Characterizing and Aggregating Attack Graph-based Security Metrics

MIT Lincoln Labs Seminar

July 7, 2010

Nwokedi C. Idika

The Institute for Information Infrastructure Protection (I3P) on Security Metrics

Security metrics is considered a top **4** research & development priority through 2019

INFOSEC Research Council (IRC) on Security Metrics

Enterprise security metrics considered a top **8** research priority through 2015

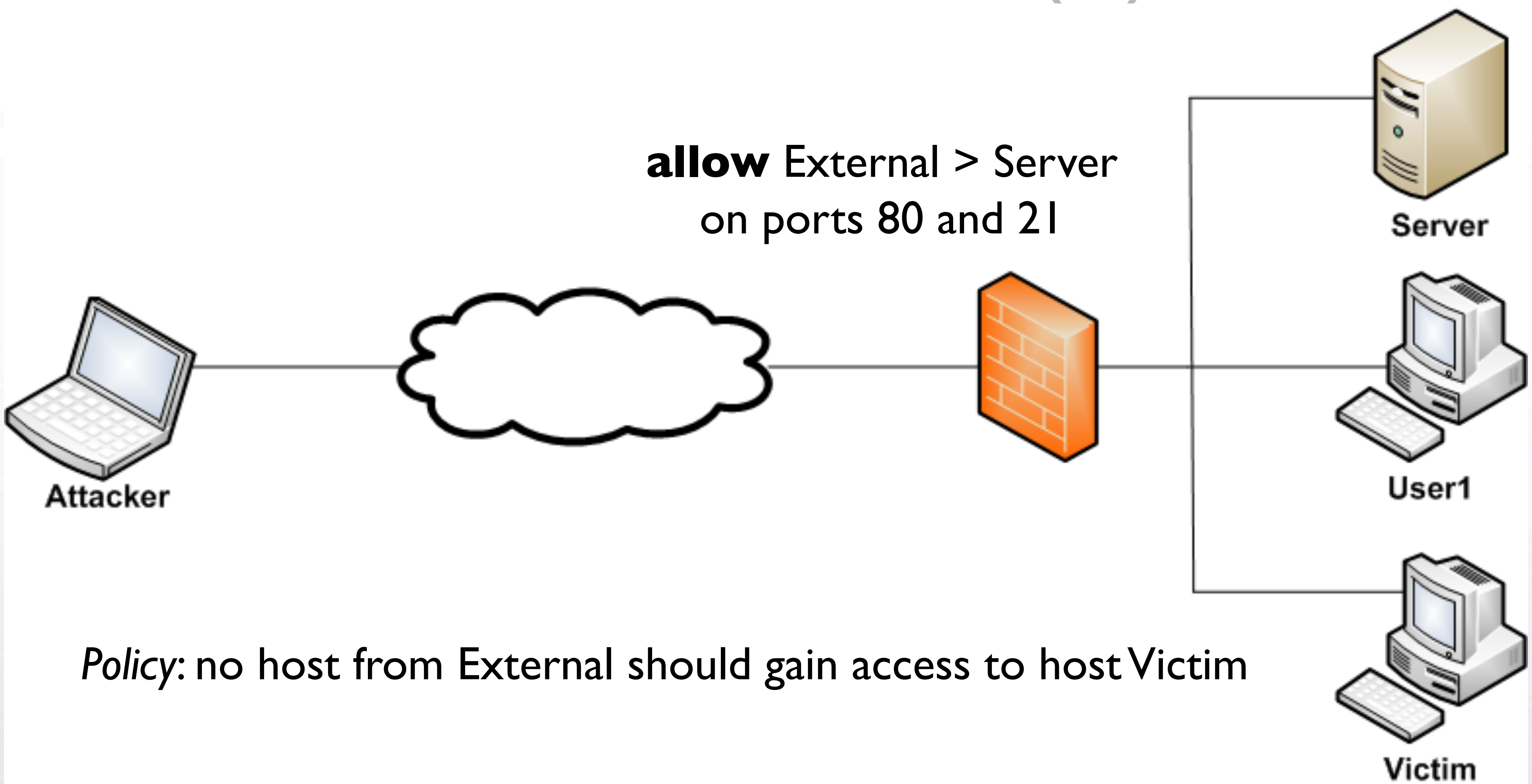
Attack Graph-based Security Metric

- a value derived from measuring attack graph properties

Attack Graph

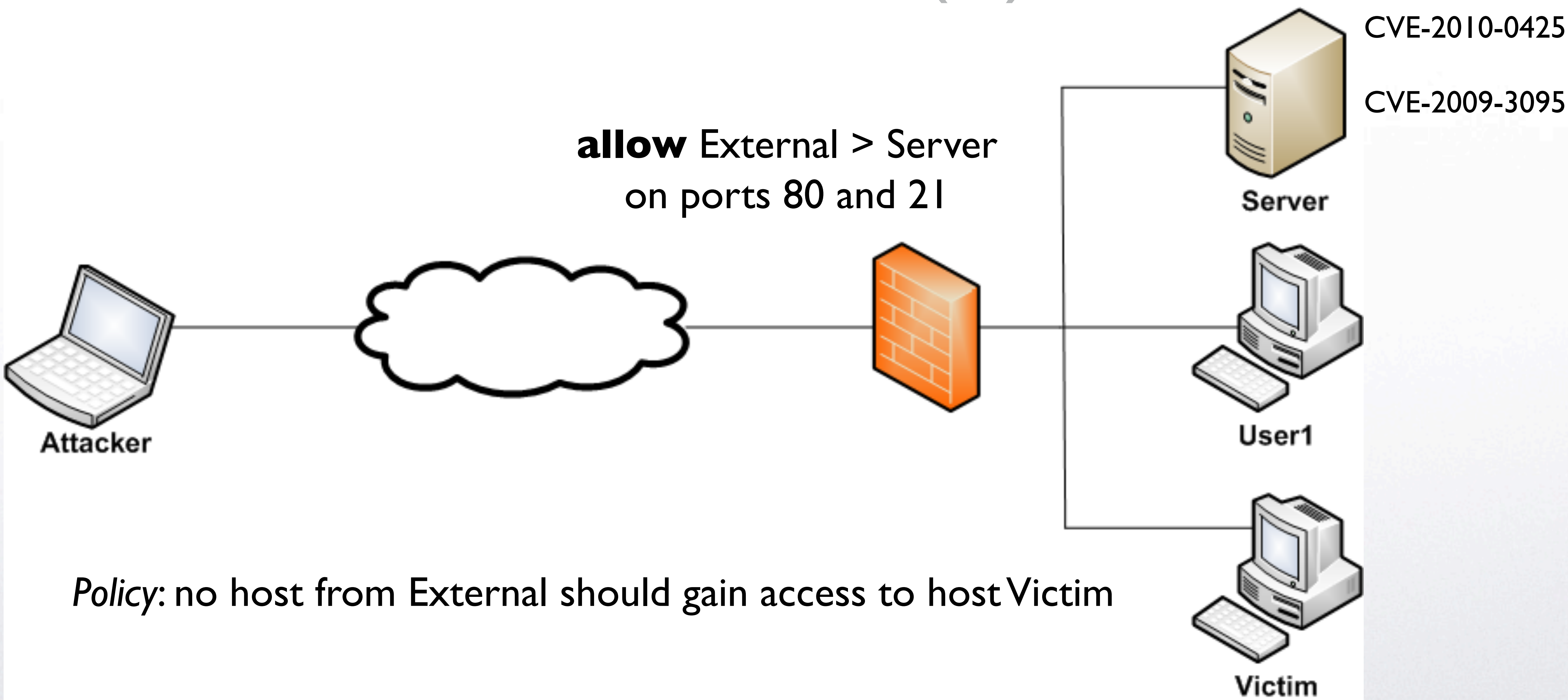
- an abstraction divulging the *potential* ways an attacker can leverage interdependencies among vulnerabilities to violate a security policy

The Attack Graph Generation Process (1)



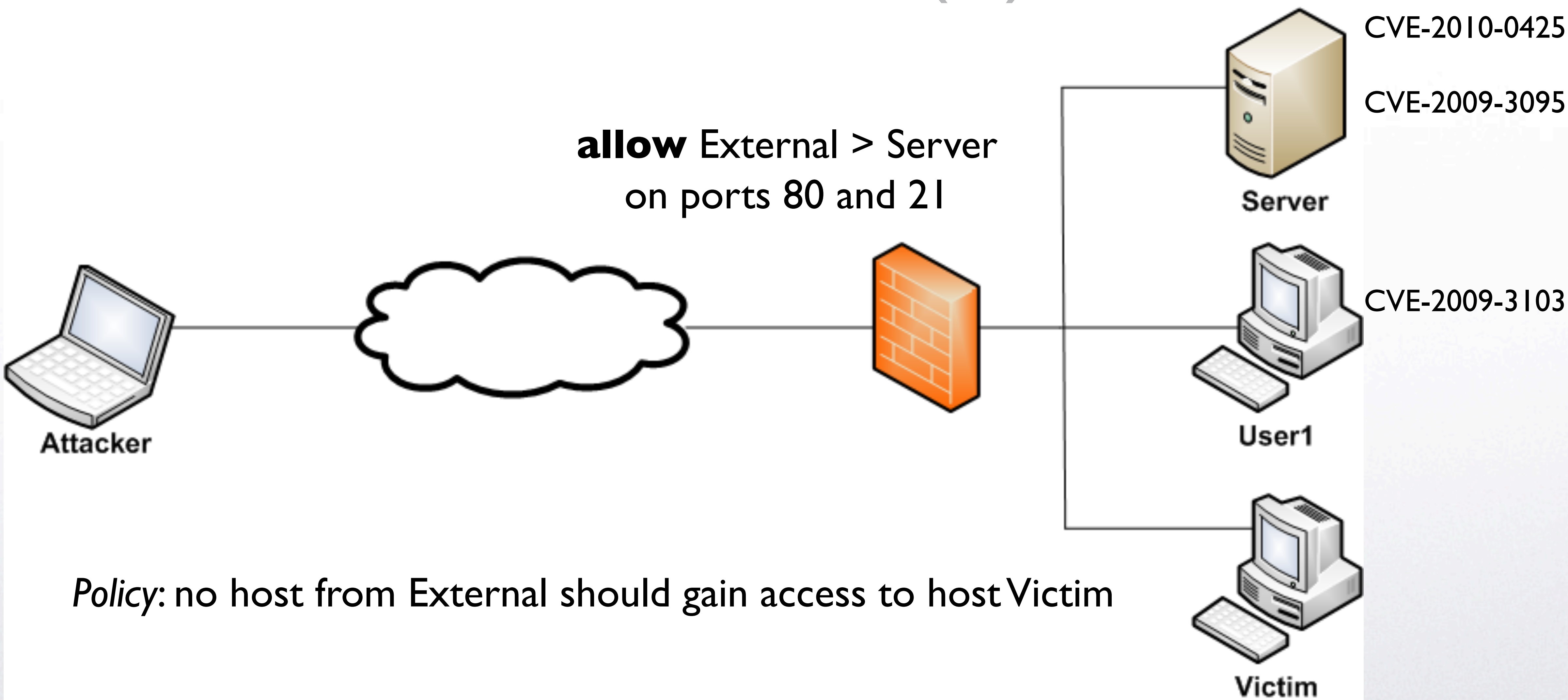
Policy: no host from External should gain access to host Victim

The Attack Graph Generation Process (I)

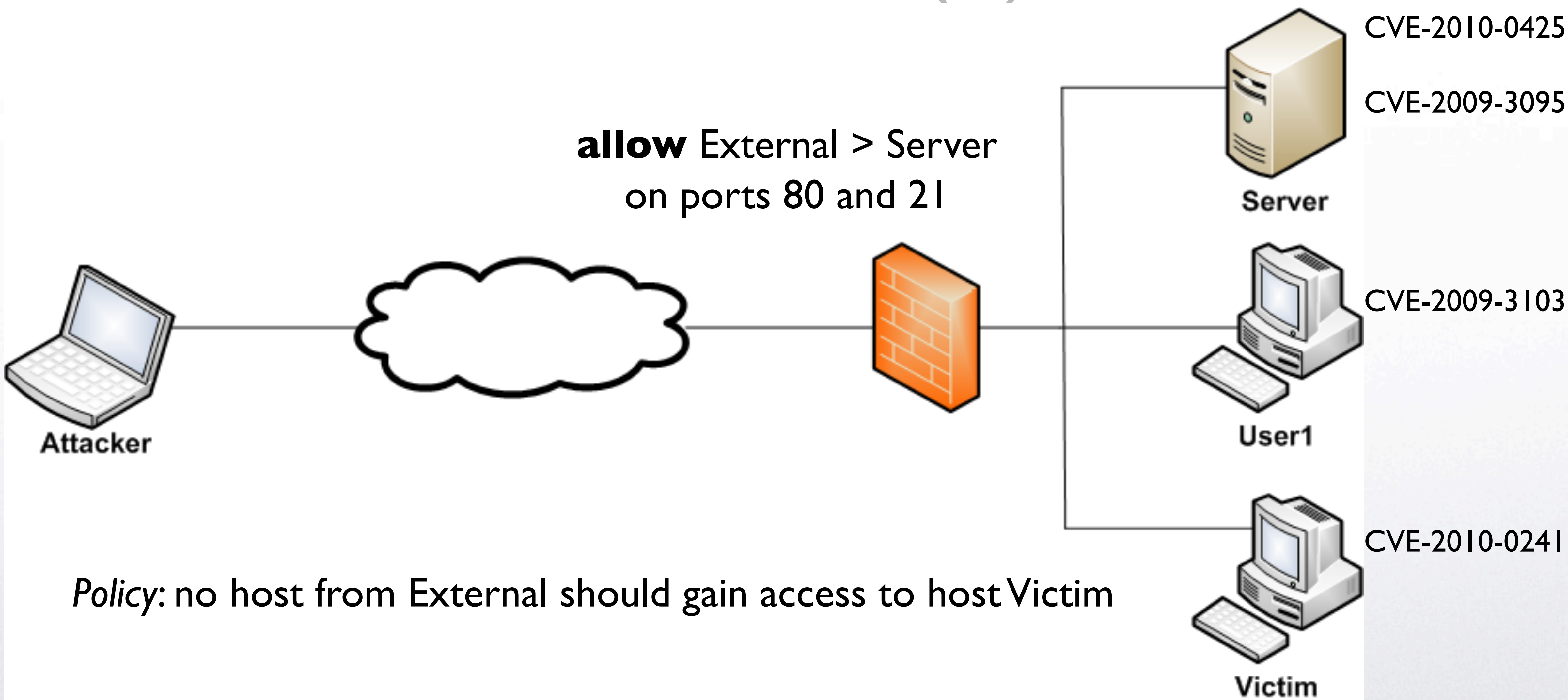


Policy: no host from External should gain access to host Victim

The Attack Graph Generation Process (I)

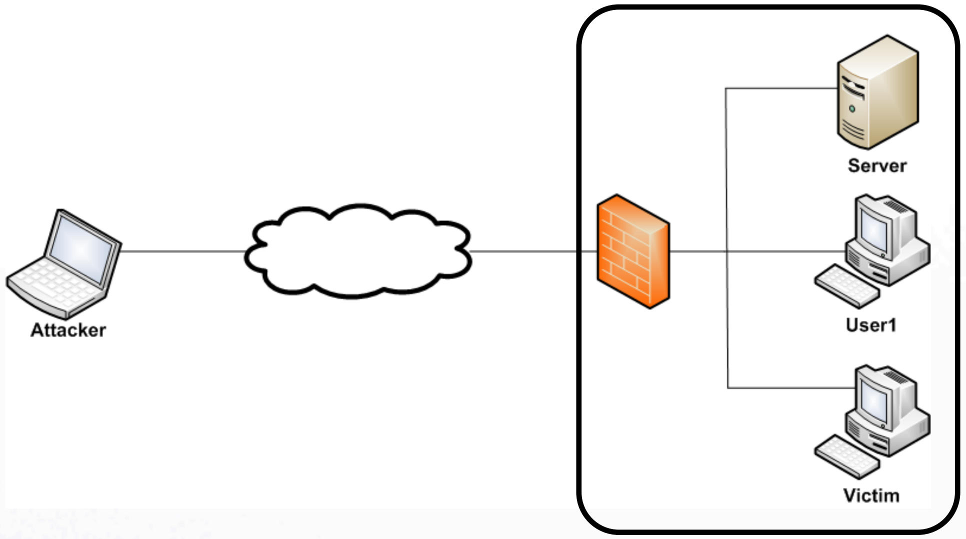


The Attack Graph Generation Process (I)



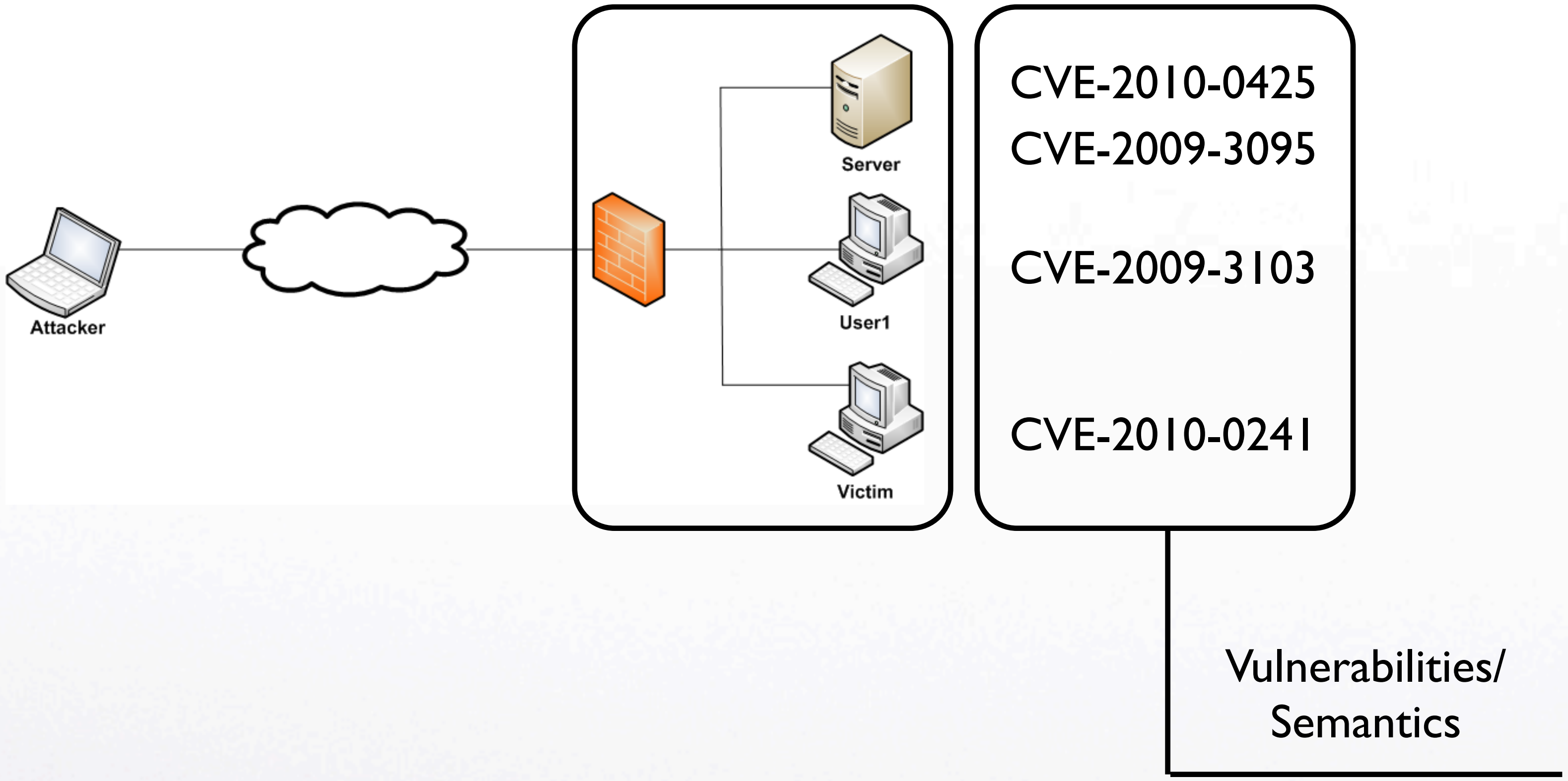
Policy: no host from External should gain access to host Victim

The Attack Graph Generation Process (2)

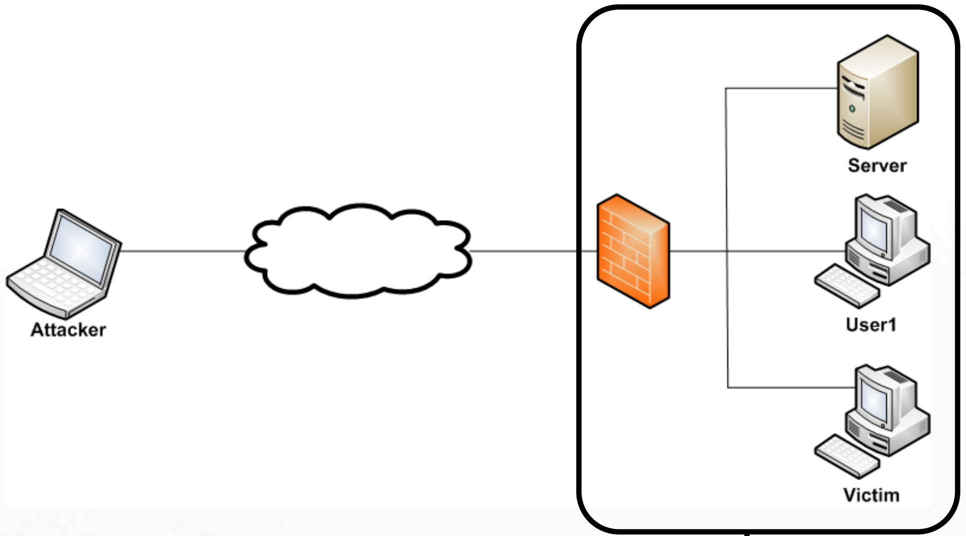


CVE-2010-0425
CVE-2009-3095
CVE-2009-3103
CVE-2010-0241

The Attack Graph Generation Process (2)



The Attack Graph Generation Process (2)



CVE-2010-0425
CVE-2009-3095

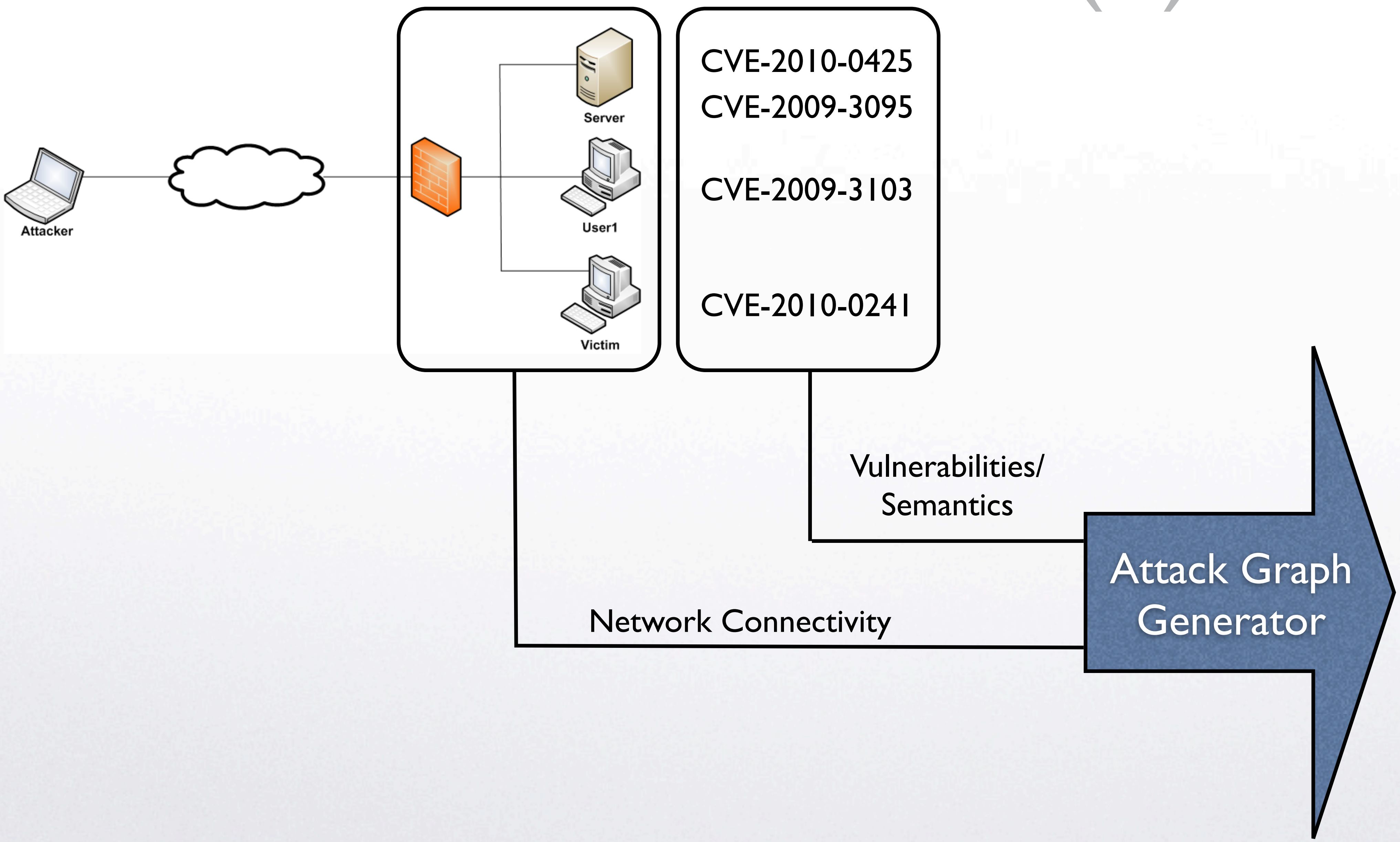
CVE-2009-3103

CVE-2010-0241

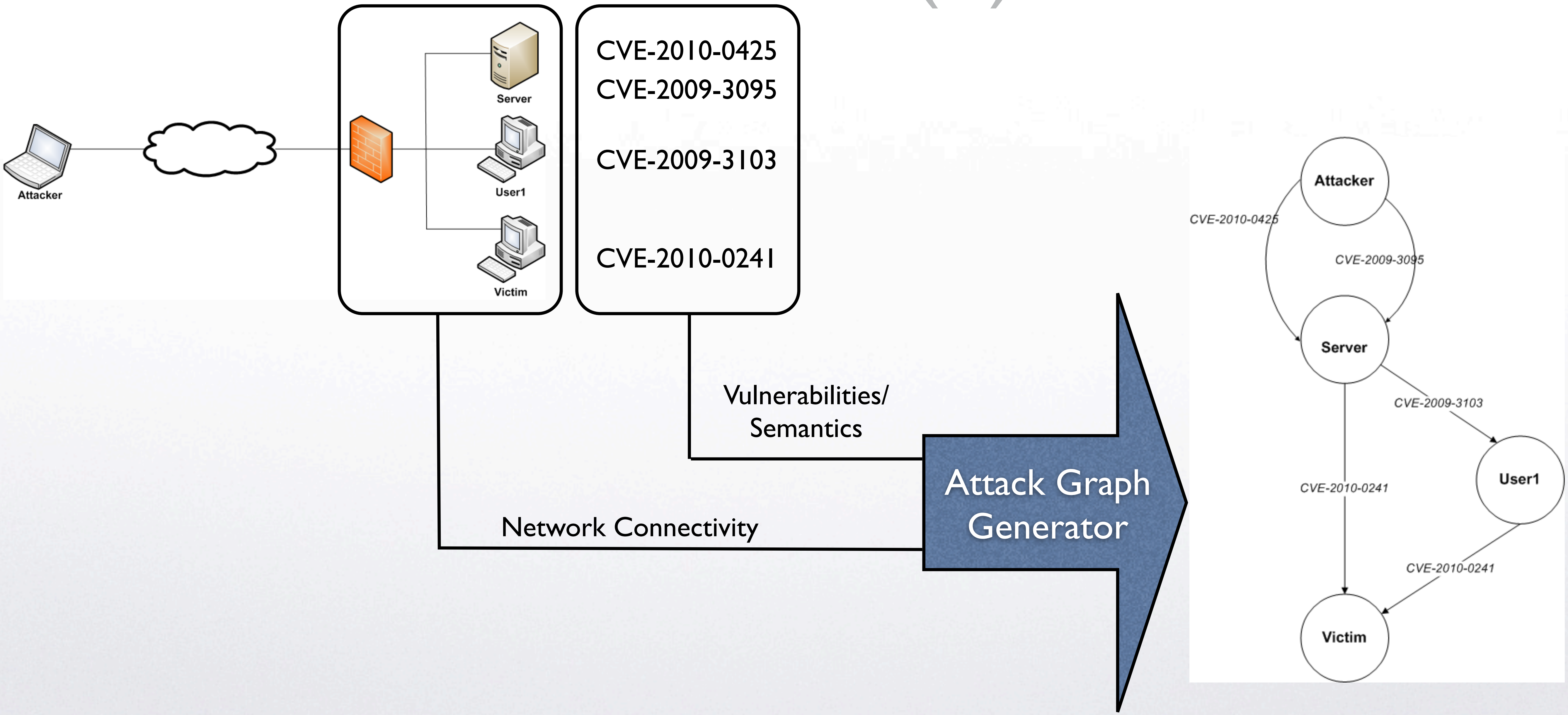
Vulnerabilities/
Semantics

Network Connectivity

The Attack Graph Generation Process (2)

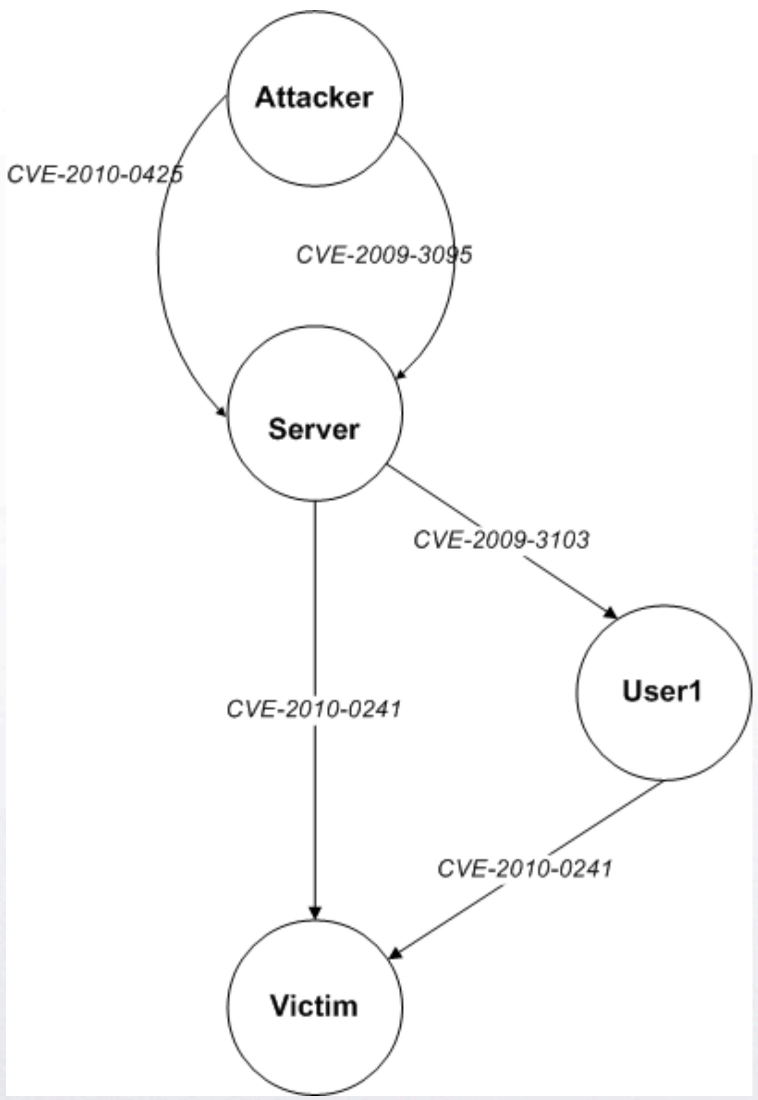


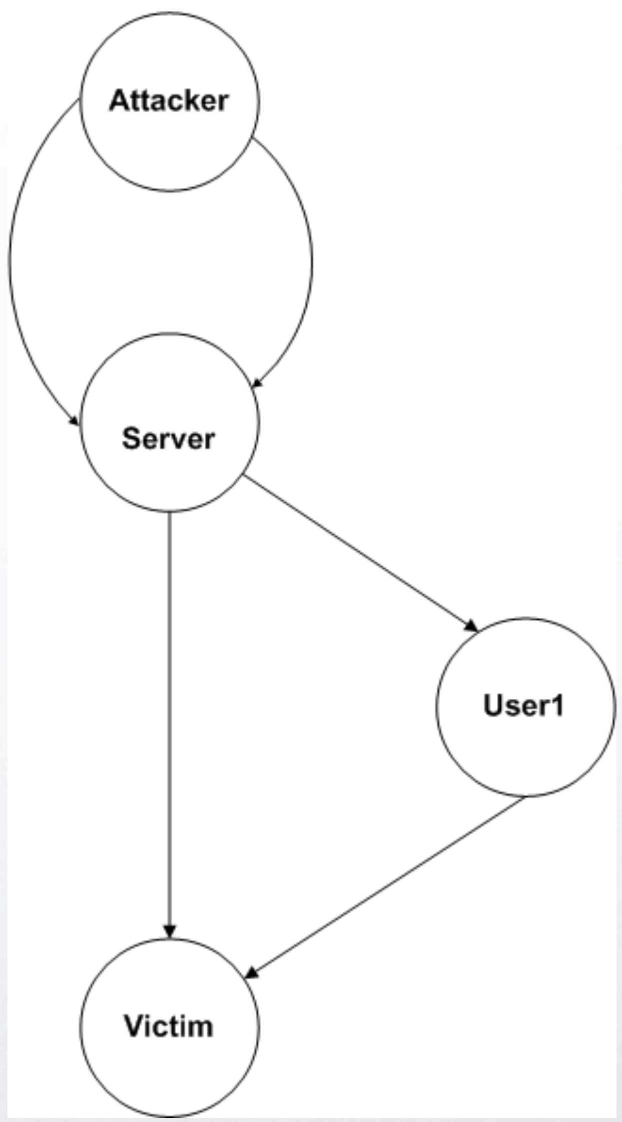
The Attack Graph Generation Process (2)



Condition-oriented Attack Graphs

Labeled-edge attack graph





Label-free-edge attack graph

Overview

- Modeling Attack Path Complexity
- Aggregating Attack Graph-based Security Metrics When Comparing Networks
- Providing an Efficient Computation of the Number of Paths Metric
- Using Multiple Attack Graph-based Security Metrics for Network Hardening

Modeling Attack Path Complexity

A Kolmogorov Complexity-inspired Approach

A Kolmogorov Complexity-inspired Approach

- Kolmogorov Complexity claims that the complexity of a string is equal to the smallest program that can produce this string

A Kolmogorov Complexity-inspired Approach

- Kolmogorov Complexity claims that the complexity of a string is equal to the smallest program that can produce this string
- We use a modified language of Regular Expressions to model attack path complexity

Language for Attack Path Complexity (1)

- Alphabet
  - *A* corresponds to the vulnerabilities found in all attack graphs being considered
- Constants
  - ϵ corresponds to the empty string
  - $v_i \in A$ denotes a vulnerability from one of the attack graphs being considered
  - ∅ corresponds to the empty set

Language for Attack Path Complexity (2)

Let S **and** T **be two strings comprised of characters from** ***A***

- Operations
  - ST evaluates to the concatenation of string S and T
  - () provides priority ordering of evaluation
  - $(S)^+$ the expression S may repeat more than one time but must appear once
  - S^k repeat S *k* times

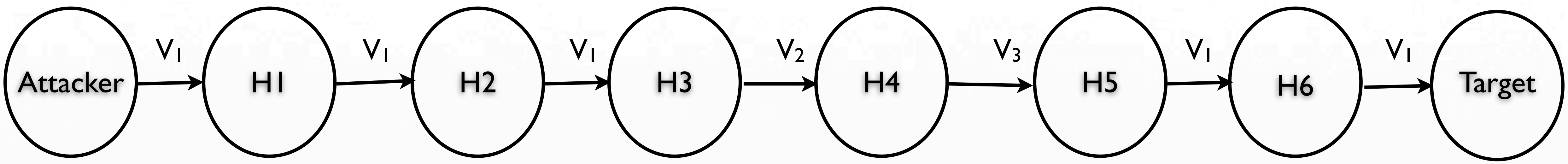
Language for Attack Path Complexity (3)

- Operations

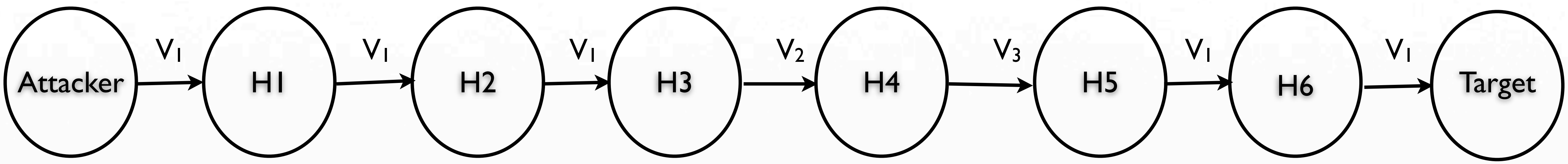
Let E_1 **and** E_2 **be expressions of the language**

  - $E_1{}^{[m]}E_2$ evaluates to inserting E_1 at index m in E_2
  - $E_1{}^{[m_1],[m_2],\ldots[m_n]}E_2$ evaluates to inserting E_1 into indices m_1 through m_n of E_2
  - $E_1{}^{k[m]}E_2$ evaluates to inserting $E_1{}^k$ at index m in E_2
  - $E_1{}^{k,[m]}E_2$ evaluates to concatenating $E_1{}^k$ to E_2 , and inserting E_1 into index m of E_2
  - $E_1{}^{k,[m_1],[m_2],\ldots[m_n]}E_2$ evaluates to concatenating $E_1{}^k$ to E_2 , and inserting E_1 into indices m_1 through m_n of E_2

Kolmogorov Complexity Example

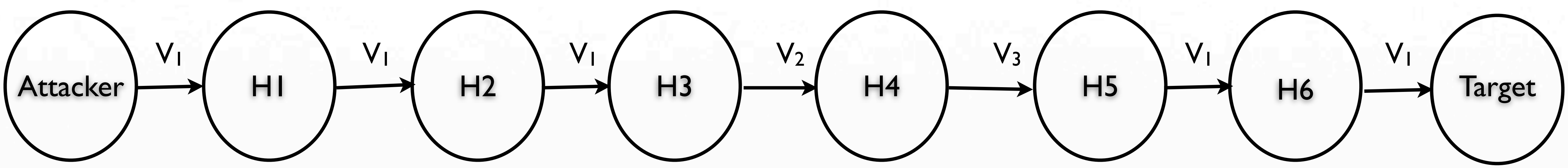


Kolmogorov Complexity Example



A Qualitative Representation: $v_1^{3,[2]} v_2 v_3 v_1$

Kolmogorov Complexity Example



A Qualitative Representation: $v_1^{3,[2]} v_2 v_3 v_1$

The Quantitative Representation: $v_1 v_1 v_1 v_2 v_3 v_1 v_1$

Aggregating Attack Graph-based Security Metrics

Previously Proposed Attack Graph-based Security Metrics

Previously Proposed Attack Graph-based Security Metrics

- Capability Metrics - in terms of attacker capability
  - Number of Paths (Ortalo et al. '99), Weakest Adversary (Pamula et al. '06), Network Compromise Percentage (Lippmann et al. '06)

Previously Proposed Attack Graph-based Security Metrics

- Capability Metrics - in terms of attacker capability
  - Number of Paths (Ortalo et al. ’99), Weakest Adversary (Pamula et al. ’06), Network Compromise Percentage (Lippmann et al. ’06)
- Complexity Metrics - in terms of attack effort
  - Shortest Path (Phillips & Swiler ’98), Mean of Path Lengths (Li & Vaughn ’06)

A Critical Issue Attack Graph-based Security Metrics Miss

A Critical Issue Attack Graph-based Security Metrics Miss

- Security is a *multidimensional* entity

A Critical Issue Attack Graph-based Security Metrics Miss

- Security is a *multidimensional* entity
- All proposed security metrics are *unidimensional*

A Critical Issue Attack Graph-based Security Metrics Miss

- Security is a *multidimensional* entity
- All proposed security metrics are *unidimensional*
- Our approach for comparing 2 networks

A Critical Issue Attack Graph-based Security Metrics Miss

- Security is a *multidimensional* entity
- All proposed security metrics are *unidimensional*
- Our approach for comparing 2 networks
  - Combine metrics measuring distinct attributes of network security

A Critical Issue Attack Graph-based Security Metrics Miss

- Security is a *multidimensional* entity
- All proposed security metrics are *unidimensional*
- Our approach for comparing 2 networks
  - Combine metrics measuring distinct attributes of network security
  - Resolve conflicts by measuring *relevant subsets of attack paths*

Assistive Metrics

- Mean of Path Lengths (MPL)
- *Standard Deviation of Path Lengths (SDPL)*
- *Median of Path Lengths (MePL)*
- *Mode of Path Lengths (MoPL)*

Decision Metrics

- *K-step Capability Accumulation (KCA)*
- *Normalized Mean of Path Lengths (NMPL)*
- Shortest Path (SP), Number of Paths (NP), Network Compromise Percentage (NCP), Weakest Adversary (WA)

K-step Capability Accumulation Metric

$$Cap_h(G) = \cup_h capabilities(n)$$

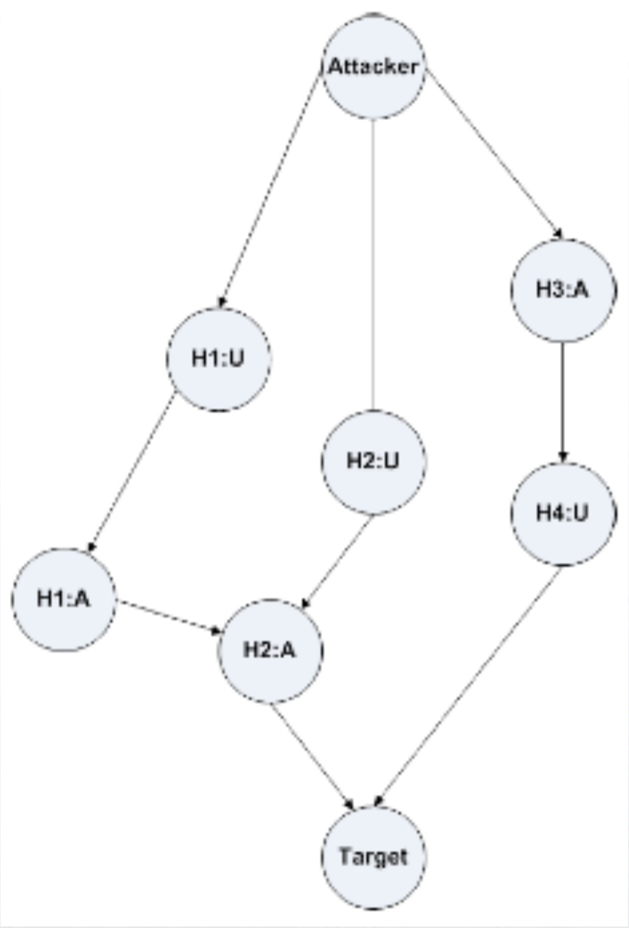
K-step Capability Accumulation Metric

$$Cap_h(G) = \cup_h capabilities(n)$$

$$KCA_k(G) = \cup_{i=0}^{k} Cap_i(G)$$

K-step Capability Accumulation Metric

G_1

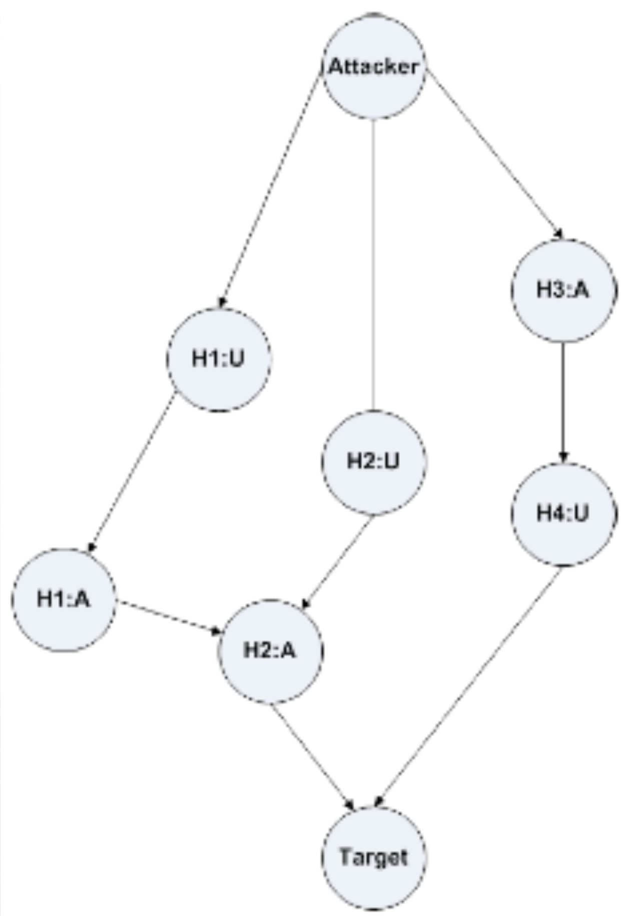


$$Cap_h(G) = \cup_h capabilities(n)$$

$$KCA_k(G) = \cup_{i=0}^{k} Cap_i(G)$$

K-step Capability Accumulation Metric

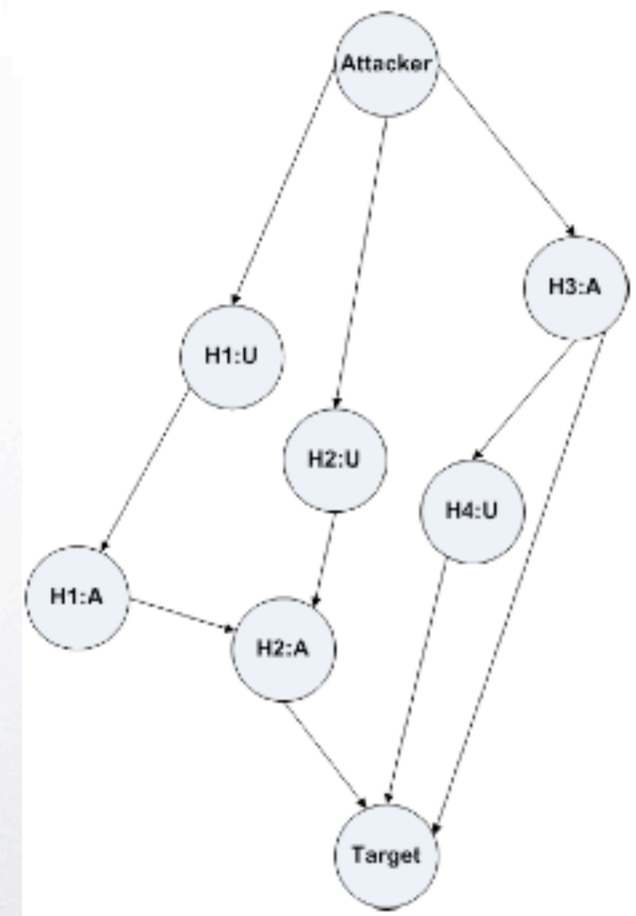
G_1



$$Cap_h(G) = \cup_h capabilities(n)$$

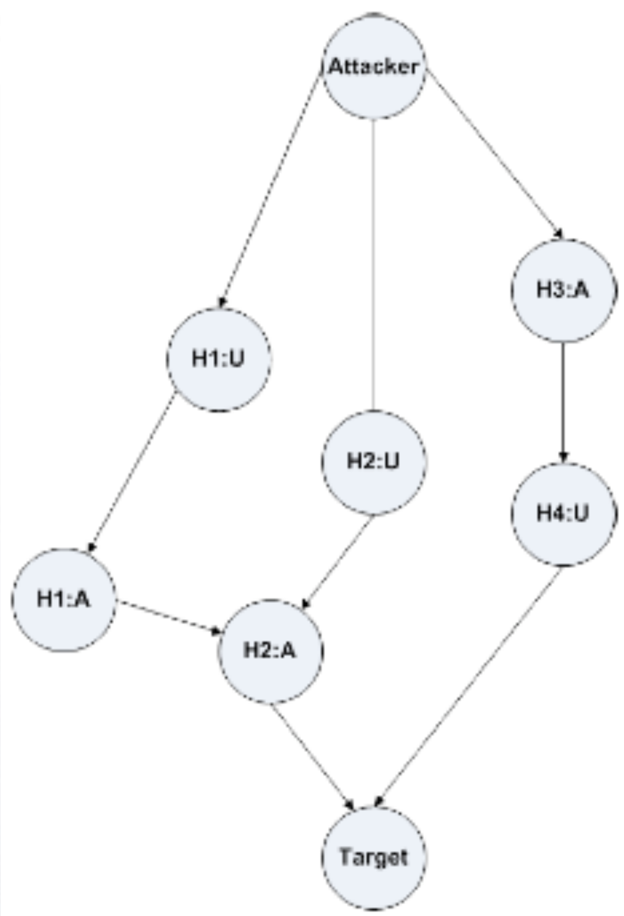
$$KCA_k(G) = \cup_{i=0}^{k} Cap_i(G)$$

G_2



K-step Capability Accumulation Metric

G_1

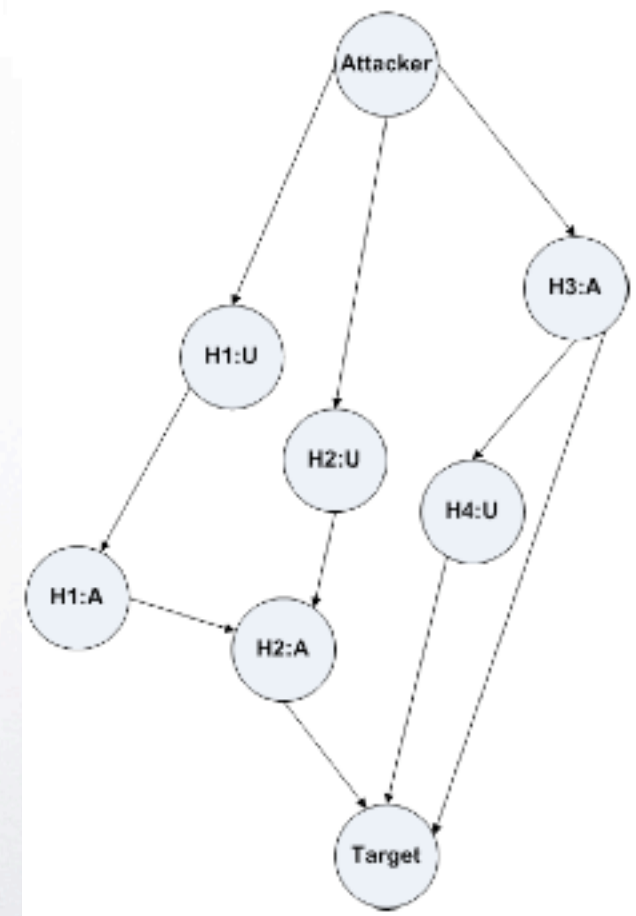


$$Cap_h(G) = \cup_h capabilities(n)$$

$$KCA_k(G) = \cup_{i=0}^{k} Cap_i(G)$$

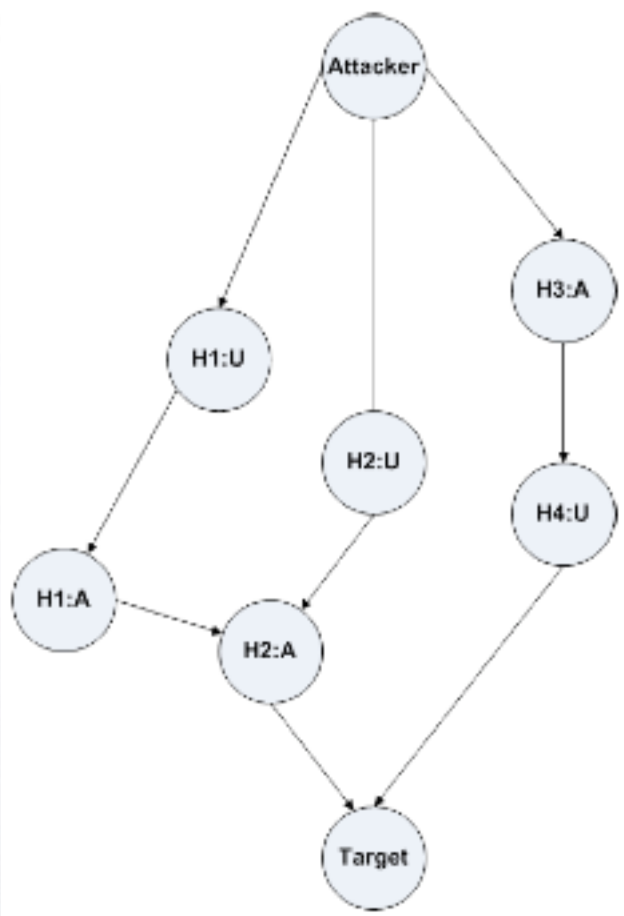
$$KCA_1(G_1) = KCA_1(G_2)$$

G_2



K-step Capability Accumulation Metric

G$_1$



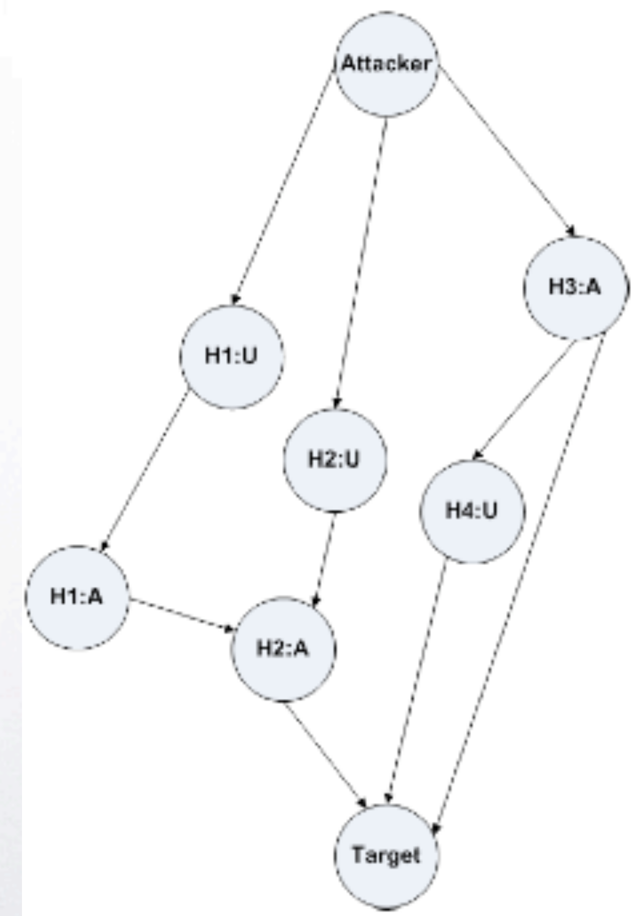
$$Cap_h(G) = \cup_h capabilities(n)$$

$$KCA_k(G) = \cup_{i=0}^{k} Cap_i(G)$$

$$KCA_1(G_1) = KCA_1(G_2)$$

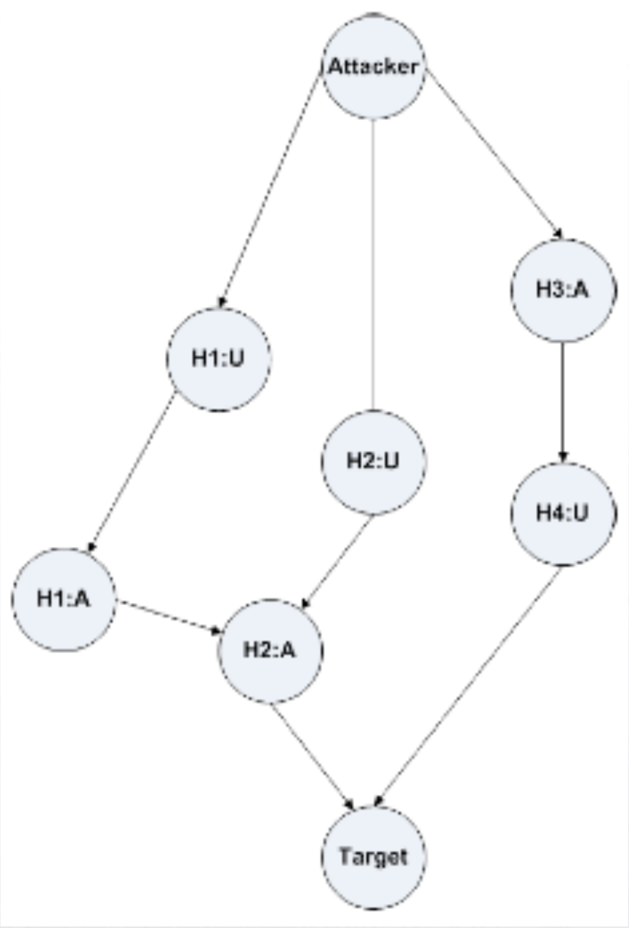
$$KCA_2(G_1) < KCA_2(G_2)$$

G$_2$



K-step Capability Accumulation Metric

G_1



$$Cap_h(G) = \cup_h capabilities(n)$$

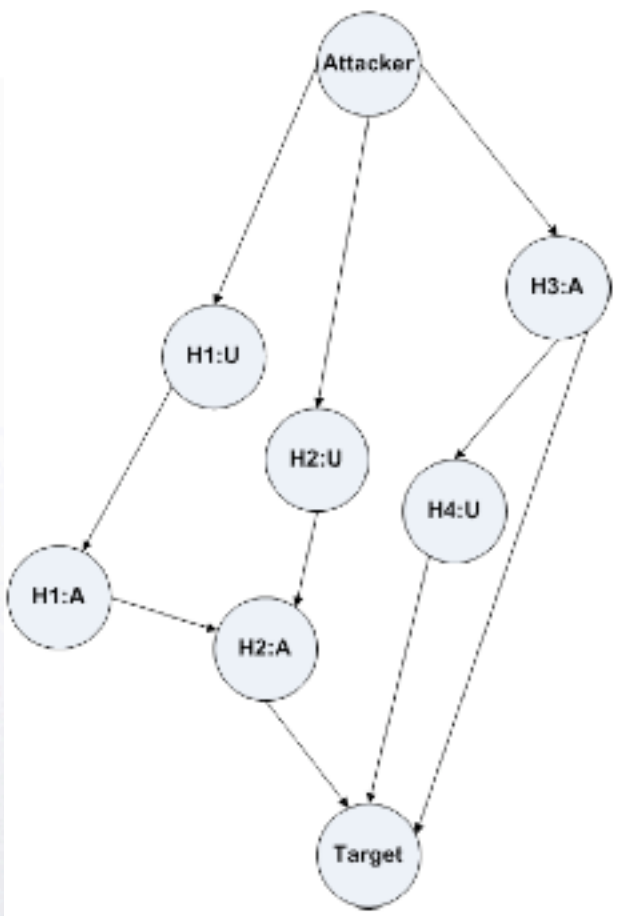
$$KCA_k(G) = \cup_{i=0}^{k} Cap_i(G)$$

$KCA_1(G_1) = KCA_1(G_2)$

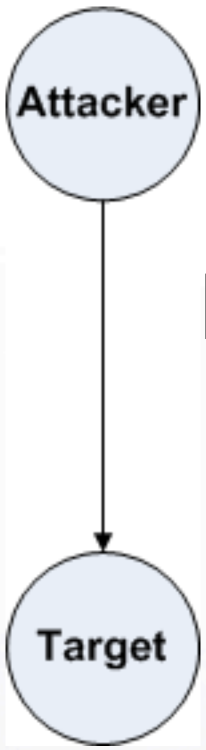
$KCA_2(G_1) < KCA_2(G_2)$

G_1 is more secure than G_2

G_2

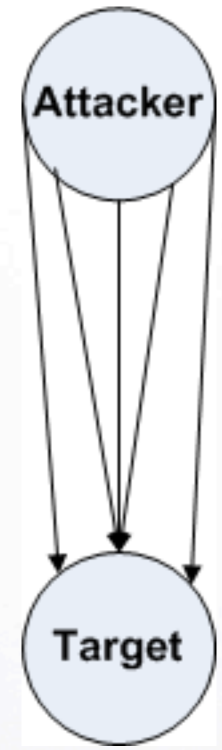


NMPL: A Problem with MPL



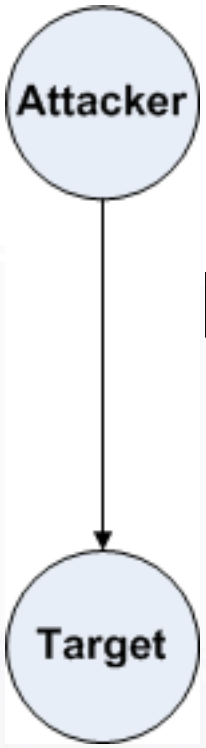
G_1

MPL claims G_1 & G_2 are equal



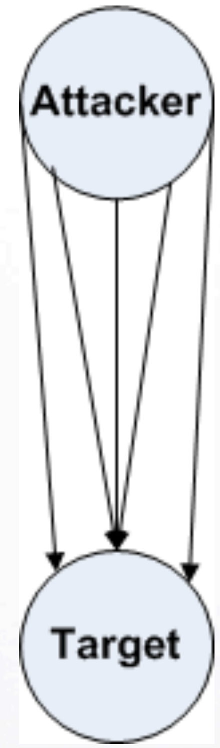
G_2

NMPL: A Problem with MPL



G_1

MPL claims G_1 & G_2 are equal



G_2

NMPL(G_1) = 1 edge and NMPL(G_2) = 0.2 edges

Thus, NMPL claims G_1 is more secure

Aggregation Algorithm (I)

```
for each decision metric m_d in M do
    R_d U eval((x, y, m_d) = apply(m_d, G_1, G_2))
end for

if strictly_dominates(R_d), majority_dominates(R_d), or ties(R_d) then
    Done
else
    enlist_assistive_metrics(G_1, G_2, M)
end if
```

We use SP, NP, and NMPL for decision metrics

Aggregation Algorithm (2)

for each m_d in M **do**
    if m_d equals SP **then**
        R_a U eval((x, y, m_d) = apply(m_d, extract(G_1, MoPL), extract(G_2, MoPL))
        R_a U eval((x, y, m_d) = apply(m_d, extract(G_1, SDPL), extract(G_2, SDPL))
    else if m_d equals NP **then**
        MePL' = min(MePL(G_1), MePL(G_2))
        R_a U eval((x, y, m_d) = apply(m_d, extract(G_1, MePL'), extract(G_2, MePL'))
        R_a U eval((x, y, m_d) = apply(m_d, extract(G_1, SDPL), extract(G_2, SDPL))
    else if m_d equals NMPL **then**
        MePL' = min(MePL(G_1), MePL(G_2))
        R_a U eval((x, y, m_d) = apply(m_d, extract(G_1, MePL'),extract(G_2, MePL'))
        R_a U eval((x, y, m_d) = apply(m_d, extract(G_1, MoPL), extract(G_2, MoPL))
        R_a U eval((x, y, m_d) = apply(m_d, extract(G_1, SDPL),extract(G_2, SDPL))
    end if
end for

if strictly_dominates(R_a), majority_dominates(R_a), or ties(R_a) **then**
    Done **else** Undecided
end if

Assumptions for Algorithm Evaluation

- The number of paths in the attack graph vary more in value than attack path length values
  - Number of paths range: 1 - 2000
  - Attack path lengths range: 1 - 50

Algorithm Evaluation

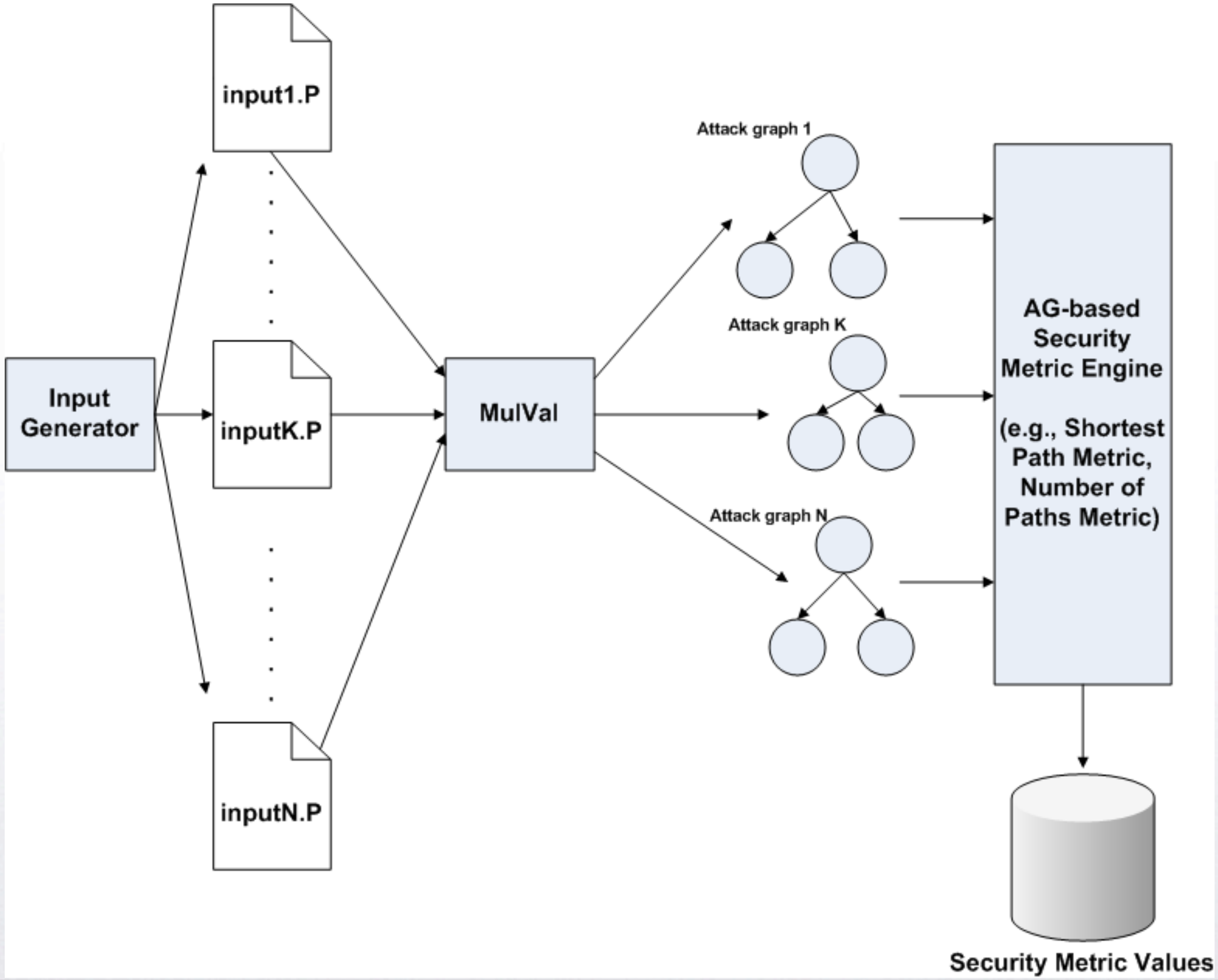
	SP, NP	SP, NMPL	NP, NMPL	SP, NP, NMPL
% Decided	**48.4**	**78**	**99.9**	**99.9**
% Strictly Dominated	4	4	99	4
% Majority Dominated	0	0	0	95
% Equal	0.4	0	0	0
% Strictly Dominated+	10	10	0.1	0.1
% Majority Dominated+	34	64	0.8	0.8
% Equal+	0	0	0	0

+ = enlisting the use of assistive metrics

Generated two disjoint sets of 1000 attack graphs each: 1 million comparisons

Providing an Efficient Computation for the Number of Paths Metric

Experiment Setup



Extracted Equation for Number of Paths Metric on a Flat Network

$$NP(G_t) = \begin{cases} v_t & \text{if } t = 1, \\ v_t NP(G_{t-1}) + NP(G_{t-1}) & t > 1. \end{cases}$$

Extracted Equation for Number of Paths Metric on a Flat Network

$$NP(G_t) = \begin{cases} v_t & \text{if } t = 1, \\ v_t NP(G_{t-1}) + NP(G_{t-1}) & t > 1. \end{cases}$$

A Deterministic Version

When $v_t = c$, $NP(G_t) = c(c+1)^{t-1}$ for $t \geq 1$

Practical Issue

Practical Issue

- 15 host network, 1 target, single remotely exploitable vulnerability on each host

Practical Issue

- 15 host network, 1 target, single remotely exploitable vulnerability on each host
- MulVal on Linux Kernel Version 2.6.32.3, Intel x86 64-bit Architecture, 3GHz CPU, 4GB RAM

Practical Issue

- 15 host network, 1 target, single remotely exploitable vulnerability on each host
- MulVal on Linux Kernel Version 2.6.32.3, Intel x86 64-bit Architecture, 3GHz CPU, 4GB RAM
- In THREE HOURS of computation, NO attack graph was generated

Practical Issue

- 15 host network, 1 target, single remotely exploitable vulnerability on each host
- MulVal on Linux Kernel Version 2.6.32.3, Intel x86 64-bit Architecture, 3GHz CPU, 4GB RAM
- In THREE HOURS of computation, NO attack graph was generated

$$\mathbf{NP(G) = 2^{14}}$$

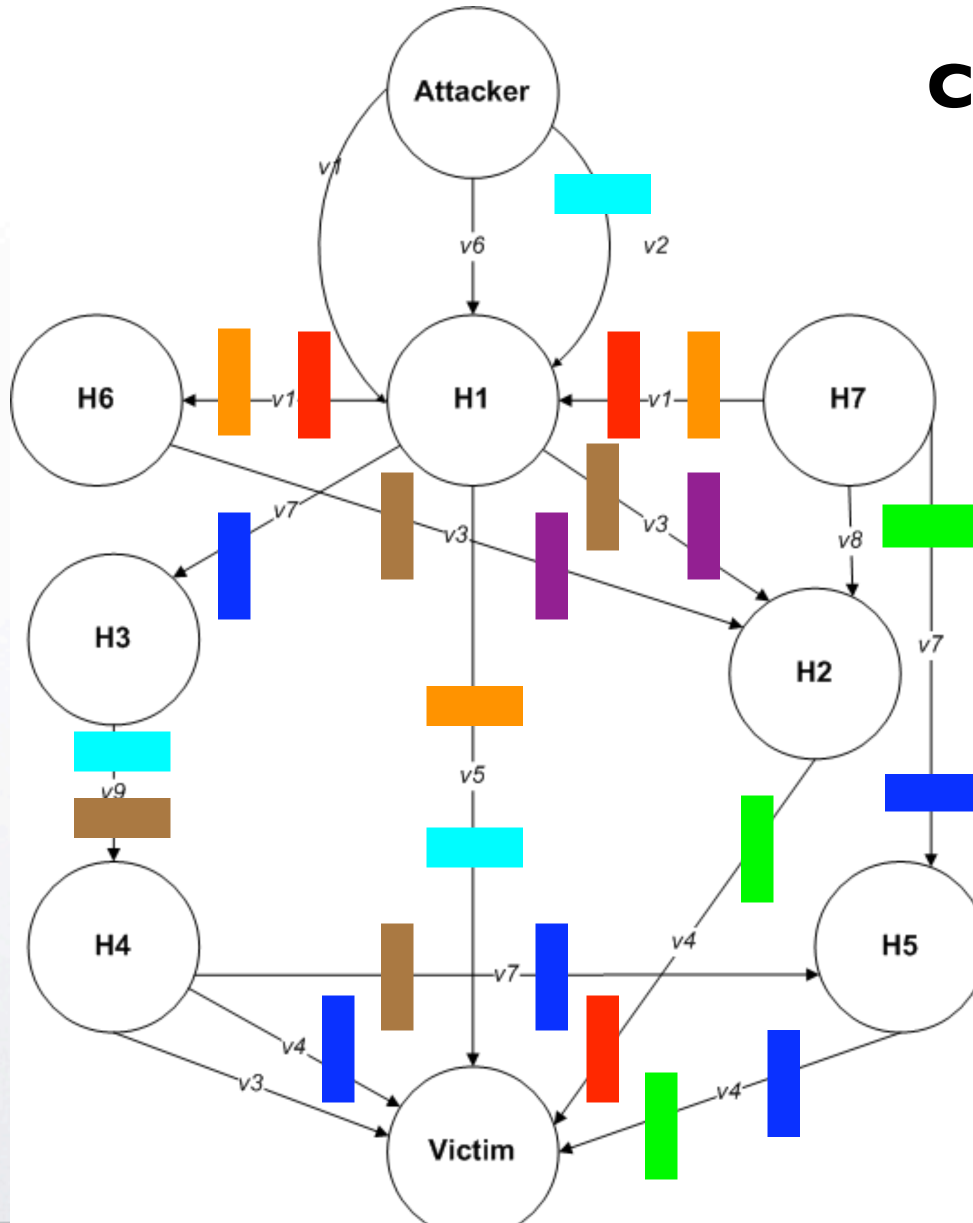
Using Multiple Metrics for Network Hardening

Network Hardening

Network Hardening

- The Goal
  - Choose some subset of possible countermeasures to implement that will provide optimal protection to the network

A Reason Why Network Hardening Can Be Difficult?



Countermeasures

- C1
- C2
- C3
- C4
- C5
- C6
- C7

Previous Approaches

Previous Approaches

- Eliminate all vulnerabilities

Previous Approaches

- Eliminate all vulnerabilities
  - Not always practical

Previous Approaches

- Eliminate all vulnerabilities
  - Not always practical
- Remove key vulnerabilities

Previous Approaches

- Eliminate all vulnerabilities
  - Not always practical
- Remove key vulnerabilities
  - Jha et al., “Two formal analysis on attack graphs” 2002

Previous Approaches

- Eliminate all vulnerabilities
  - Not always practical
- Remove key vulnerabilities
  - Jha et al., "Two formal analysis on attack graphs" 2002
  - Noel et al., "Efficient Minimum-cost hardening via exploit dependency graphs" 2003

Previous Approaches

- Eliminate all vulnerabilities
  - Not always practical
- Remove key vulnerabilities
  - Jha et al., "Two formal analysis on attack graphs" 2002
  - Noel et al., "Efficient Minimum-cost hardening via exploit dependency graphs" 2003
- Use a network security metric

Previous Approaches

- Eliminate all vulnerabilities
    - Not always practical
- Remove key vulnerabilities
    - Jha et al., "Two formal analysis on attack graphs" 2002
    - Noel et al., "Efficient Minimum-cost hardening via exploit dependency graphs" 2003
- Use a network security metric
    - Phillips and Swiler, "A graph-based approach for network vulnerability analysis" 1998

Previous Approaches

- Eliminate all vulnerabilities
  - Not always practical
- Remove key vulnerabilities
  - Jha et al., "Two formal analysis on attack graphs" 2002
  - Noel et al., "Efficient Minimum-cost hardening via exploit dependency graphs" 2003
- Use a network security metric
  - Phillips and Swiler, "A graph-based approach for network vulnerability analysis" 1998
  - Lippmann et al., "Validating and restoring defense in depth using attack graphs" 2006

Our Approach

- Determine budget
- Determine attack graph-based security metrics of interest
- Generate attack graph
- Determine the cost of implementing each countermeasure noting vulnerabilities each mitigates
- Apply Dynamic Programming (DP) algorithm

Relevant DP Algorithm Variables

- Countermeasures are labeled 1 to N
- Each countermeasure (j) has a cost (q_j) and security benefit (m_j)

R_l^j = maximum security possible with x $\subseteq$ {1,2,3,...,j} with a cost equal exactly to l.

$$R_l^j = \begin{cases} R_l^{j-1} & \text{if } q_j > l; \\ max\{R_l^{j-1}, R_{l-q_j}^{j-1} + m_j\} & \text{otherwise.} \end{cases}$$

Maximizing Multiple Metrics

Maximizing Multiple Metrics

- Aggregate Objective Function

Maximizing Multiple Metrics

- Aggregate Objective Function
  - Translate each metric such that:

Maximizing Multiple Metrics

- Aggregate Objective Function
  - Translate each metric such that:
    - each metric is on the same scale

Maximizing Multiple Metrics

- Aggregate Objective Function
  - Translate each metric such that:
    - each metric is on the same scale
    - an increasing value = security improvement

Maximizing Multiple Metrics

- Aggregate Objective Function
  - Translate each metric such that:
    - each metric is on the same scale
    - an increasing value = security improvement
    - an decreasing value = security degradation

Metric Translations

- $SP(G)_r = SP(G)/maxLength(G)$
- $NP(G)_r = NP(G)^{-1}$
- $NMPL(G)_r = NMPL(G)/(maxLength(G)NP(G)_r)$
- $NCP(G)_r = 1 - (NCP(G)/100)$
- $WA(G)_r = weakestSet(C)/|C|$, where C is the set of all attacker attributes
- $KCA(G)_r = 1 - attained(B)/|B|$, where B is the set of all network capabilities

Using Metric Translations

$$R_l^j = \begin{cases} R_l^{j-1} & \text{if } q_j > l; \\ max\{R_l^{j-1}, R_{l-q_j}^{j-1} + m_j\} & \text{otherwise.} \end{cases}$$

$$m_j = w_1 SP(G_{l-q_j}^{j-1})_r + w_2 NP(G_{l-q_j}^{j-1})_r + w_3 NMPL(G_{l-q_j}^{j-1})_r + w_4 NCP(G_{l-q_j}^{j-1})_r + w_5 WA(G_{l-q_j}^{j-1})_r + w_6 KCA(G_{l-q_j}^{j-1})_r$$

Using Metric Translations

$$R_l^j = \begin{cases} R_l^{j-1} & \text{if } q_j > l; \\ max\{R_l^{j-1}, R_{l-q_j}^{j-1} + m_j\} & \text{otherwise.} \end{cases}$$

$$m_j = w_1 SP(G_{l-q_j}^{j-1})_r + w_2 NP(G_{l-q_j}^{j-1})_r + w_3 NMPL(G_{l-q_j}^{j-1})_r + w_4 NCP(G_{l-q_j}^{j-1})_r + w_5 WA(G_{l-q_j}^{j-1})_r + w_6 KCA(G_{l-q_j}^{j-1})_r$$

How should they be weighted?

Thank You. Questions?