real-world attack w/o uncertainty

What’s the problem?

PDDL & Planner w/o Uncertainty!
Penetration Testing w/ uncertainty

What kind of uncertainty?

Penetration testing has insider knowledge. But can’t know *everything!* OS versions, applications installed, ...
Penetration Testing w/ uncertainty

What kind of uncertainty?

Penetration testing has insider knowledge. But can’t know *everything*! OS versions, applications installed, …

- **Classical solution:**
  1. gather information (run scans);
  2. attack (run exploits)

  - Still simplified: scans don’t yield perfect knowledge
  - Exhaustive scans expensive (runtime, traffic)
Penetration Testing w/ uncertainty

What kind of uncertainty?

Penetration testing has insider knowledge. But can’t know *everything!* OS versions, applications installed, …

**Classical solution:**
(I) gather information (run scans); (II) attack (run exploits)

- Still simplified: scans don’t yield perfect knowledge
- Exhaustive scans expensive (runtime, traffic)

**Our solution:** explicit model of uncertainty in POMDP

- POMDP plans intelligently mix (I) and (II)
- Grounds attack planning w/ uncertainty in formal framework
- Only related work: neither of these [SRL11]
Penetration Testing w/ uncertainty

What kind of uncertainty?
Penetration testing has insider knowledge. But can’t know *everything!* OS versions, applications installed, . . .

- **Classical solution:**
  (I) gather information (run scans); (II) attack (run exploits)
  - Still simplified: scans don’t yield perfect knowledge
  - Exhaustive scans expensive (runtime, traffic)

- **Our solution:** explicit model of uncertainty in POMDP
  - POMDP plans intelligently mix (I) and (II)
  - Grounds attack planning w/ uncertainty in formal framework
  - Only related work: neither of these [SRL11]
  - And, yes, it doesn’t scale . . . (to be continued)
Agenda

1. Motivation

2. The Search for an Efficient Solution
   - Two primitives
   - Using the primitives in a Network Graph
   - Integration with a Pentesting Tool

3. The Search for a Better Model
   - POMDPs
   - Penetration Testing as POMDPs
   - Experiments

4. Discussion
Markov Decision Process (MDP)

Definition

An MDP is a tuple \( \langle S, A, T, r \rangle \) where:

- \( S \) is the state space
- \( A \) is the action space
- \( T : S \times A \times S \to [0, 1] \) is the transition function
  - \( T(s, a, s') \) is the probability of coming to state \( s' \) when executing action \( a \) in state \( s \)
- \( r : S \times A \to \mathbb{R} \) is the reward function

Solution: policy \( \pi : S \to A \)

Objective: maximize expected reward \( E \left[ \sum_{t=0}^{\infty} r_t \bigg| \pi \right] \)
Partially Observable MDP (POMDP)

Definition

A POMDP is a tuple \( \langle S, A, T, r, O, O, b_0 \rangle \) where:

- \( \langle S, A, T, r \rangle \) is a Markov decision process
- \( O \) is the space of observations
- \( O : S \times A \times O \rightarrow [0, 1] \) is the observation function
  - \( O(s, a, o) \) is the probability of making observation \( o \) when executing action \( a \) in state \( s \)
- \( b_0 \) is the initial belief (probability distribution over \( S \))
POMDP Policies

Definition

Solution: policy $\pi : \mathcal{H} \rightarrow \mathcal{A}$ ($\mathcal{H}$: action/observation histories)

Objective: maximize expected reward $E \left[ \sum_{t=0}^{\infty} r_t \mid b_0, \pi \right]$

Equivalent: policy $\pi : \mathcal{B} \rightarrow \mathcal{A}$ where $\mathcal{B} = \Pi(S)$
Solving POMDPs

**Is it hard?**
- $S$: all states (= all possible configurations)
- Belief states $b$: probability distributions over $S$
- ... and we need to *reason* about this stuff!

**How to do it?**
- Here: SARSOP [KHL08]
- Approximate belief value based on selected belief states
  (get hyperplane for each, compute upper envelope)

**What about scaling??**
- Bad
- Long-term proposal: use in “1-machine case”, design
  global solution by decomposition + approximation
Motivation
The Search for an Efficient Solution
- Two primitives
- Using the primitives in a Network Graph
- Integration with a Pentesting Tool

The Search for a Better Model
- POMDPs
- Penetration Testing as POMDPs
- Experiments

Discussion
# Birds-Eye View

## States
- Network structure static and fully known
- Combinations of configuration parameters . . .
- . . . as relevant to modeled exploits!

## Actions
- Exploits: succeed/fail depending on state
- Scans: return observation depending on state
- Both are deterministic!

## Rewards
- \( r = V - T - D \): value of computer, runtime, detection risk
- \( V \): human decision; \( T, D \): estimate using statistics

## Initial belief
- Probability distribution over configurations
  \[\Rightarrow\] uncertainty from point of view of pentesting tool
Example: Actions

actions:

Probe-M0-p445
OSDetect-M0

Exploit-M0-win2000-SMB
Exploit-M0-win2003-SMB
Exploit-M0-winXPsp2-SMB

Terminate

"Terminate" action: give planner the choice to "give up" if expected costs outweigh expected reward
Example: States (1 Machine)

states:

M0-win2000
M0-win2000-p445
M0-win2000-p445-SMB
M0-win2000-p445-SMB-vuln
M0-win2000-p445-SMB-agent

M0-win2003
M0-win2003-p445
M0-win2003-p445-SMB
M0-win2003-p445-SMB-vuln
M0-win2003-p445-SMB-agent

M0-winXPsp2
M0-winXPsp2-p445
M0-winXPsp2-p445-SMB
M0-winXPsp2-p445-SMB-vuln
M0-winXPsp2-p445-SMB-agent

M0-winXPsp3
M0-winXPsp3-p445
M0-winXPsp3-p445-SMB

terminal
Example: Scans – Port Scan

...
O: Probe-M0-p445: M0-win2003 : closed-port 1
...
O: Probe-M0-p445: M0-winXPsp2 : closed-port 1
O: Probe-M0-p445: M0-winXPsp2-p445 : open-port 1
O: Probe-M0-p445: M0-winXPsp2-p445-SMB : open-port 1
...
O: Probe-M0-p445: M0-winXPsp3 : closed-port 1
O: Probe-M0-p445: M0-winXPsp3-p445 : open-port 1
O: Probe-M0-p445: M0-winXPsp3-p445-SMB : open-port 1
Example: Scans – OS Detection

O: OSDetect-M0: M0-win2000 : win 1
O: OSDetect-M0: M0-win2000-p445 : win 1
...
O: OSDetect-M0: M0-win2003 : win 1
O: OSDetect-M0: M0-win2003-p445 : win 1
...
O: OSDetect-M0: M0-winXPsp2 : winxp 1
O: OSDetect-M0: M0-winXPsp2-p445 : winxp 1
...
O: OSDetect-M0: M0-winXPsp3 : winxp 1
O: OSDetect-M0: M0-winXPsp3-p445 : winxp 1
...
Example: Exploit SAMBA Server on Port 445

T: Exploit-M0-win2003-SMB identity
    * 0
    M0-win2003-p445-SMB-agent 1

O: Exploit-M0-win2003-SMB: * : * 0
O: Exploit-M0-win2003-SMB: * : no-agent 1
    agent-installed 1
What is our “Initial Belief”??

- **Regular penetration testing**
  - Run tests every \( T \) time units (days)
  - Possibly changed OS, applications (versions), . . .
    \[ \rightarrow \text{Uncertainty in } b_0, \text{ function of } T \]

- **How to derive } b_0(T)?**
  - In general: *formal model of system evolution* . . .
  - Here: (a) individual updates; (b) perfect knowledge at \( T = 0 \)

```
\begin{tikzpicture}[->,>=stealth,shorten >=1pt,auto,node distance=1.5cm,thick]
  
  \node (v1) [shape=circle] {1};
  \node (v2) [shape=circle, right of=v1, xshift=2cm] {2};
  \node (v3) [shape=circle, right of=v2, xshift=2cm] {3};
  \node (v4) [shape=circle, right of=v3, xshift=2cm] {4};
  \node (v5) [shape=circle, right of=v4, xshift=2cm] {5};

  \path
    (v1) edge node [above] {\( p_{1,2} \)} (v2)
    (v2) edge node [above] {\( p_{2,3} \)} (v3)
    (v3) edge node [above] {\( p_{3,4} \)} (v4)
    (v4) edge node [above] {\( p_{4,5} \)} (v5)
    (v1) edge node [above] {\( p_{1,1} \)} (v2)
    (v2) edge node [above] {\( p_{2,2} \)} (v3)
    (v3) edge node [above] {\( p_{3,3} \)} (v4)
    (v4) edge node [above] {\( p_{4,4} \)} (v5)
    (v5) edge node [above] {\( p_{5,5} \)} (v1);

  \end{tikzpicture}
```

“each day: either no change, or upgrade, or upgrade to latest version”
Motivation

The Search for an Efficient Solution

- Two primitives
- Using the primitives in a Network Graph
- Integration with a Pentesting Tool

The Search for a Better Model

- POMDPs
- Penetration Testing as POMDPs
- Experiments

Discussion
Test Examples

Problem generator with 3 parameters:

- **Number** $M$ of machines in network
  Agent on machine $M_0$, $M$ “behind” $M_0$ in fully connected network

- **Number** $E$ of exploits considered
  $E \geq M$, distributed evenly across machines

- **Time delay** $T$ (days) since last pentest
  Update parameters estimated by hand

Here: $1 \leq M \leq 7; 1 \leq E \leq 50; 0 \leq T \leq 200$
Scaling $T$

Scaling $T$ against $M$

Scaling $T$ against $E$
Scaling $E$ and $M$

Scaling $E$ against $M$; $T = 10$

Scaling $E$ against $M$; $T = 80$
Joint work with researchers at INRIA (Nancy, France)

Jörg Hoffmann, author of FF [Hof01] and Metric-FF [Hof02], reference tools for “classical” planning.

Olivier Buffet, author of books and tools on Markov decision process [SB10].

- [SBH11] Penetration Testing == POMDP Solving?
## Agenda

1. **Motivation**

2. **The Search for an Efficient Solution**
   - Two primitives
   - Using the primitives in a Network Graph
   - Integration with a Pentesting Tool

3. **The Search for a Better Model**
   - POMDPs
   - Penetration Testing as POMDPs
   - Experiments

4. **Discussion**
Probabilistic Planner: Summary

First direction . . . We have presented:

- An **attack model** based on exploits metrics:
  - 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 feasibility of planning and verifying attacks in **real-life scenarios**.
POMDP model: Is it worth it?

Second direction . . . POMDPs make better hackers!

(a) Beliefs: likelihood of particular vulnerabilities
   \[\Rightarrow\text{ order exploits by promise}\]

(b) Belief transitions: update “promise” as more information comes in
   \[\Rightarrow\text{ order exploits dynamically}\]

(c) Belief transitions vs. rewards (time/risk): trade-off
    observation gain against its cost
   \[\Rightarrow\text{ apply scans only where needed/profitable}\]
POMDP model: What have we gained?

- More accurate model of attack planning w/ uncertainty
- Scales “Ok” in 1-target-machine case
- Can deliver better plans thus more effective pentesting
  - Policy = stronger notion of plan
  - Contemplates all possible histories of actions / observations.
- No independence assumptions
  - Understand the limits of what can be done with state-of-the-art POMDP planners
Bridging the language gap

- Separate the problem from potential solutions.
- Communicate our problem to the AI / Planning community → they’re looking for practical applications!

**Solving:** PoC implementation shows feasibility  
Scaling to large networks → decompose/approximate with 1-target-machine cases

**Basic AI:** these POMDPs have particular properties . . . → open path for further research
That’s all folks!

Thanks for your attention!
Questions?

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

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.

Jörg Hoffmann.
FF: The fast-forward planning system.

Jörg Hoffmann.
Extending FF to numerical state variables.

H. Kurniawati, D. Hsu, and W. Lee.
SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces.
In *RSS IV*, 2008.
In *Workshop on Intelligent Security (SecArt 2010)*, 2010.

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

In *H2HC Conference, Sao Paulo, Brazil*, 2009.

O. Sigaud and O. Buffet, editors. *Markov Decision Processes and Artificial Intelligence*.
References III

Carlos Sarraute, Olivier Buffet, and Jörg Hoffmann.
Penetration testing == POMDP planning?
In *Proceedings of the 3rd Workshop on Intelligent Security (SecArt’11), at IJCAI*, 2011.

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.