Economics of Security

I. Overview
What is security?

Ensure the intended operation of a system in the presence of adversaries.

What is safety?
Who are the players?

- users
- ...
Who are the players?

- Users
- ISP
- Content provider
- Adversary
- Software vendor
Security solutions

- don’t be online
- educate users
- ISP surveillance
- better software standards
- cyber-insurance
- understand the attackers
Attacker’s advantage

- **technology**
  - attack proactive – defense reactive
  - attack cheap – defense expensive
  - attack easy to measure – defense hard to measure

- **policy**
  - attack illegal – defense must be lawful

- **human factor**
  - attack primary goal – defense secondary goal
Security solutions per discipline

- technology
  - Security Protocols
  - Network Security

- policy
  - Security Policy
  - Economics of Security
  - Public Policy

- human factor
  - Security Incentives
  - Economics of Security
Games people play

- Understand human behavior
  - microeconomics – game examples?
- everything is a game
- define a game:
  - players
  - strategy
  - payoff
Prisoner’s Dilemma

goal: minimize cost
Prisoner’s Dilemma

<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-2,-2</td>
<td>-5,-1</td>
</tr>
<tr>
<td>C</td>
<td>-1,-5</td>
<td>-3,-3</td>
</tr>
</tbody>
</table>

goal: maximize payoff
### Prisoner’s Dilemma - solution

A goal: maximize payoff

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>3,3</td>
<td>1,4</td>
</tr>
<tr>
<td>D</td>
<td>4,1</td>
<td>2,2</td>
</tr>
</tbody>
</table>
Prisoner’s Dilemma - solution

A
B
C
D

Pareto optimal

3,3

1,4

4,1

2,2

Nash equilibrium

goal: maximize payoff
Example
- Players: A and B
- Strategies: C or D
- Payoff: matrix

General
- Players: n players (1..n), player i
- Strategies: $s_i$
- Payoff: $U_i$
Concepts

- strictly dominates
  \[ u_i(s_i, s_{-i}) > u_i(s'_i, s_{-i}), \forall s_{-i} \]
- iterated strict dominance
  - iteratively exclude dominated strategies
- Nash equilibrium
  \[ u_i(s^*_i, s^*_{-i}) > u_i(s'_i, s^*_{-i}), \forall s'_i \]
- Pareto-optimal
  - no player can increase her payoff without decreasing the payoff of another player

(J Nash, “Equilibrium points in N-person games”, 1950)
Prisoner’s Dilemma - solution

group A
A

B

C

D

C

3,3

1,4

D

4,1

2,2

goal: maximize payoff
Prisoner’s Dilemma - solution

A  B  C  D
A  3,3  1,4
B  4,1  2,2
C  3,3  1,4
D  4,1  2,2

goal: maximize payoff
Prisoner’s Dilemma - solution

![Prisoner's Dilemma payoff matrix]

goal: maximize payoff
Security Prisoner’s Dilemma

Information sharing

<table>
<thead>
<tr>
<th></th>
<th>Share</th>
<th>Not</th>
</tr>
</thead>
<tbody>
<tr>
<td>Share</td>
<td>4-1,4-1</td>
<td>2-1,4</td>
</tr>
<tr>
<td>Not</td>
<td>4,2-1</td>
<td>2,2</td>
</tr>
</tbody>
</table>

Examples:
- virus scanners
- blacklist companies
- …
Establishing cooperation

- Prisoner’s Dilemma = bad news
- Can we do better?
  – Yes: iterated Prisoner’s Dilemma
Repeated games

- repeated interaction between the players (in stages)
- **move**: decision in one interaction
- **strategy**: defines how to choose the next move, given the previous moves
- **history**: the ordered set of moves in previous stages
  - most prominent games are history-1 games (players consider only the previous stage)
- **initial move**: the first move with no history
- finite-horizon vs. infinite-horizon games
- stages denoted by $t$ (or $k$)
Utilities: Objectives in the repeated game

- finite-horizon vs. infinite-horizon games
- myopic vs. long-sighted repeated game

**Myopic:** \( \overline{u}_i = u_i(t + 1) \)

**Long-sighted finite:** \( \overline{u}_i = \sum_{t=0}^{T} u_i(t) \)

**Long-sighted infinite:** \( \overline{u}_i = \sum_{t=0}^{\infty} u_i(t) \)

**Payoff with discounting:** \( \overline{u}_i = \sum_{t=0}^{\infty} u_i(t) \cdot \omega^t \)

\( 0 < \omega \leq 1 \) is the discounting factor
Strategies in the Iterated Prisoner’s Dilemma game

- usually, history-1 strategies, based on different inputs:
  - others’ behavior: \( m_i(t+1) = s_i[m_{-i}(t)] \)
  - others’ and own behavior: \( m_i(t+1) = s_i[m_i(t), m_{-i}(t)] \)
  - payoff: \( m_i(t+1) = s_i[u_i(t)] \)

Example strategies in the IPD:

<table>
<thead>
<tr>
<th>Blue (t)</th>
<th>initial move</th>
<th>F</th>
<th>D</th>
<th>strategy name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red (t+1)</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>AllC</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>F</td>
<td>D</td>
<td>Tit-For-Tat (TFT)</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>AllD</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>D</td>
<td>F</td>
<td>Anti-TFT</td>
</tr>
</tbody>
</table>
Theorem: In the Iterated Prisoner’s Dilemma, if both players play AllD, it is a Nash equilibrium.

Theorem: In the Iterated Prisoner’s Dilemma, both players playing TFT is a Nash equilibrium as well.

The Nash equilibrium \( s_{\text{Blue}} = \text{TFT} \) and \( s_{\text{Red}} = \text{TFT} \) is Pareto-optimal (but \( s_{\text{Blue}} = \text{AllD} \) and \( s_{\text{Red}} = \text{AllD} \) is not)!
Chicken Game

II. Game Theory Primer
Concepts

- **best response**
  \[ u_i(s_i, s_{-i}) > u_i(s'_i, s_{-i}), \forall s'_i \]

- **Nash equilibrium**
  \[ u_i(s^*_i, s^*_{-i}) > u_i(s'_i, s^*_{-i}), \forall s'_i, \forall i \]
  - mutual best responses

- **Pareto-optimal**
  - no player can increase her payoff without decreasing the payoff of another player
Security Chicken Game

Security investment

Examples:
- sysadmins working
- providing blacklists
- students doing homeworks

Free riding
Practice: Competing botnets

- McColo takedown 2008
  - several botnets had C&C servers
  - Srizbi, Rustock
  - Game:
    - one server each
    - aggressive: 10, cost: 2
    - low-key: 8, cost: 1
    - discovered: 5
Tragedy of the commons

- Free riding is a problem
  (G. Hardin, “The Tragedy of the commons”, 1968)
Economics of Security

III.
Example: Optimal security investment
Information security investment

How much to spend?

Where to invest?
Security investment with uncertainty

**perfect info – true cost of attacks**

**uncertainty – true cost of attacks**

---

**III. Optimal Security Investments**
Iterated Weakest Link (IWL) model

true cost of attack

expected cost of attack

proactive defense \((k_1)\)

reactive defense

R. Böhme and T. Moore, WEIS 2009
Iterated Weakest Link (IWL) model

- true cost of attack
- expected cost of attack $k_1$
- proactive defense
- reactive defense

**decide**

Defender

R. Böhme and T. Moore, WEIS 2009
IWL – Total profit

a – asset
r – return if no attack [0,1]
z – loss due to attack [0,1]
σ – attack cost uncertainty
1 – unit defense cost

proactive defense ($k_1$)

Defender

Attacker

<table>
<thead>
<tr>
<th>t</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>benefit</td>
<td>a(r-z)</td>
<td>a(r-z)</td>
<td>a(r-z)</td>
<td>ar</td>
<td>ar</td>
<td>ar</td>
</tr>
<tr>
<td>cost</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>

R. Böhme and T. Moore, WEIS 2009

Economics of Security

III. Optimal Security Investments
Reactive defense summary

reactive defense compared to proactive

attack intensity

security spending

ROSI

R. Böhme and T. Moore, WEIS 2009
Pentesting model

true cost of attack

expected cost of attack

proactive defense ($k_1$)

Defender

reactive defense

Attacker
Pentesting model

**Economics of Security**

### III. Optimal Security Investments

**true cost of attack**

**expected cost of attack**

**proactive defense**

- **penetration testing**

**decide**

**Defender**

**Attacker**

1. True cost of attack
2. Expected cost of attack
3. Proactive defense ($k_1$)
4. Penetration testing
5. Reactive defense

---

**Note:** The diagram illustrates the decision-making process in pentesting, focusing on the alignment of costs and investments for optimal security. The elements marked with 'X' indicate points of decision or action.
Extensive-form games

- usually to model sequential decisions
- game represented by a tree
- **Sequential** Chicken (Security Investment) game: Blue plays first, then Red plays.

Reward for successful defense: 4
Cost of defense: 1

\[(4-1,4-1) (4-1,4) (4,4-1) (1,1)\]
The strategy defines the moves for a player for every node in the game, even for those nodes that are not reached if the strategy is played.

strategies for Blue: I, N

strategies for Red: II, IN, NI and NN

If they have to decide independently: three Nash equilibria

(N,NI), (N,II) and (I,NN)
Backward induction

- Solve the game by reducing from the final stage
- Eliminates Nash equilibria that are *incredible threats*

### Incredible threat: (I, NN)

\[
\begin{array}{c|c|c}
\text{Red} & \text{Blue} \\
\hline
\text{I} & (4-1,4-1) & (4-1,4) \\
\text{N} & (4,4-1) & (1,1) \\
\end{array}
\]
Subgame perfection

- Extends the notion of Nash equilibrium

**One-deviation property:** A strategy $s_i$ conforms to the *one-deviation property* if there does not exist any node of the tree, in which a player $i$ can gain by deviating from $s_i$ and apply it otherwise.

**Subgame perfect equilibrium:** A strategy profile $s$ constitutes a subgame perfect equilibrium if the one-deviation property holds for every strategy $s_i$ in $s$.

Finding subgame perfect equilibria using backward induction

Subgame perfect equilibria: $(N, NI)$ and $(N, II)$

---

<table>
<thead>
<tr>
<th>State</th>
<th>Payoffs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red I</td>
<td>(4-1,4-1)</td>
</tr>
<tr>
<td>Blue N</td>
<td>(4-1,4)</td>
</tr>
<tr>
<td>Red I</td>
<td>(4,4-1)</td>
</tr>
<tr>
<td>Red N</td>
<td>(1,1)</td>
</tr>
</tbody>
</table>
Pentesting model

a – asset
r – return if no attack [0,1]
z – loss due to attack [0,1]
σ – attack cost uncertainty
1 – unit defense cost
c – cost of pentesting

<table>
<thead>
<tr>
<th>t</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>benefit</td>
<td>a(r-z)</td>
<td>ar</td>
<td>ar</td>
<td>ar</td>
<td>ar</td>
<td>ar</td>
</tr>
<tr>
<td>cost</td>
<td>4+c</td>
<td>6+c</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

III. Optimal Security Investments
Pentesting helps - costs

\[ k \]

\[ \text{ } a \cdot z \]

\[ K \]

\[ k \]

\[ 0 \]

\[ t \]
Pentesting helps - costs
Pentesting helps – total profit
Return on pentesting (ROPT)

\[
ROPT = \frac{\text{loss (security + no pentesting)} - \text{loss (security + pentesting)}}{\text{avg. security investment}}
\]
Pentesting summary

reactive defense compared to proactive

penetration testing compared to reactive w/o pentesting

attack intensity
security spending
ROSI
ROPT