Using Strategy Objectives for Network Security Analysis

Elie Bursztein

Stanford University / LSV, Ens-Cachan

Inscrypt 2009

Introduction

Work purpose

Analyzing and anticipating computer networks attacks.

Network complexity: The Pentagon Case

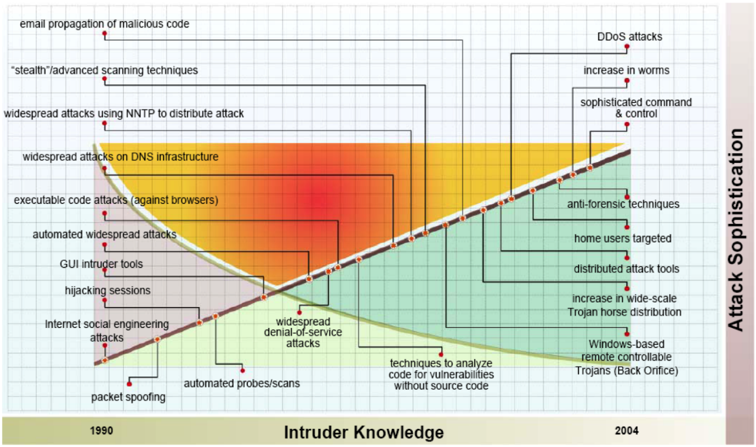
Huge network

- 15 000 LAN Networks
- 7 000 000 Computers

Huge Security problems

- Flash Drive banned due to a virus spread (Nov 2008).
- 1500 computers taken (Jun 2007)

Attack Complexity



Some Epic Failures

- 2004 Bouygues Telecom: 2 servers downs → 3 200 000 cellphones down
- 2005 Japan Mitsubishi: 1 computer infected → 40 MB of confidential reports leaked on a P2P network
- 2007 Apple: 1 computer in the production line infected → 150 000 ipods infected by the trojan RavMonE.exe

Outline

- Network Security
  - Attacks
- Game
- Strategy
- Automated Analysis
- Conclusion

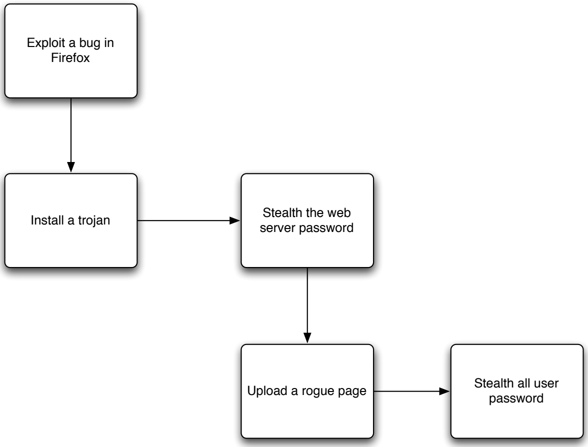
Vulnerabilities

- A vulnerability is a software bug that can be exploited by attacker to gain privilege.
- An exploit is the piece of software that takes advantage of a software bug.
- A 0day exploit is an exploit for an undisclosed vulnerability.

Vulnerabilities as Step stones

- Large networks may suffer multiple vulnerabilities
- Patches and counter-measures need to be prioritized
- A minor vulnerability can turn into a major hole when used as a step-stone

Illustration of a Complex attack



The Need for Automation

Attack analysis can't be done by hand: network and attack are just too complex and big for that.

We need models and tools for this !

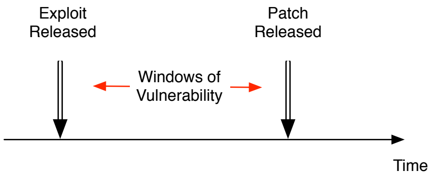
Attack Graph Frameworks

- 1998: Use of model-checking for host security [RS98]
- 2000: Use of model-cheking for network [RA00]
- 2004: First complete framework that constructs the attack scenario [SW04]
- 2005: Mulval [Ou05] a framework based on Datalog.
- 2006: NetSpa [ALI06] a framework that scale up to 50 000 nodes.

Time is the Essence

Network security is a race between Intruder and Administrator.

Windows of vulnerability



The Need for Time

Without time meaningless actions are allowed in the model.

- Administrator can patch 1000 services instantly.
- Intruder can compromise 1000 services before the administrator have a chance to react.

Without time concurrent actions can't be modeled.
Ex: *Administrator may patch a service while Intruder tries to exploit it.*

Time and Game

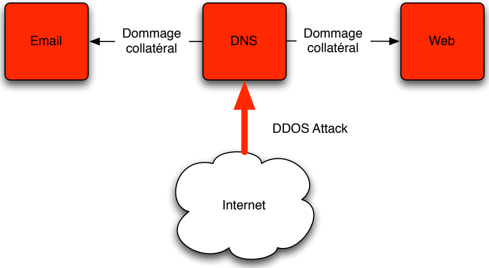
Model

Timed automaton game [AFHMS].

Property

Property can be written in Timed Alternating-Time Temporal Logic [AHK06].

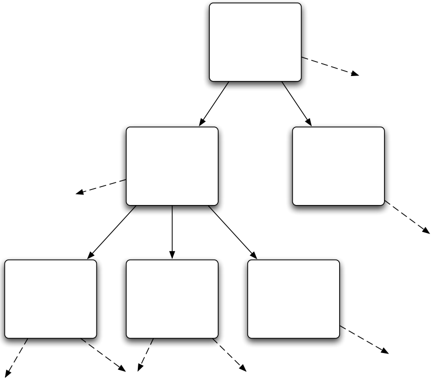
Collateral Effects



Outline

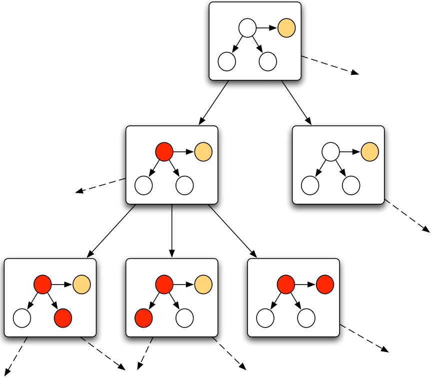
Dual layer structure

The Upper-layer is the timed automaton game, the Lower-layer represents the network state.



Dual layer structure

The Upper-layer is the timed automaton game, the Lower-layer represents the network state.



Lower-layer: the network state

The lower layer is composed of

- The dependency graph
- A set of states (atomic proposition)

Web Service Receipt

To build a web service you need:

- A HTTP frontend to serve the data

Web Service Receipt

To build a web service you need:

- A HTTP frontend to serve the data
- A SQL backend to store the data

Web Service Receipt

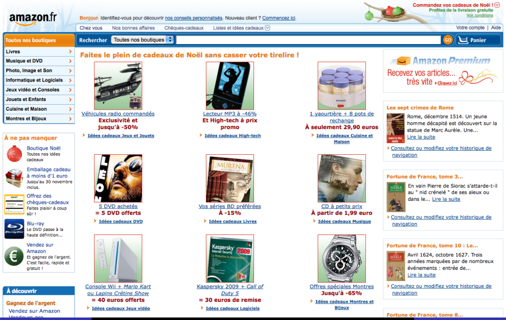
To build a web service you need:

- A HTTP frontend to serve the data
- A SQL backend to store the data
- A way to administrate the service

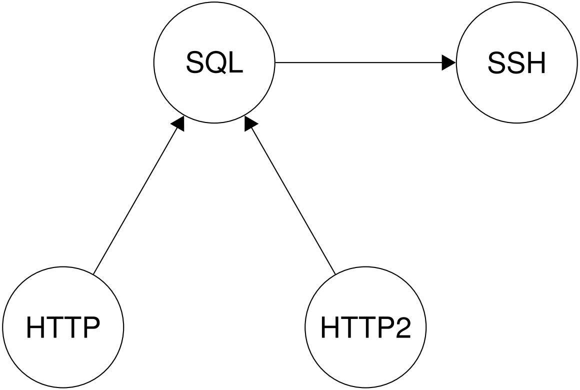
Web Service Receipt

To build a web service you need:

- A HTTP frontend to serve the data
- A SQL backend to store the data
- A way to administrate the service



The Dependency graph



Set of States

	SSH	SQL	HTTP1	HTTP2
Vulnerable	$\top$	$\bot$	$\bot$	$\bot$
Compromised	$\bot$	$\bot$	$\bot$	$\bot$

Rule Syntax

- φ_{pre}: Preconditions.

Rule syntax:

$$\Gamma : \quad \textbf{Pre } \varphi_{pre} \\ \longrightarrow \Delta, p, a, c \\ \textbf{Effect } \varphi_{eff}$$

Rule Syntax

Rule syntax:

$$\Gamma : \quad \textbf{Pre } \varphi_{pre} \longrightarrow \Delta, p, a, c \textbf{ Effect } \varphi_{eff}$$

- φ_{pre}: Preconditions.
- Δ: Time required to complete the action.

Rule Syntax

Rule syntax:

$$\Gamma : \quad \textbf{Pre } \varphi_{pre} \longrightarrow \Delta, p, a, c \quad \textbf{Effect } \varphi_{eff}$$

- φ_{pre}: Preconditions.
- Δ: Time required to complete the action.
- p: The player that executes the rule.

Rule Syntax

Rule syntax:

$$\Gamma : \quad \textbf{Pre } \varphi_{pre} \\ \longrightarrow \Delta, p, a, c \\ \textbf{Effect } \varphi_{eff}$$

- φ_{pre}: Preconditions.
- Δ: Time required to complete the action.
- p: The player that executes the rule.
- a: Rule name.

Rule Syntax

Rule syntax:

$$\Gamma : \quad \textbf{Pre } \varphi_{pre} \longrightarrow \Delta, p, a, c \ \textbf{Effect } \varphi_{eff}$$

- φ_{pre}: Preconditions.
- Δ: Time required to complete the action.
- p: The player that executes the rule.
- a: Rule name.
- c: Rule cost.

Rule Syntax

Rule syntax:

$$\Gamma : \quad \textbf{Pre } \varphi_{pre} \longrightarrow \Delta, p, a, c \quad \textbf{Effect } \varphi_{eff}$$

- φ_{pre}: Preconditions.
- Δ: Time required to complete the action.
- p: The player that executes the rule.
- a: Rule name.
- c: Rule cost.
- φ_{eff}: Effects.

Rule Syntax

Rule syntax:

$$\Gamma : \quad \textbf{Pre } \varphi_{pre} \longrightarrow \Delta, p, a, c \quad \textbf{Effect } \varphi_{eff}$$

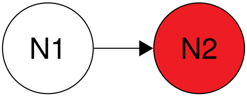
- φ_{pre}: Preconditions.
- Δ: Time required to complete the action.
- p: The player that executes the rule.
- a: Rule name.
- c: Rule cost.
- φ_{eff}: Effects.

Rules Syntax

$$
\begin{array}{lcll}
F & ::= & A & \text{atomic propositions, in } \mathcal{A} \\
  & | & \top & \text{true} \\
  & | & \neg F & \text{negation} \\
  & | & F \wedge F & \text{conjunction} \\
  & | & \Diamond F &
\end{array}
$$

◇ Semantics

◇ *Vulnerable*: One of the successors is vulnerable.

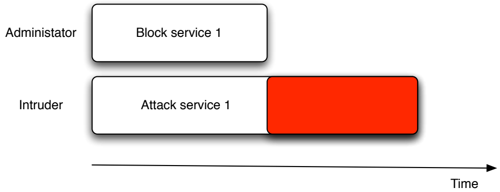


Rule Example

$$\Gamma: \quad \textbf{Pre } \textit{Vulnerable} \\ \longrightarrow 4, \text{A}, \text{Patch}, 500 \\ \textbf{Effect } \neg \textit{Vulnerable} \wedge \neg \textit{Compromise}$$

The Element of Surprise

if the opponent alters the service state *during the player rule execution then the player is taken by suprise!*



Decidability

Decidability

Model-checking TATL over anticipation games is EXPTIME-Complete [BGL,ASIA'07].

Outline

Network Security

Game

Strategy
- What is a strategy ?
- Using strategy
- Play Example

Automated Analysis

Conclusion

From counter-example to strategy

- An attack is a counter-example.

From counter-example to strategy

- An attack is a counter-example.
- Typically you end-up with many counter-examples.

From counter-example to strategy

- An attack is a counter-example.
- Typically you end-up with many counter-examples.

The problem

Which counter-example should the administrator look at first ?

From counter-example to strategy

- An attack is a counter-example.
- Typically you end-up with many counter-examples.

The problem

Which counter-example should the administrator look at first ?

- Which attack is the most devastating ?
- What service to patch first ?

Costs and Rewards

To find the most meaningful counter-example we need some additional informations.

Costs and Rewards

To find the most meaningful counter-example we need some additional informations.

- Cost: Each action has a cost.

Costs and Rewards

To find the most meaningful counter-example we need some additional informations.

- Cost: Each action has a cost.
- Reward: Each network asset has a value.

Costs and Rewards

To find the most meaningful counter-example we need some additional informations.

- Cost: Each action has a cost.
- Reward: Each network asset has a value.

$$
\begin{array}{lcll}
\mathcal{O} & ::= & O & \text{Objective} \in \phi \\
 & | & \mathcal{O} \wedge \mathcal{O} & \\
 & | & MAX(O) & \text{maximize the value} \\
 & | & MIN(O) & \text{minimize the value} \\
 & | & O < x & x \in \mathbb{N} \\
 & | & O > x & x \in \mathbb{N}
\end{array}
$$

Relation between Cost and Time

Assumption

The faster an action is, the more costly it is.

Real world examples of this assumption:

- Exploit: 0day versus Public exploit.
- Response team: 24/24h versus 8h /day

Definition

Strategy objectives are a tuple:

$$\mathcal{S} = (Na, Pl, Ob, Or, Co)$$

- *Na*: Strategy name
- *Pl*: The player
- *Ob*: Numerical objectives
- *Or*: Strict preference order
- *Co*: Constraints.

Example

$\mathcal{S} = (Patch, A, Min(Cost) \wedge Max(OCost)$
$, OCost > Cost, \blacksquare \neg Compromised)$

Computing Assets value

- Using the same value for each asset.
- Assigning value by hand.
- Computing automatically the value with a ranking algorithm [EB,INSCRYPT'08].

Which Objectives to choose ?

- Minimizing cost (patch)
- Maximizing reward (attack)

Which Objectives to choose ?

- Minimizing cost (patch)
- Maximizing reward (attack)

Wrong answer !

Player performs the best when his opponent makes mistakes.

Game theory classical optimal criterion such as Nash equilibrium and Pareto optimality are not applicable.

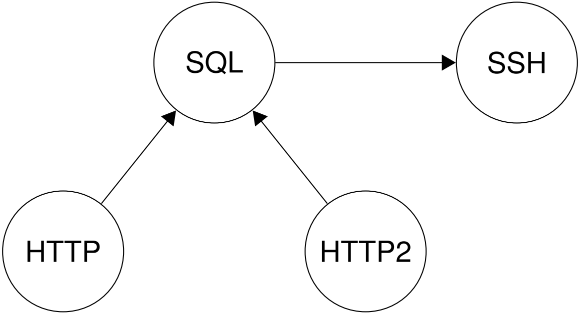
Dominant Strategy

The notion of dominant strategy was informally introduced in biology [H67] in 1967.

(Strictly) Dominant Strategy

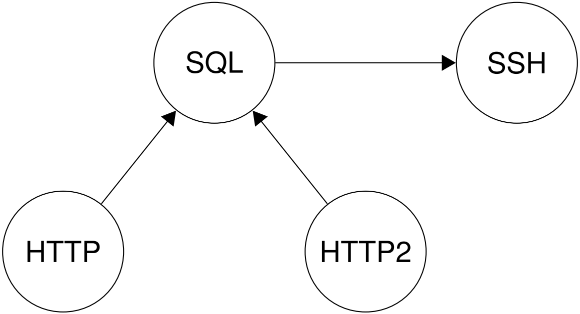
The (strictly) dominant strategy is the player strategy that beats the maximum number of (every) opponent strategies.

The Lower Layer



	SSH	SQL	HTTP1	HTTP2
Vulnerable	$\top$	$\bot$	$\bot$	$\bot$
Compromised	$\bot$	$\bot$	$\bot$	$\bot$

The Lower Layer



	SSH	SQL	HTTP1	HTTP2
Value	1	100	10	10

Intruder Rules

Γ : **Pre** *Vulnerable* $\wedge \neg$*Compromise*
$\longrightarrow$ 2, I, Exploit 0day, 20000
Effect *Compromise*

Intruder Rules

$$\Gamma : \quad \textbf{Pre}\ \mathit{Vulnerable} \wedge \neg \mathit{Compromise}$$
$$\longrightarrow 2, \text{I}, \text{Exploit 0day}, 20000$$
$$\textbf{Effect}\ \mathit{Compromise}$$
$$\Gamma : \quad \textbf{Pre}\ \mathit{Vulnerable} \wedge \neg \mathit{Compromise}$$
$$\longrightarrow 10, \text{I}, \text{Exploit Public}, 500$$
$$\textbf{Effect}\ \mathit{Compromise}$$

Intruder Rules

Γ : **Pre** *Vulnerable* $\wedge \neg$*Compromise*
$\longrightarrow$ 2, I, Exploit 0day, 20000
Effect *Compromise*

Γ : **Pre** *Vulnerable* $\wedge \neg$*Compromise*
$\longrightarrow$ 10, I, Exploit Public, 500
Effect *Compromise*

Γ : **Pre** $\neg$*Compromise* $\wedge \Diamond$*Compromised*
$\longrightarrow$ 1, I, Propagation, 5000
Effect *Compromise*

Intruder Rules

Γ : **Pre** $Vulnerable \wedge \neg Compromise$
$\longrightarrow$ 2, I, Exploit 0day, 20000
Effect $Compromise$

Γ : **Pre** $Vulnerable \wedge \neg Compromise$
$\longrightarrow$ 10, I, Exploit Public, 500
Effect $Compromise$

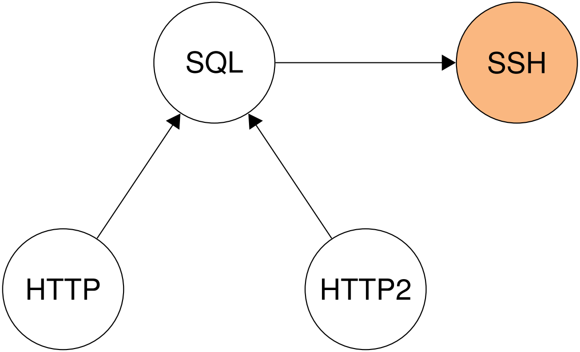
Γ : **Pre** $\neg Compromise \wedge \Diamond Compromised$
$\longrightarrow$ 1, I, Propagation, 5000
Effect $Compromise$

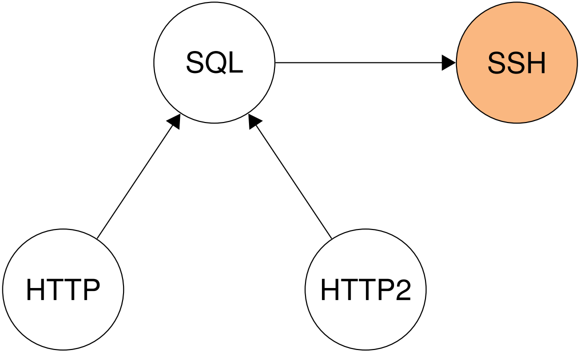
Administrator Rules

$$\Gamma : \quad \textbf{Pre}\ \textit{Vulnerable} \longrightarrow 4, A, \text{Patch}, 500\ \textbf{Effect}\ \neg \textit{Vulnerable} \wedge \neg \textit{Compromise}$$

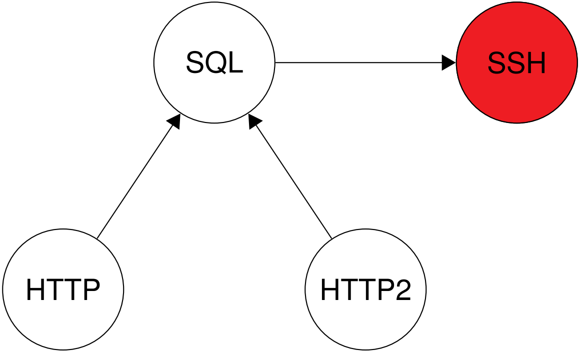
Strategy

$$S = (\text{Attack}, I, MAX(Reward) \wedge Max(OCost), OCost > Reward, \blacklozenge Compromised)$$

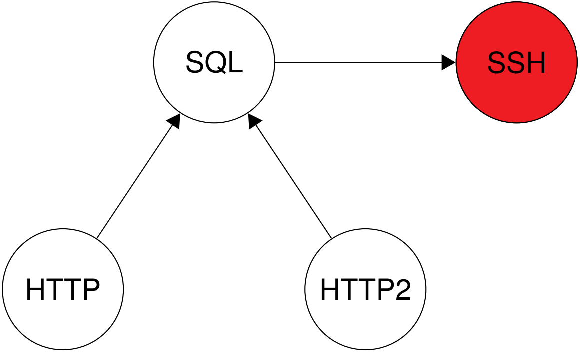

SQL
SSH
HTTP
HTTP2



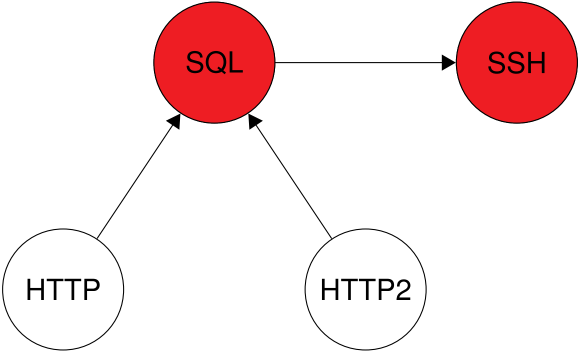
T	P	Action	Rule	Target	Succ	Payoff	Cost
0	A	choose	Patch	SSH	$\perp$	-	-
0	I	choose	Exp 0 Day	SSH	$\perp$	-	-



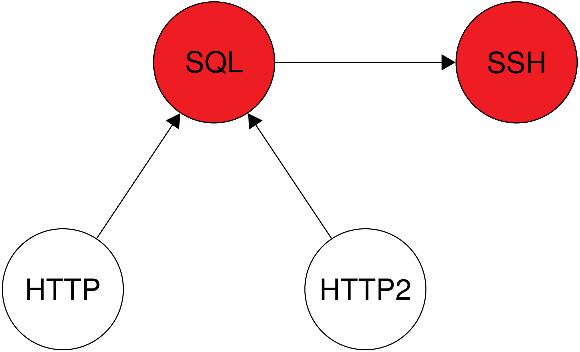
T	P	Action	Rule	Target	Succ	Payoff	Cost
	A	In Progress	Patch	SSH	$\perp$	-	-
2	I	execute	Exp 0 Day	SSH	$\perp$	1	20000



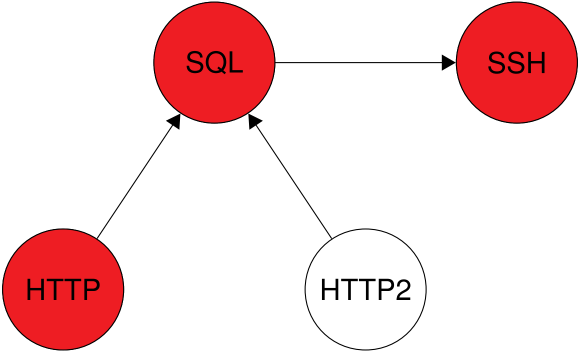
T	P	Action	Rule	Target	Succ	Payoff	Cost
	A	In Progress	Patch	SSH	$\perp$	-	-
2	I	choose	propagation	SQL	SSH	-	-



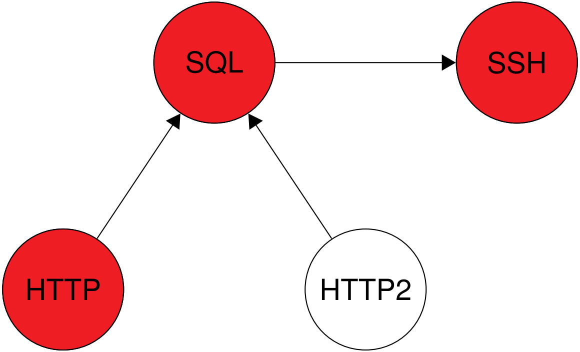
T	P	Action	Rule	Target	Succ	Payoff	Cost
	A	In Progress	Patch	SSH	$\perp$	-	-
3	I	**execute**	propagation	SQL	SSH	101	25000



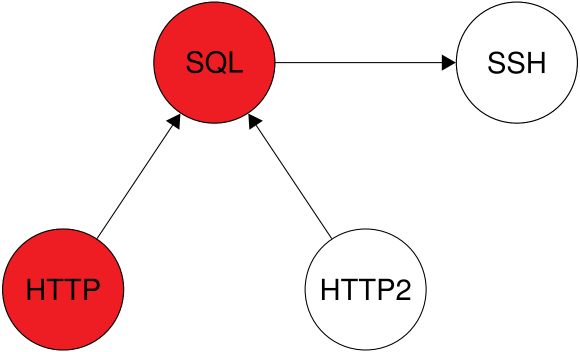
T	P	Action	Rule	Target	Succ	Payoff	Cost
	A	In Progress	Patch	SSH	$\perp$	-	-
3	I	choose	propagation	HTTP1	SQL	-	-



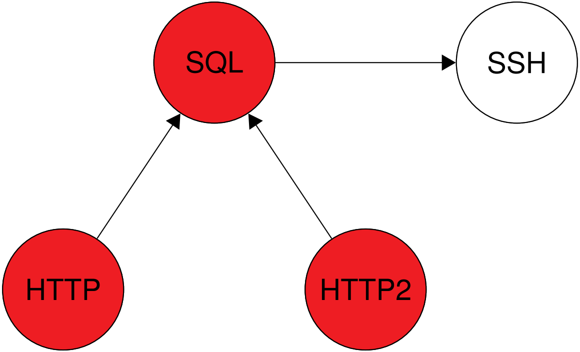
T	P	Action	Rule	Target	Succ	Payoff	Cost
	A	In Progress	Patch	SSH	$\perp$	-	-
4	I	**execute**	propagation	HTTP1	SQL	111	30000



T	P	Action	Rule	Target	Succ	Payoff	Cost
	A	In Progress	Patch	SSH	$\perp$	-	-
4	I	choose	propagation	HTTP2	SQL	-	-



T	P	Action	Rule	Target	Succ	Payoff	Cost
4	A	**execute**	Patch	SSH	SQL	1	500
	I	InProgress	propagation	HTTP2	SQL	-	-



T	P	Action	Rule	Target	Succ	Payoff	Cost
	A		$\perp$			1	500
5	I	**execute**	propagation	HTTP2	SQL	121	35000

Extending the model

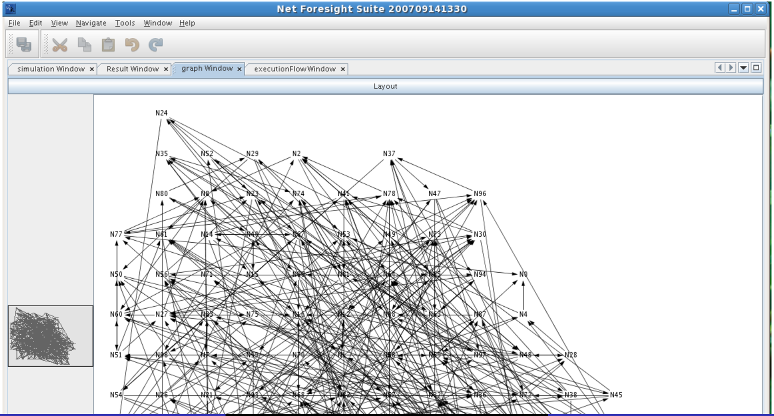
We extended the anticipation game framework [EB,FAST'08] in order to model

- Multiples network cooperation
- Cost over the time (penalty)
- Timeline of events

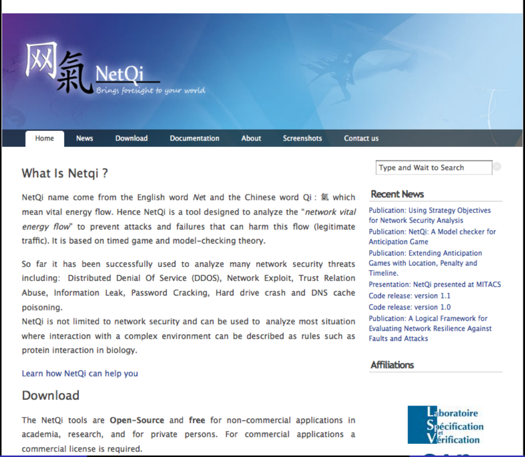
Outline

The Tool

We create an implementation in C ($\approx$ 6500 lines) of the anticipation game framework called NetQi [EB,ATVA'08].



HomePage


NetQi
Brings foresight to your world
Home
News
Download
Documentation
About
Screenshots
Contact us
What Is Netqi ?
NetQi name come from the English word Net and the Chinese word Qi : 氣 which mean vital energy flow. Hence NetQi is a tool designed to analyze the "network vital energy flow" to prevent attacks and failures that can harm this flow (legitimate traffic). It is based on timed game and model-checking theory.
So far it has been successfully used to analyze many network security threats including: Distributed Denial Of Service (DDOS), Network Exploit, Trust Relation Abuse, Information Leak, Password Cracking, Hard drive crash and DNS cache poisoning.
NetQi is not limited to network security and can be used to analyze most situation where interaction with a complex environment can be described as rules such as protein interaction in biology.
Learn how NetQi can help you
Download
The NetQi tools are Open-Source and free for non-commercial applications in academia, research, and for private persons. For commercial applications a commercial license is required.
Type and Wait to Search
Recent News
Publication: Using Strategy Objectives for Network Security Analysis
Publication: NetQi: A Model checker for Anticipation Game
Publication: Extending Anticipation Games with Location, Penalty and Timeline.
Presentation: NetQi presented at MITACS
Code release: version 1.1
Code release: version 1.0
Publication: A Logical Framework for Evaluating Network Resilience Against Faults and Attacks
Affiliations
Laboratoire Spécification et Vérification

Case study

Nb Nodes	Nb Dep	Strategy	type	Time
5200	27000	Defense	Exact	2.4 sec
5200	27000	Intrusion	Approximate	55 sec

Outline

Network Security

Game

Strategy

Automated Analysis

Conclusion

Conclusion

In this work we have

- Developed the notion of strategy
- Show how strategy allow to select the most interesting play
- Implemented the model in order to show the effectiveness of the approach.

Perspective

- Finding network key services.
- Using dynamic costs and rewards.
- Modeling various classes of attackers.