Using AI Techniques to improve Pentesting Automation

Carlos Sarraute

Core Security Technologies

and *Ph.D. program in Informatics Engineering, ITBA*

Hackito Ergo Sum – April 8-10, 2010

Brief presentation

- My company: Core Security Technologies
  - Boston (USA)
    - marketing and sales
  - Buenos Aires (Argentina)
    - research and development
- About me:
  - M.Sc. in Mathematics from UBA
  - I have worked in CoreLabs since 2000
  - coordinate research activities (e.g. Bugweek) and publication of advisories
  - one of my focus areas: applying Artificial Intelligence techniques to solve problems from the security field

Outline

- Motivation
- The Attack Model
- Pentest tool / Planner integration
- Performance and Scalability evaluation
- Conclusion / Future work

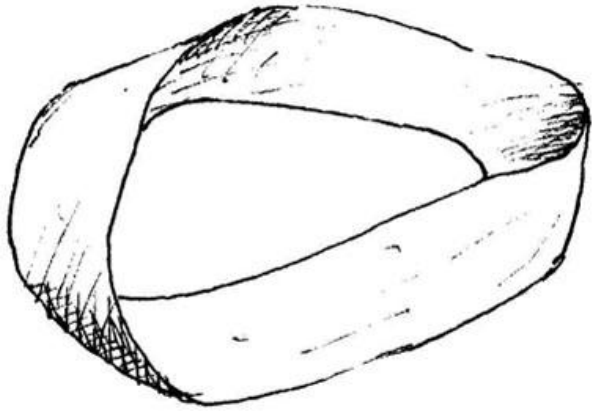
Motivation

Penetration testing frameworks

- Facilitate the work of network penetration testers
- Make the assessment of network security more accessible
- Main tools available:
  - Core Impact (since 2001)
  - Immunity Canvas (since 2002)
  - Metasploit (since 2003)
    - open source, owned by Rapid7 since 2009
- Difference with vuln scanners (Nessus, Retina, …)
  - launch real exploits for vulnerabilities

The evolution of pentesting

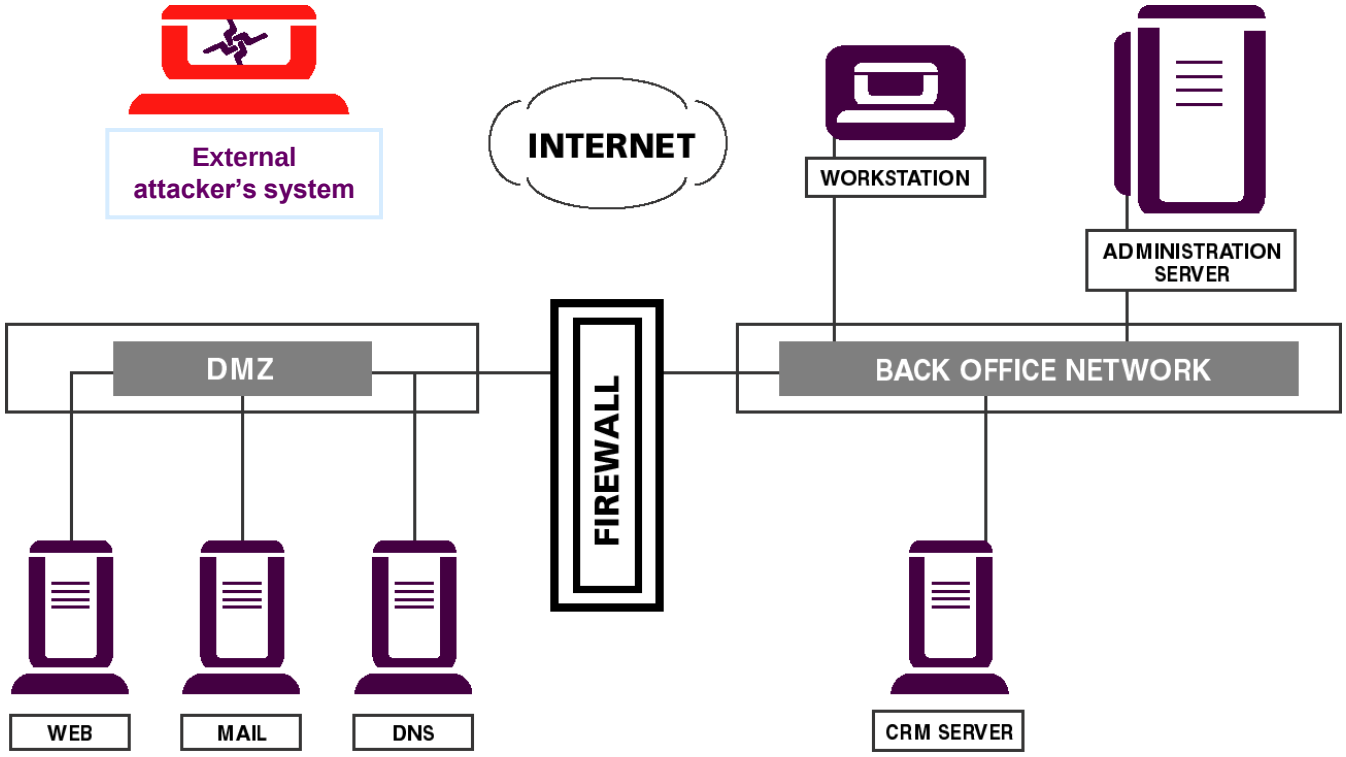
- Pentesting tools have become more complex
  - shipping more exploits
  - shipping more information gathering modules
- Cover new attack vectors
  - Client-side
    - The perimeter is inside-out!
  - WiFi
  - WebApps



- Organizations are evolving
  - technological and infrastructure complexity

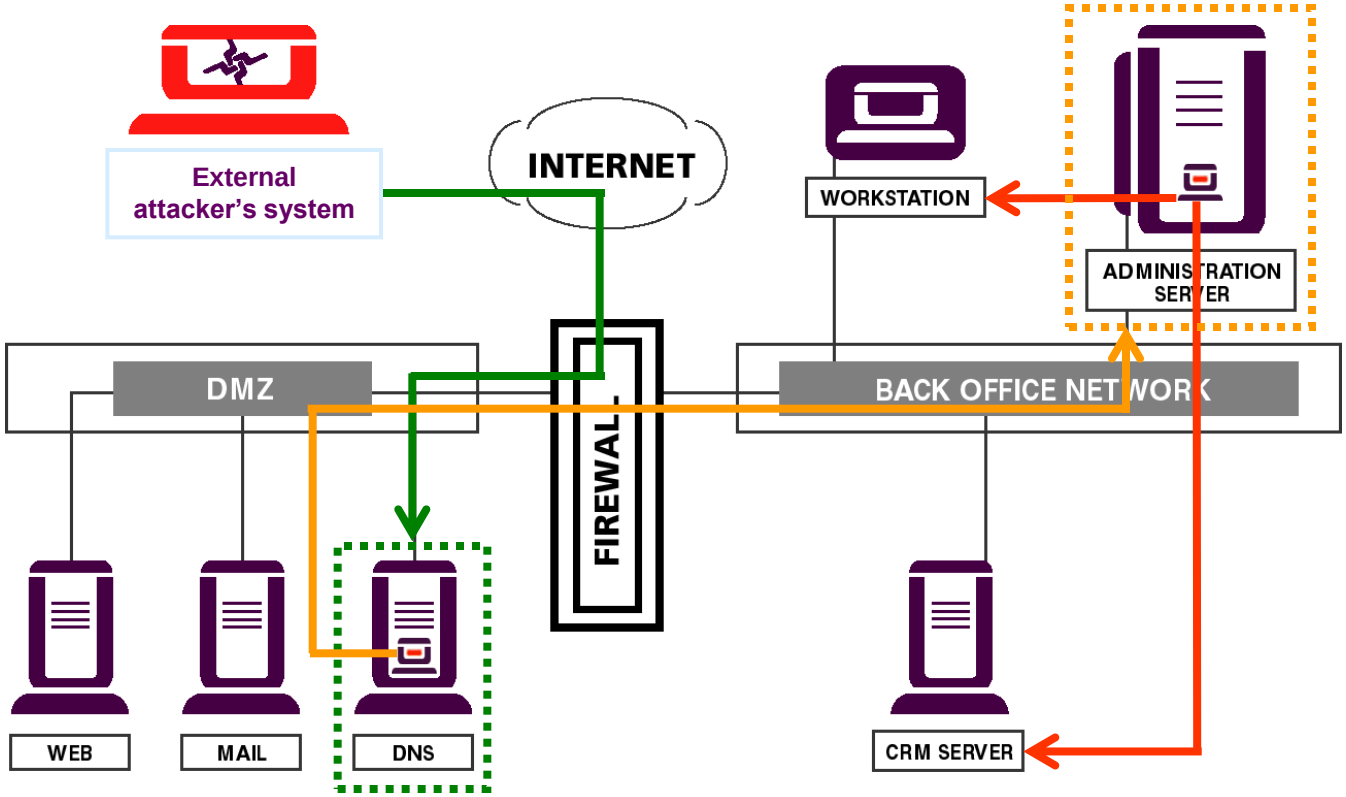
→ Need for automation!

Sample pentest scenario



Anatomy of a real-world attack

A sophisticated real-world attacker will leverage trust relationships to gain access to more valuable information assets



Base camp

A target server is attacked and compromised

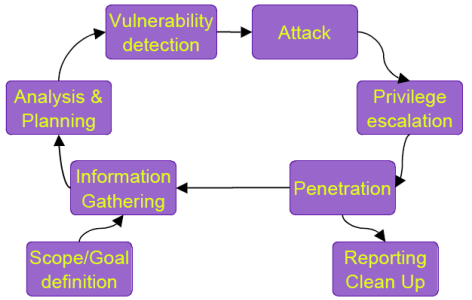
The acquired server is used as vantage point to penetrate the corporate net

Further attacks are performed as an internal user

The Attack Model

Construction of an Attack Model

- 2003: "Building computer network attacks"
  - model = abstraction of the pentest practice
  - technical report with the details of the model



- 2003: "Modern intrusion practices"
  - presentation at BlackHat by Gera Richarte

- Roadmap for the work on attack planning.

The model components

- Goals
  - Objectives of the attack
    - Obtain credit card numbers from the Database server
- Assets
  - Anything an attacker may need during the attack
    - OperatingSystemAsset, TCPConnectivityAsset and AgentAsset
- Actions
  - Atomic step that can be part of an attack
    - An exploit, a TCP connection and an OS identification method
- Agents: actors who can perform actions

Attack Graph nodes

- The graph nodes are Actions and Assets

- Every action has an associated result
  - an Exploit gives as result an Agent on the target machine
- Actions have requirements (preconditions or subgoals)
  - Exploits are platform dependent and require knowledge of the Operating System of the target before execution
  - an HTTP Exploit requires an open port (and connectivity)

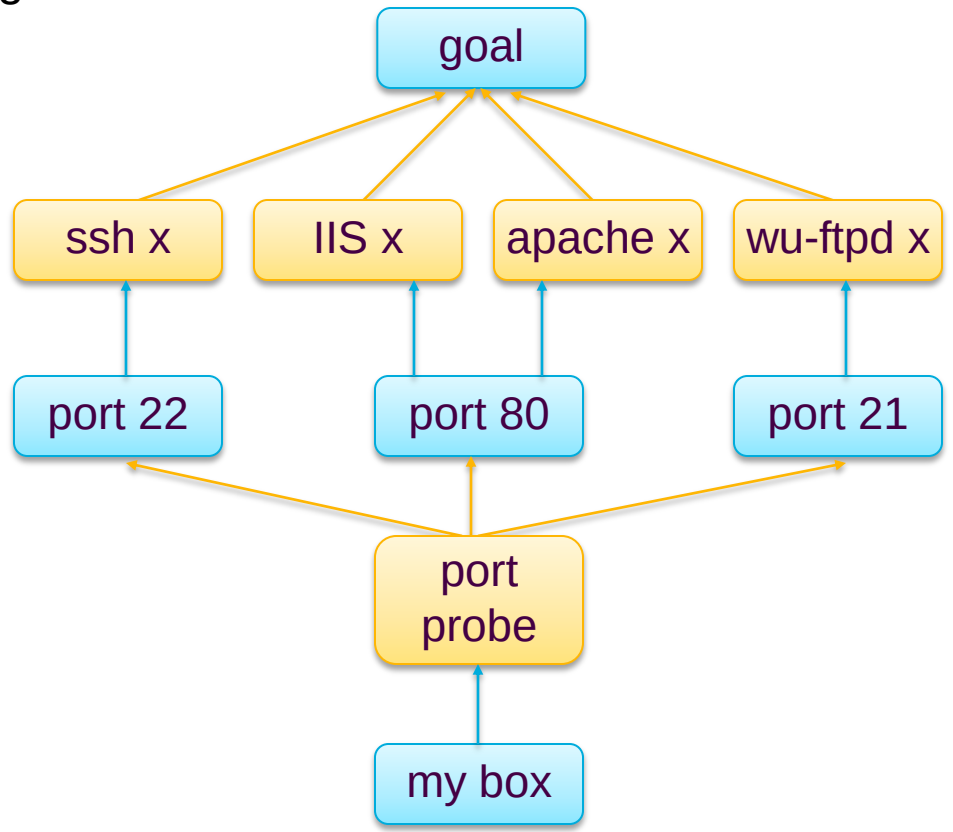
Very small example of attack planning

Goal: To gain control of any host in target network

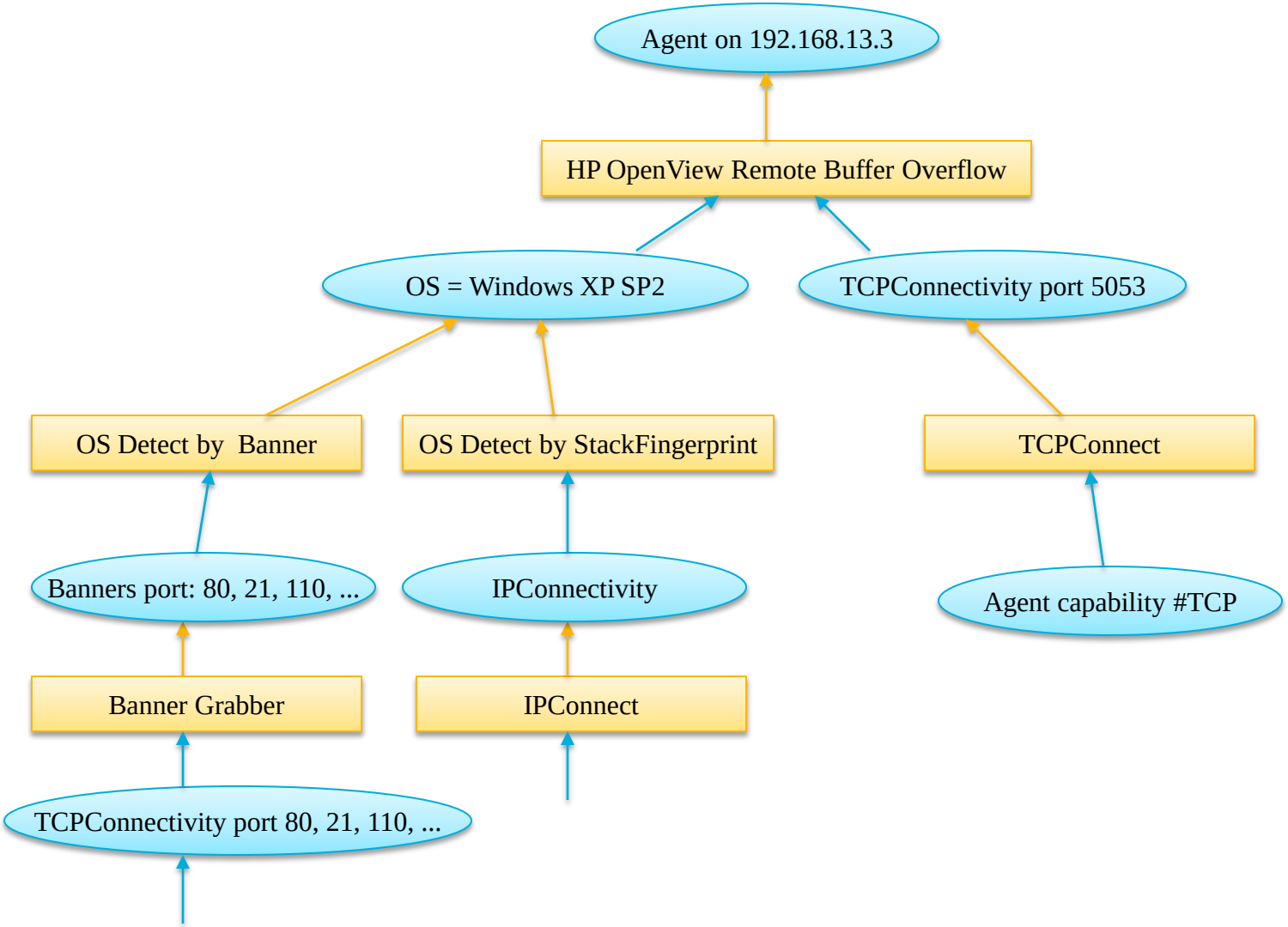
Assets: Target's IP address
Control of my box
A set of IG tools and exploits

Actions:
test if a given port is open (port probe)
exploit ssh (on an OpenBSD)
exploit wu-ftpd (on a Linux)
exploit IIS (on a Windows)
exploit apache (on a Linux)

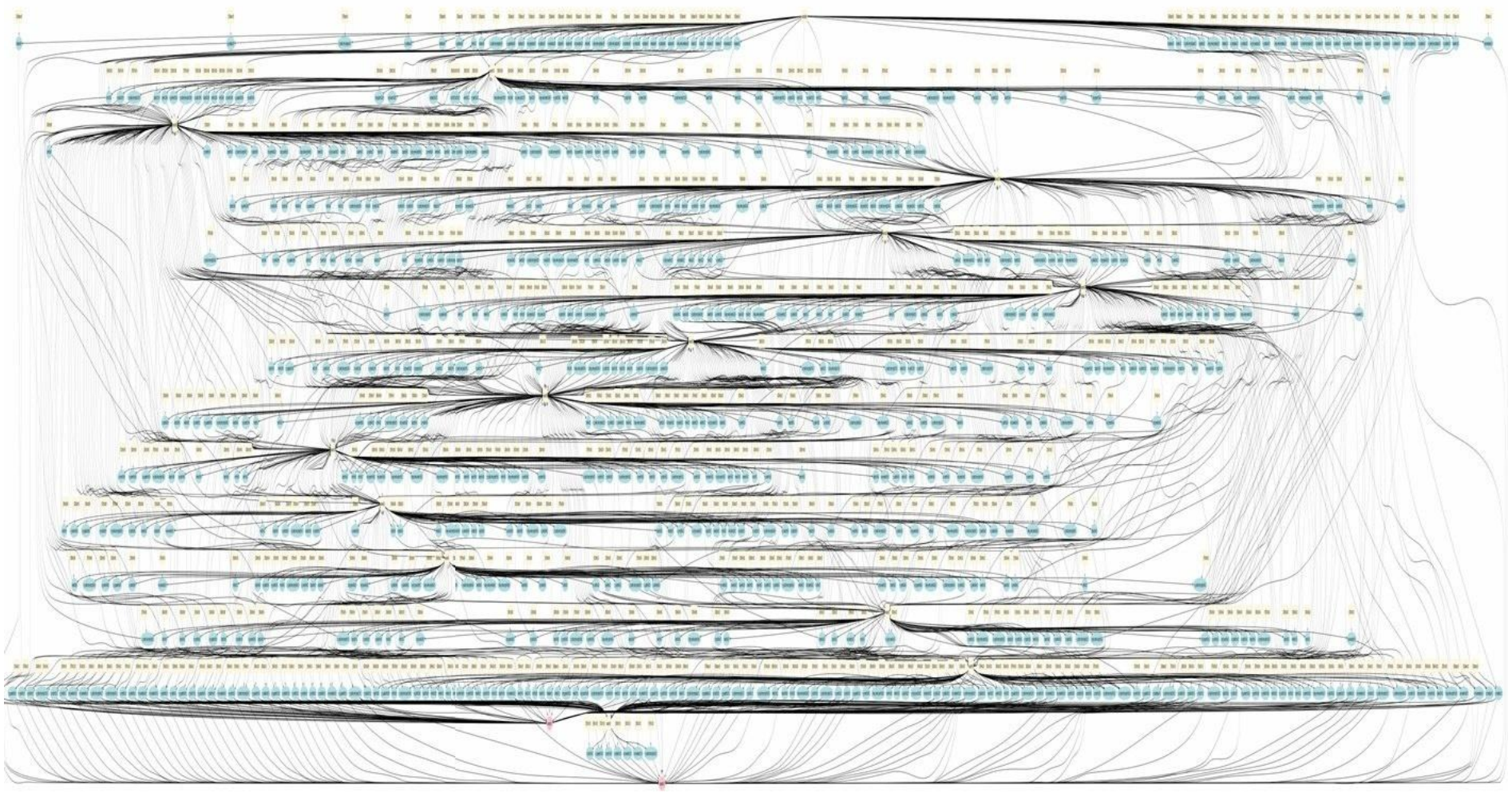
Plan:
Probe only ports 22, 80 and 21.
Probe port 80 first!
As soon as a port is found open, run an exploit.
Keep probing other ports only if exploit fails.



Alternated layers of actions and assets



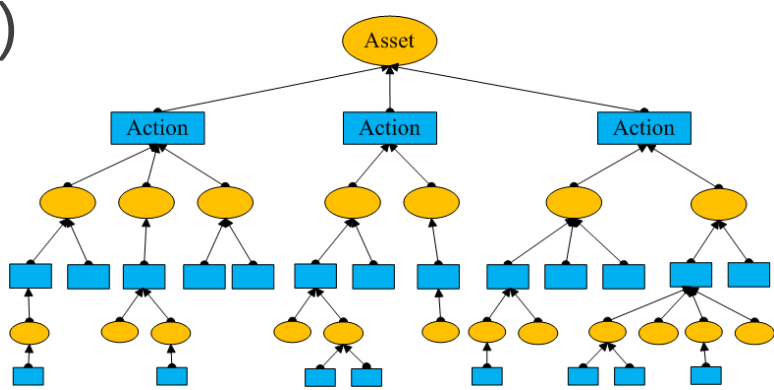
An Attack Graph, a bit more real



From Noel – Jajodia: "Managing Attack Graph Complexity Through Visual Hierarchical Aggregation"

How did the story continue?

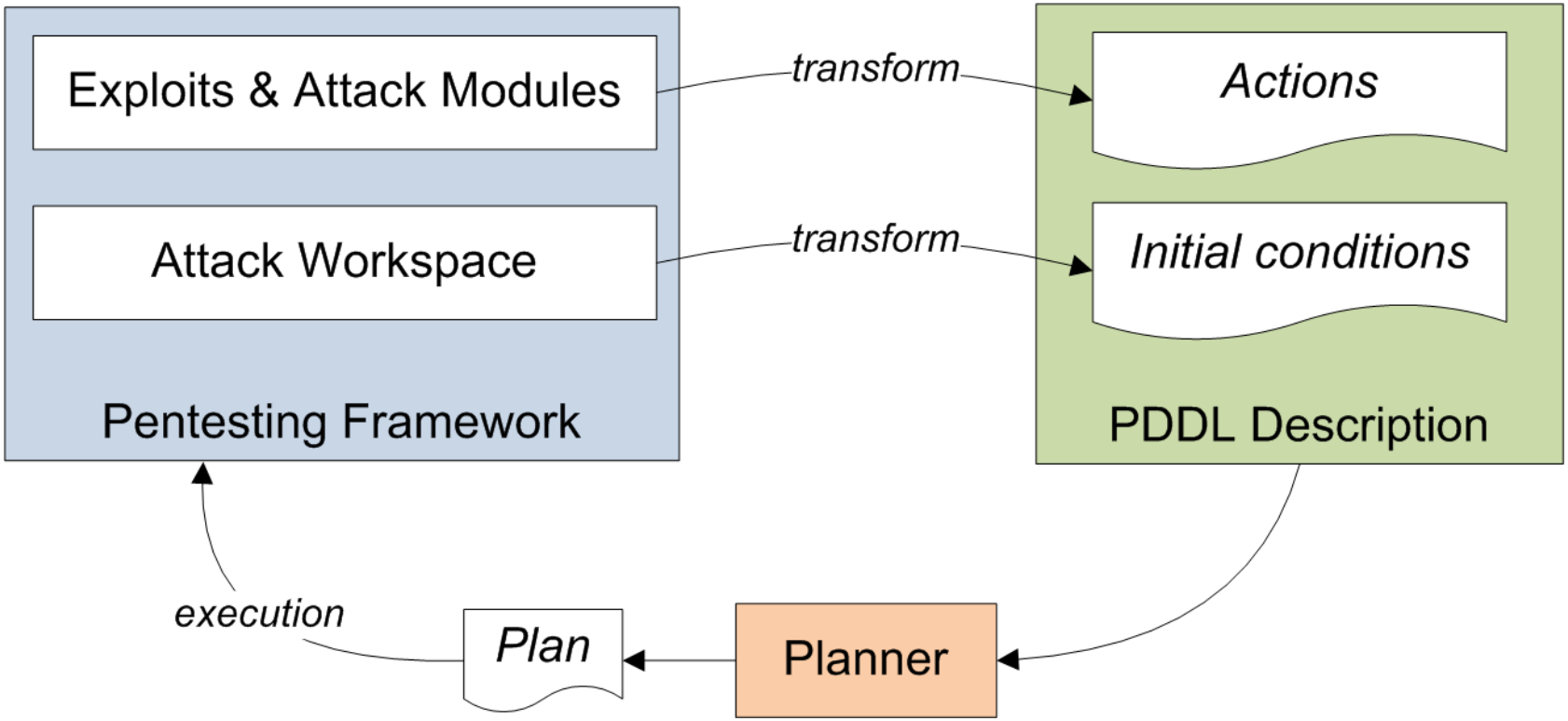
- 2008: PacSec presentation (with A.Weil)
  - Practical approach for automation
  - Ships with pentest tool (Impact)
  - Limitations:
    - no pivoting
    - no optimizations



- 2009: FRHACK presentation
  - Algorithm for probabilistic planning
  - More theoretical (demonstrations of the algorithm)
  - Research prototype
- Today: Planner integrated with our pentest tool

Pentest tool / Planner integration

Architecture of our solution



What is PDDL ?

- PDDL = Planning Domain Definition Language

- Language designed for the International Planning Competition
  - Lots of algorithms understand PDDL
  - Use the winning algorithms to generate plans
  - Compare different planners in our particular domain

Types of objects

- Objects have types
  - Helps to reduce the complexity of the search
  - Use less memory

network	operating_system
host	OS_version
port	OS_edition
port_set	OS_build
application	OS_servicepack
agent	OS_distro
privileges	kernel_version

Predicates - for connectivity

- Assets are translated as predicates.
- Examples:
  - (connected_to_network *?s - host ?n - network*)
  - (IP_connectivity *?s - host ?t - host*)
  - (TCP_connectivity *?s - host ?t - host ?p - port*)
  - (TCP_listen_port *?h - host ?p - port*)
  - (UDP_listen_port *?h - host ?p - port*)
- Maximal arity = 3

Predicates - for the Operating System info

- Lots of predicates for the OS information
  - we need detailed info to evaluate the reliability of the exploits
- Examples:
  - (has_OS *?h* - *host ?os* - *operating_system*)
  - (has_OS_version *?h* - *host ?osv* - *OS_version*)
  - (has_OS_edition *?h* - *host ?ose* - *OS_edition*)
  - (has_OS_build *?h* - *host ?osb* - *OS_build*)
  - (has_OS_servicepack *?h* - *host ?ossp* - *OS_servicepack*)
  - (has_architecture *?h* - *host ?a* - *OS_architecture*)

Model-related action (1)

```
(:action IP_connect
:parameters (?s - host ?t - host)
:precondition (
    and (compromised ?s)
    (exists (?n - network)
        (and (connected_to_network ?s ?n)
            (connected_to_network ?t ?n))))
:effect
    (IP_connectivity ?s ?t)
)
```

→ Note the "exists"

Model-related action (2)

```
(:action TCP_connect
:parameters (?s - host ?t - host ?p - port)
:precondition (
    and (compromised ?s)
        (IP_connectivity ?s ?t)
        (TCP_listen_port ?t ?p))
:effect
    (TCP_connectivity ?s ?t ?p)
)
```

Sample Exploit (1)

```
(:action HP_OpenView_Remote_Buffer_Overflow_Exploit
:parameters (?s - host ?t - host)
:precondition (and (compromised ?s)
    (and (has_OS ?t Windows)
         (has_OS_edition ?t Professional)
         (has_OS_servicepack ?t Sp2)
         (has_OS_version ?t WinXp)
         (has_architecture ?t I386))
    (has_service ?t ovtrcd)
    (TCP_connectivity ?s ?t port5053)
)
:effect (and (installed_agent ?t high_privileges)
    (increase (time) 4)
))
```

Sample Exploit (2)

```
(:action HP_OpenView_Remote_Buffer_Overflow_Exploit
:parameters (?s - host ?t - host)
:precondition (and (compromised ?s)
    (and (has_OS ?t Solaris)
         (has_OS_version ?t V_10)
         (has_architecture ?t Sun4U))
    (has_service ?t ovtrcd)
    (TCP_connectivity ?s ?t port5053)
)
:effect (and (installed_agent ?t high_privileges)
    (increase (time) 6)
))
```

Measuring execution times

- Measure results of exploit executions in testing lab
  - 748 virtual machines in Core's testing lab
  - different OS and installed applications
  - all the exploits are executed every night

- Get feedback from the users
  - anonymized feedback program in Impact
    - sensitive data is filtered out before sending it
  - natural option for Metasploit (in my opinion)

Planners

- FF = Fast-Forward (Hoffmann 2000)
  - winner of the planning competition in 2000
- Metric-FF (Hoffmann 2002)
  - actions can have numerical effects
  - winner of the competition in 2002
  - still used as a baseline in the planning competitions
  - we tweaked it to use less memory
- SGPlan (Chen, Hsu et al. 2006)
  - based on Metric-FF
  - divides the main problem in subproblems

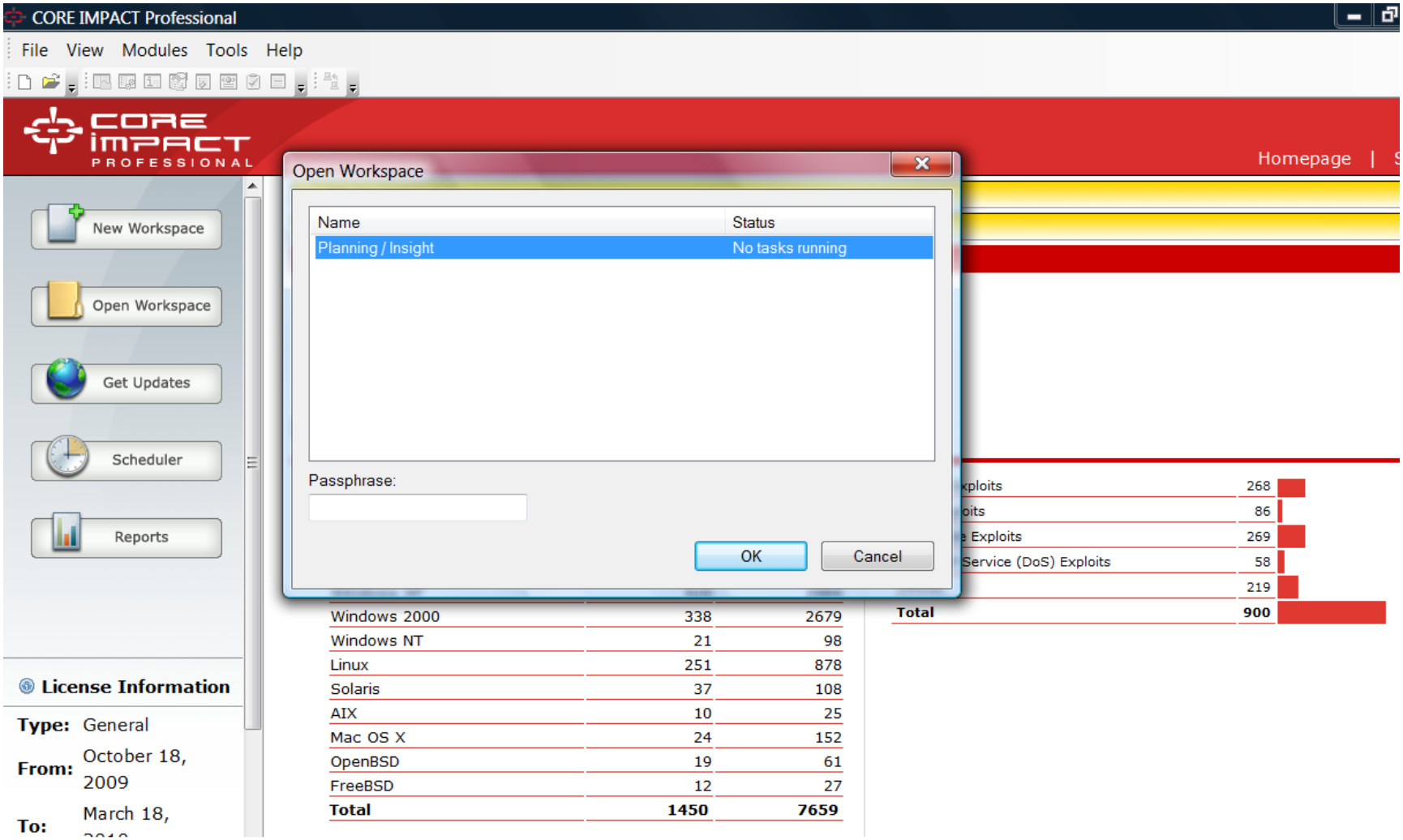
Planner's search heuristics

- 2005: "An annotated review of past papers on attack graphs" (Lippmann and Ingols)
  - The main limitation of previous work on Attack Graphs is scalability

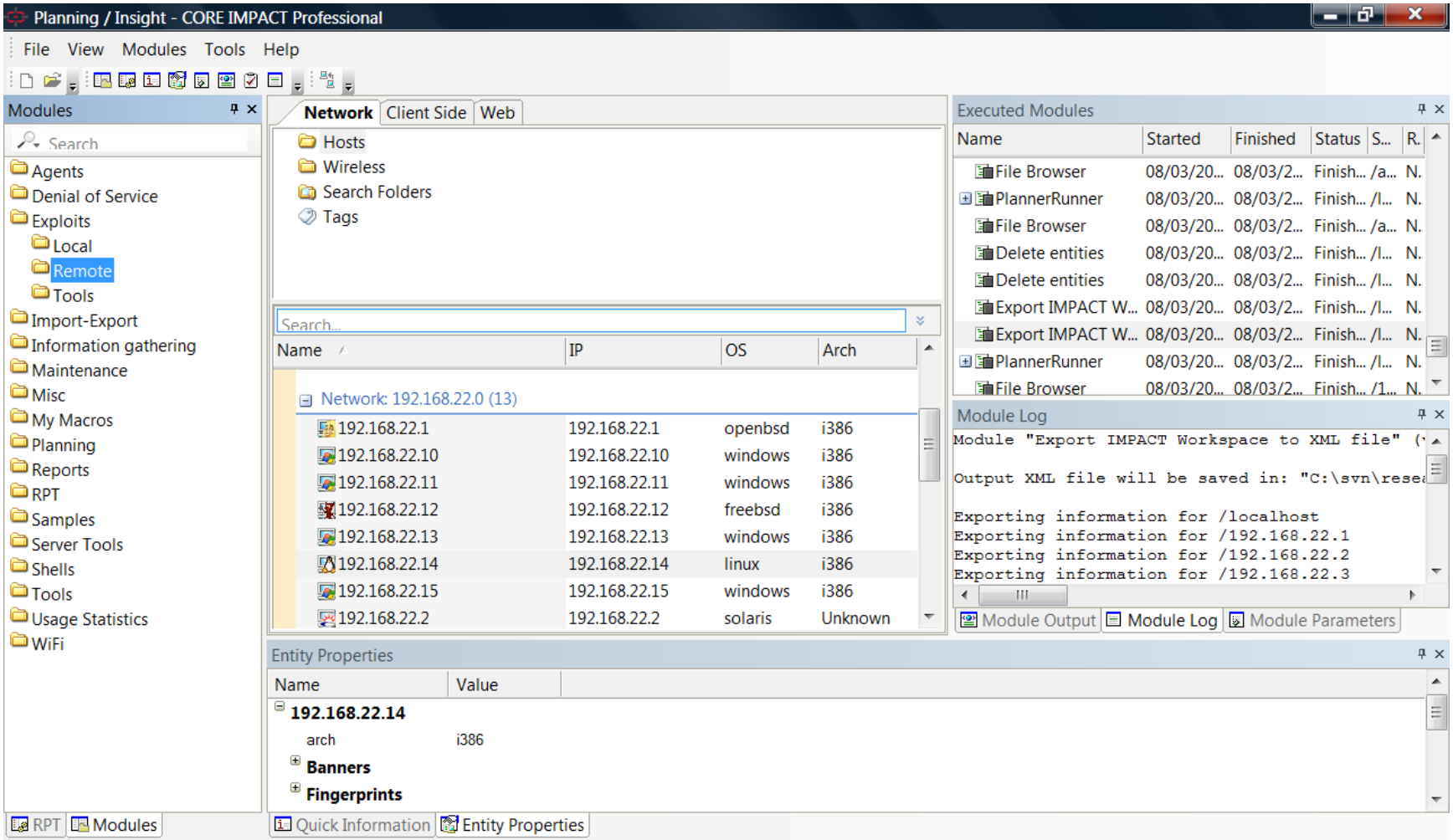
- Don't construct the complete graph!
  - Use an heuristic to explore the state space
  - There are several variations of **A* search** to find attack paths
  - Heuristics: solve a relaxed version of the problem

Small Demo

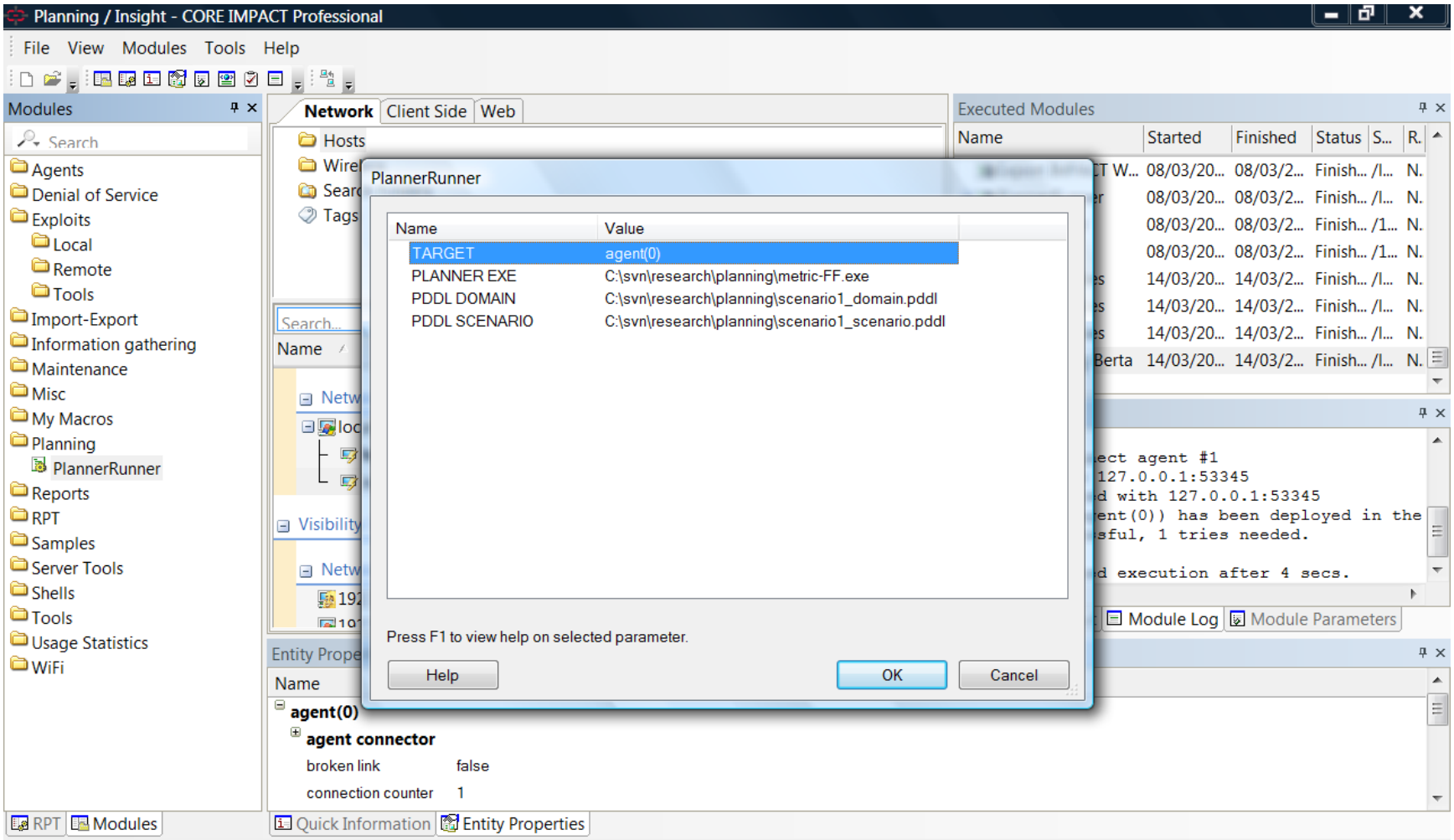
Open workspace for Planning scenario



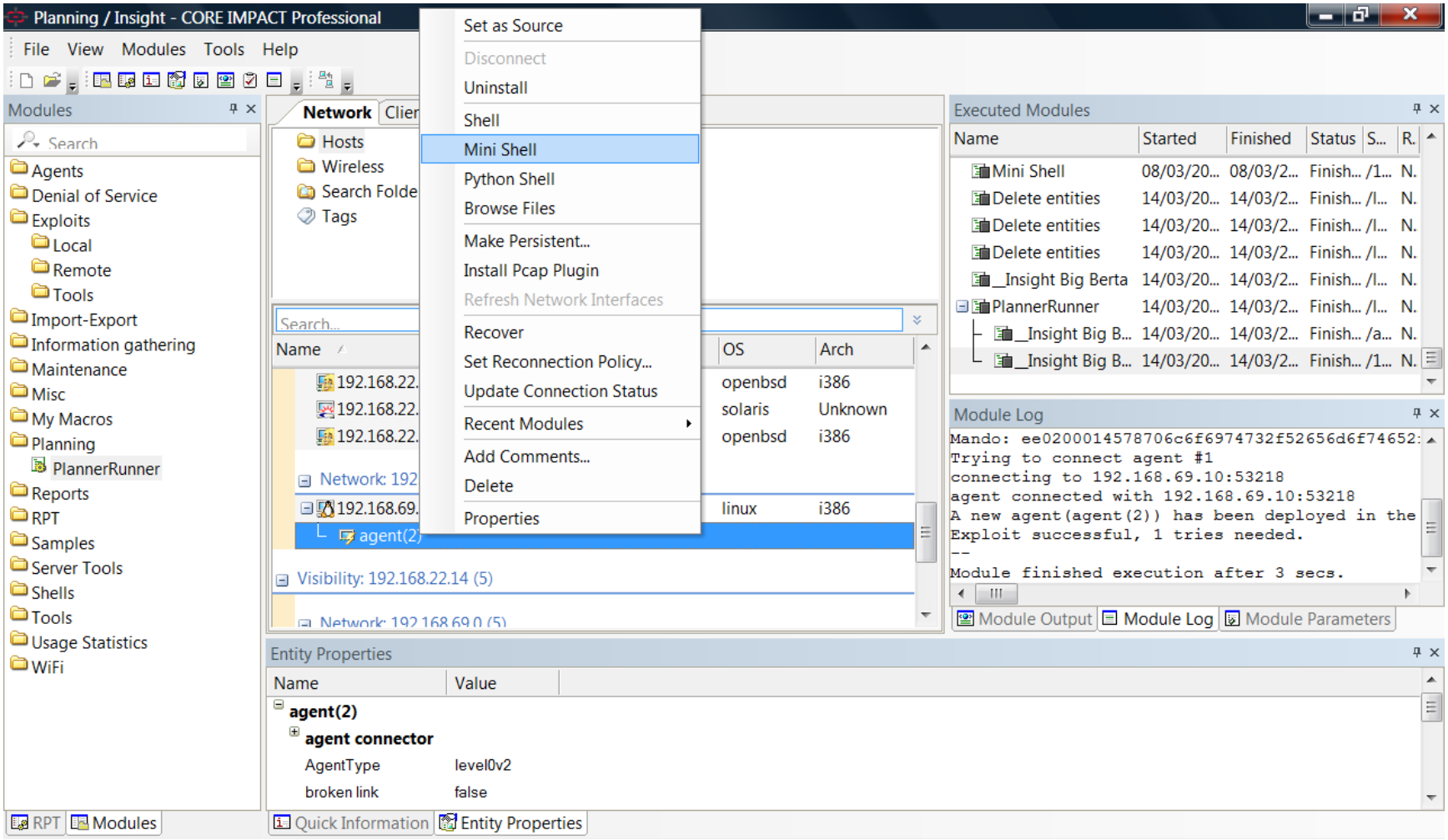
Network discovery of the target network



Launch PlannerRunner



Two agents installed after plan execution

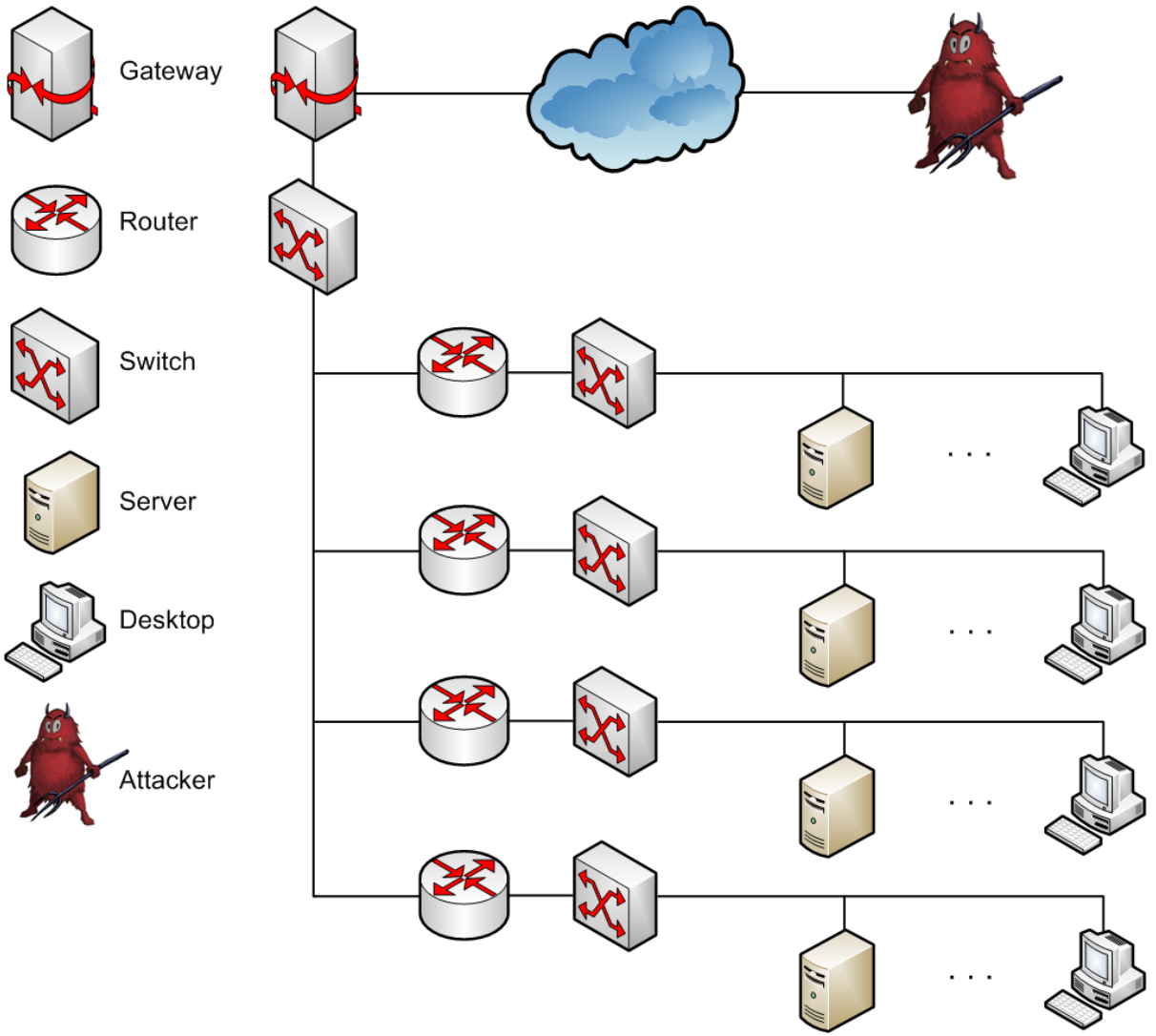


Performance and Scalability evaluation

Testing scenarios

- Evaluate the performance of the planners in terms of memory consumption and running time, in scenarios with:
  - increasing number of machines
  - increasing number of pivoting steps
- Generated scenarios by varying these metrics
  - up to 300 machines
  - up to 20 pivoting steps

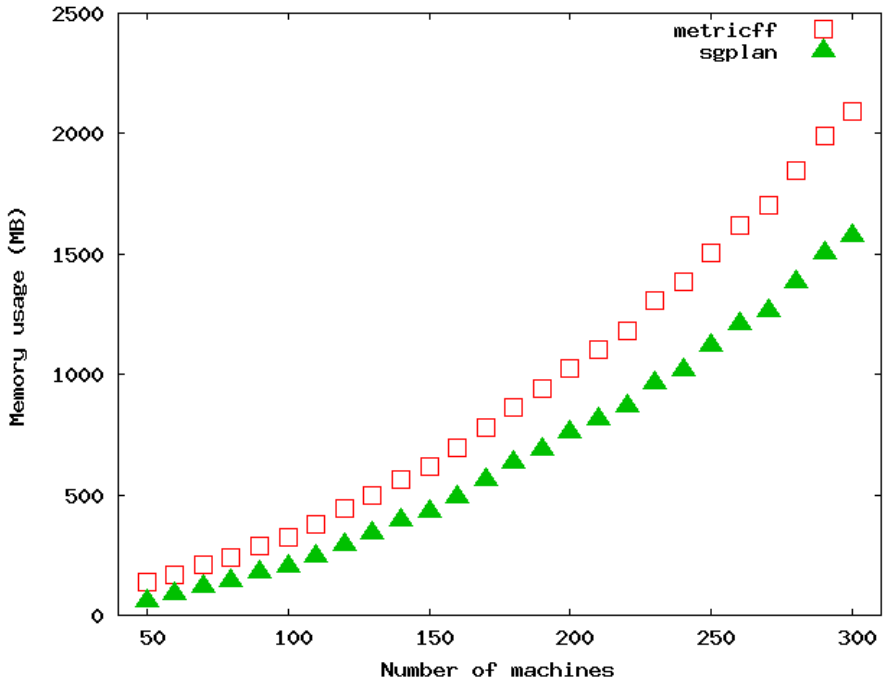
Test network for the evaluation

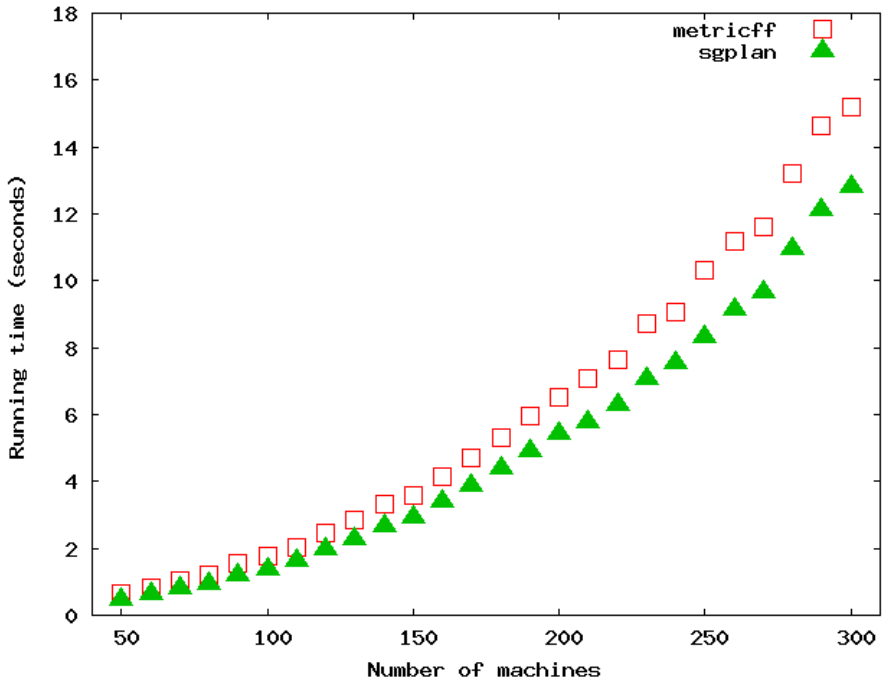


Network Simulator used

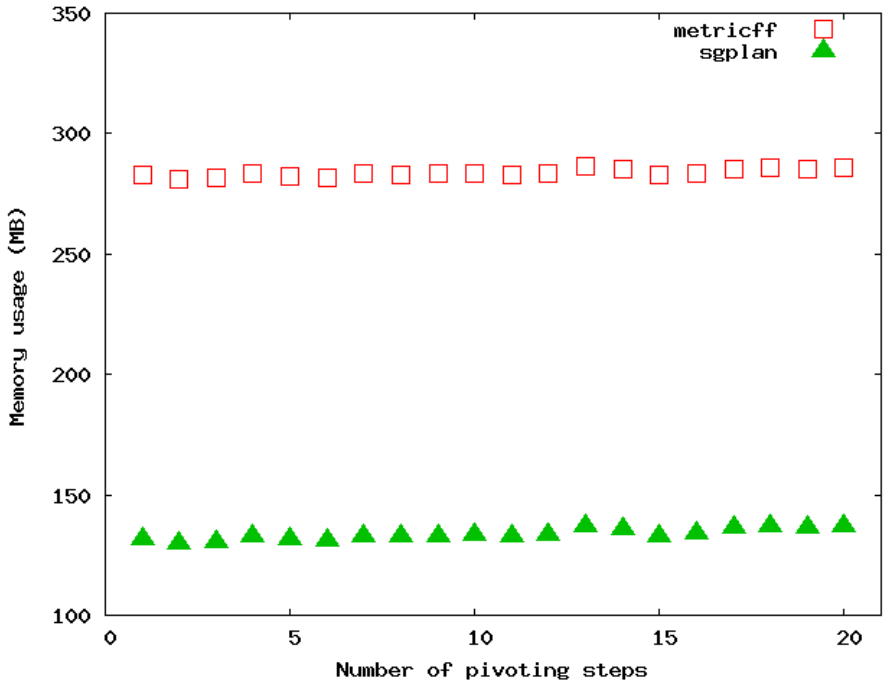
- 2009: "Simulating cyber-attacks for fun and profit"
  - presented at SimuTools (by F.Miranda)
- Network simulator designed to be
  - lightweight
    - simulates up to thousands of machines on one PC
  - realistic from the attacker's point of view
  - simulates at the system call level
  - transparent for a proxy-call agent

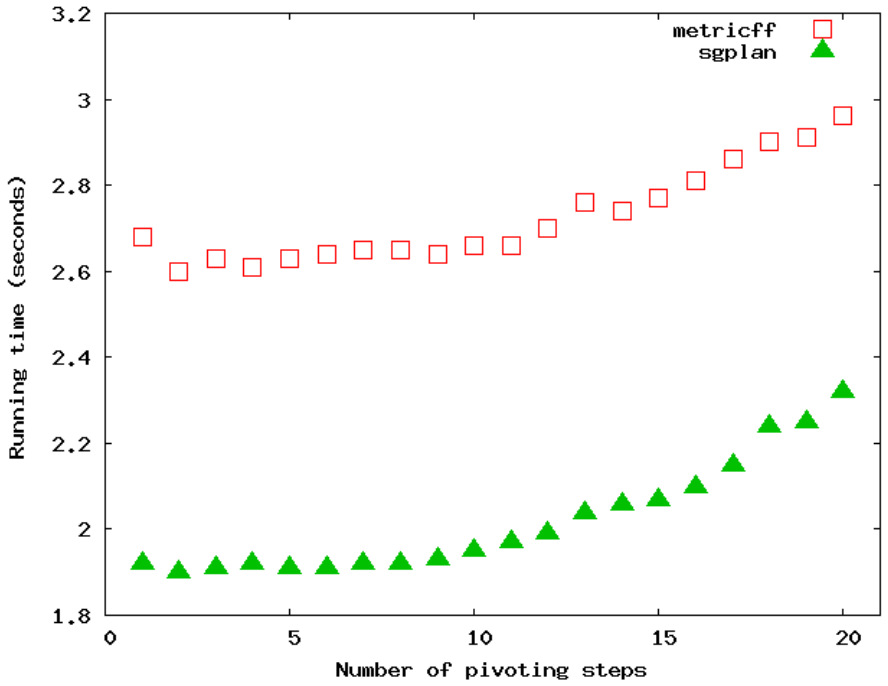
Increasing number of machines





Increasing number of pivoting steps





Conclusion / Future work

Conclusion

- Attack model realistic from the attacker's point of view
- The model scales to real-size scenarios
  - hundreds of machines
  - thousands of actions
- Don’t build the complete attack graph!
  - use Metric-FF or SGPlan to explore the state space
- Successful integration with pentesting framework
- Presented details of the PDDL representation
  - PDDL planners may be useful for other projects…

Open questions for future work

- Probabilistic planner algorithm
  - integrate and test in different scenarios
  - compare with other planners
- Parallel execution of actions
- Manage uncertainty
  - Now: use RPT information gathering before planning the attack phase
  - Idea: build a "metaplanner" to generate hypotheses about the unknown bits of information

Bibliography

References (1/3)

- Ammann, P.; Wijesekera, D.; and Kaushik, S. 2002. Scalable, graph-based network vulnerability analysis. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, 217–224. ACM New York, NY, USA.
- Arce, I., and McGraw, G. 2004. Why attacking systems is a good idea. *IEEE Computer Society - Security & Privacy Magazine* 2(4).
- Arce, I., and Richarte, G. 2003. State of the art security from an attacker's viewpoint. In *PacSec Conference, Tokyo, Japan*.
- Blum, A. L., and Furst, M. L. 1997. Fast planning through planning graph analysis. *Artificial Intelligence* 90(1-2):281 – 300.
- Burns, B.; Biondi, et al. 2007. *Security Power Tools*. O'Reilly Media.
- Chen, Y.; Wah, B. W.; and Hsu, C. 2006. Temporal planning using subgoal partitioning and resolution in SGPlan. *J. of Artificial Intelligence Research* 26:369.
- Fox, M., and Long, D. 2003. PDDL2. 1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research* 20(2003):61–124.

References (2/3)

- Futoransky, A.; Notarfrancesco, L.; Richarte, G.; and Sarraute, C. 2003. Building computer network attacks. *Technical report, CoreLabs*.
- Futoransky, A.; Miranda, F.; Orlicki, J.; and Sarraute, C. 2009. Simulating cyber-attacks for fun and profit. In *2nd Internation Conference on Simulation Tools and Techniques (SIMUTools '09).*
- Ghosh, N., and Ghosh, S. K. 2009. An intelligent technique for generating minimal attack graph. In *First Workshop on Intelligent Security (Security and Artificial Intelligence) (SecArt '09)*.
- Hoffmann, J. 2001. FF: The fast-forward planning system. *AI magazine* 22(3):57.
- Hoffmann, J. 2002. Extending FF to numerical state variables. In *Proceedings of the 15th European Conference on Artificial Intelligence (ECAI-02),* 571–575.
- Jajodia, S.; Noel, S.; and OBerry, B. 2005. Topological analysis of network attack vulnerability. *Managing Cyber Threats: Issues, Approaches and Challenges* 248–266.
- Lippmann, R., and Ingols, K. 2005. An annotated review of past papers on attack graphs. *Technical report, MIT Lincoln Laboratory*.
- Noel, S., and Jajodia, S. 2005. Understanding complex network attack graphs through clustered adjacency matrices. In *Proceedings of the 21st Annual Computer Security Applications Conference*, 160–169.

References (3/3)

- Noel, S.; Elder, M.; Jajodia, S.; Kalapa, P.; OHare, S.; and Prole, K. 2009. Advances in Topological Vulnerability Analysis. In *Proceedings of the 2009 Cybersecurity Applications & Technology Conference for Homeland Security*, 124–129. IEEE Computer Society.
- Phillips, C. A., and Swiler, L. P. 1998. A graph-based system for network-vulnerability analysis. In *Workshop on New Security Paradigms*, 71–79.
- Richarte, G. 2003. Modern intrusion practices. In *Black Hat Briefings*.
- Ritchey, R., and Ammann, P. 2000. Using model checking to analyze network vulnerabilities. In *IEEE Symposium on Security and Privacy*, 156–165. IEEE Computer Society.
- Sarraute, C., and Weil, A. 2008. Advances in automated attack planning. In *PacSec Conference, Tokyo, Japan*.
- Sarraute, C. 2009. New algorithms for attack planning. In *FRHACK Conference, Besançon, France*.
- Sheyner, O.; Haines, J.; Jha, S.; Lippmann, R.; and Wing, J. 2002. Automated generation and analysis of attack graphs. *In IEEE Symposium on Security and Privacy*, 273–284. IEEE Computer Society.

Thank you!

Carlos Sarraute → carlos@corest.com

Joint work with

Jorge Lucangeli Obes → jota@corest.com

Gerardo Richarte → gera@corest.com

http://corelabs.coresecurity.com