Network security (Part II): Can we do a better job?

Rattikorn Hewett
NSF SFS Workshop
August 12-16, 2013

Outline
• State of the practices
• Drawbacks and Issues
• A proposed alternative
How can I secure this network?

**State of the practices**

1) **Admission Control**

- **Authentication**
- **Authentication**
- **Authentication**

Verifying the identification of authorized users

2) **Data Control**

- **Encryption**

Encryption/Decryption of data to be transmitted
State of the practices

3) Infection Control

Virus protection, virus removal, and infection containment

4) Security Policy

Firewall policy to protect unauthorized requests from outside the network

Common IT Security Setup

Where is the weakness of this network to hack into?

Secure enough?
State of the practices

Where is the weakness of this network to hack into?

What about IDS to detect intrusion?

Network Administrator

State of the practices

4) IDS (Intrusion Detection System)

IDS monitors network activities and alerts when attack patterns are detected

Outline

• State of the practices
• Drawbacks and Issues
• A proposed alternative

IDS

I will outsmart IDS with new tricks

Attacker

Encryption

Attacker

IDM 

Authentication

Anti-Virus

Network

Administrator

Whitacre College of Engineering
Recaps current practices & drawbacks

- Admission control, e.g., authentication
- Data control, e.g., encryption
- Infection control, e.g., anti-virus, virus removal/containment
- Security policy, e.g., firewalls, RBAC (role-based access control)
  - Most defend attack at entering points or prevent non-targeted spreading
  - What about targeted attacks in the network?
- Intrusion detection system (IDS)
  - Can’t prevent attacks
  - Can’t detect unfamiliar attacks
  - Requires resource for continuous monitoring

Other Issues ...

- Computer networks are unavoidably vulnerable as long as they have to provide services
  
  **Network Vulnerabilities**

  - Exploitable errors in Network Configurations
  - Implementation of Software Services
    - Ports & services enabled
    - Apache Chunked-Code on Apache web servers
    - Buffer overflow on Windows XP SP2 operating environments
    - TNS Listener on Oracle software for database servers

Network Security Issues

- Computer networks are vulnerable

  **Apache Chunked-Code Buffer-Overflow**
  Apache httpd version 1.3 through 1.3.24 allows remote attackers to cause a denial of service via encoded HTTP or encoded POST requests.

  **Wu-ftpd SockOverflow**
  Wu-ftpd restricted-gid!

  **CVE 2002-0392**
  Common Vulnerability & Exposure
Network Security Issues

- Computer networks are vulnerable
- Commercial scanners can only detect network vulnerabilities at individual points

Network Security: Issues

- Computer networks are vulnerable
- Commercial scanners can only detect network vulnerabilities at individual points
- Perfectly secure isolated services do not guarantee secure network of combined services

Outline

- Current state of the practices
- Issues and drawbacks
- A proposed alternative
A preventative approach

Idea:
- Pre-determine all possible attacks from network vulnerabilities
- Use results to determine appropriate actions

Network
- Vulnerabilities
- Configurations
- Security Policy

Security Model Generation
- Attack Model: all possible chains of exploits (or exploitable vulnerabilities)
- Prioritize critical path
- Select appropriate counter measures

Model Analysis
- Use results to determine appropriate actions
- Prioritize critical path
- Select appropriate counter measures

Goal:
To generate all possible attacks from network vulnerabilities

- Identify vulnerabilities of each computer in the network using a vulnerability scanner (e.g., Nessus, SAINT, OpenVAS)
- Apply all exploitable vulnerabilities for each attack state

Example of Simple Network
- Scan the vulnerabilities
- \[\text{Table: Vulnerabilities/Exploits} \]

Vulnerability | Exploit | Value
--- | --- | ---
R | Access +1 | Access +2
D | Access +1 | Access +1
\[\text{Table: Tiers, Tiers Vulnerability} \]

T | Tiers
--- | ---
Oracle (Insecure) | 2
Tiers vulnerable

W | Tiers
--- | ---
Oracle (Insecure) | 2
Example of a simple network

Can you finish the rest?

Exploit t1

Not exploitable

Host t2, access = 2

Host w, access = 1

Complete Attack Model

Goal: root access of a database server

Attack Model shows all possible attack paths

A preventative approach

Idea:

- Pre-determine all possible attacks from network vulnerabilities
- Use results to determine appropriate actions

Network
- Vulnerabilities
- Configurations
- Security Policy

Security Model

Generation

Model Analysis

Attack Model: all possible chains of exploits (or exploitable vulnerabilities)
Why model analysis? - Example

How can we prevent attack to gain root access at IP2?

\[ v_3 = \text{CVE-2004-0148} \]

"wu-ftpd 2.6.2 and earlier, with the restricted-gid option enabled, allows local users to bypass access restrictions by changing the protections to prevent access to their home directory, which causes wu-ftpd to use the root directory instead."

Counter-measure
1. Upgrade wu-ftpd to version > 2.6.2, OR
2. Replace wu-ftpd with other ftpd-service, OR
3. Stop providing ftpd-service at IP2

Issues
- The resulting attack models are huge even for a small network

How do we effectively analyze the huge attack model?
**Attack Model Analysis**

To extract useful information from security model to protect the network:

**Visualization**
- Group similar nodes for display [Noel & Jajodia, 05]
- Manual, time-consuming
- Non-systematic

**Graph-based**
- Minimization analysis to block paths [Jha et al., 02]
- Automatic
- Limited to specific models

**Markov model-based**
- Estimate likelihood of attack [Sheyner et al., 02; Mehta et al., 06; PageRank]
- Handle cyclic models

**Our approach**
- Exploit-based analysis
- Use knowledge about exploitability

**Exploit-based Analysis**

Prioritizes attack points in an attack model based on the ease in exploiting their vulnerabilities:

Easy to exploit ➔ High exploitability ➔ High priority (for fixing)

**Approach**

Estimate a probability distribution of intrusion for each attack state:
- To obtain its relative chance of being attacked using the knowledge about exploitability

**Exploitability**

- **Atomic level**
  - Exploitability of each vulnerability

  Access Vector × Access Complexity × Authentication
  
  E.g., remote, local  ➔ E.g., low efforts to exploit  ➔ E.g., no or single authentication
Exploitability

- **Atomic level**
  - Exploitability of each vulnerability (degrees 1→10)

- **Global level**
  - Exploitability of attack states in the network topology
    - Based on Markov Model (Applied to PageRank)

Markov Model

- Approximates a probability distribution of dynamic behaviors randomly evolving to a stationary state
  - Define the probability of intrusion of each attack point recursively
    - **Markov Property:**
      - The probability distribution for the future network intrusion only depends on the current states
      - Repeat the computation until no change in the probability distribution approximation
Recurrence Equation

\[ h(u, v) = \text{exploitability of exploits from state } u \text{ to } v \]
\[ r_t(u) = \text{probability of state } u \text{ being attacked at time } t \]
\[ d = \text{probability that attackers continue attacking on a current path} \]

If \( v \) is not an initial state

\[ r_{t+1}(v) = d \sum_{u \in \mathcal{V}} r_t(u) \frac{h(u, v)}{\sum_{u' \in \mathcal{V}} h(u', v)} \]

If \( v \) is an initial state

\[ r_{t+1}(v) = \frac{1}{d} \sum_{u \in \mathcal{V}} r_t(u) \frac{h(u, v)}{\sum_{u' \in \mathcal{V}} h(u', v)} \]

ExploitRank Algorithm

\begin{algorithm}
\caption{ExploitRank}
\begin{algorithmic}[1]
\STATE \textbf{function} \text{ExploitRank}\( (\mathcal{V}, d, h) \)
\STATE \hspace{1em} \textbf{in} \hspace{1em} \text{a set of initial states}
\STATE \hspace{1em} \text{for each } v \in \mathcal{V} \text{ do}
\STATE \hspace{2em} \text{\textit{ExploitRank}\( (\mathcal{V}, d, h) \)
\STATE \hspace{2em} \hspace{1em} \text{output } r_t(v)
\STATE \hspace{2em} \text{end for}
\STATE \hspace{1em} end for
\STATE \hspace{0em} \text{repeat}
\STATE \hspace{1em} \hspace{1em} \text{\textit{ExploitRank}\( (\mathcal{V}, d, h) \)
\STATE \hspace{1em} \hspace{2em} \text{output } r_t(v)
\STATE \hspace{1em} \hspace{0em} \text{end for}
\STATE \hspace{0em} \text{until } r_t(v) = r_{t+1}(v) \forall v \in \mathcal{V}
\STATE \hspace{1em} \text{return } r_t(v) \forall v \in \mathcal{V}
\STATE \textbf{end function}
\end{algorithmic}
\end{algorithm}
A simple Illustration

Some Comparisons

In Mehta et al.’s approach
- Each node has equal chance to be attacked – no use of the degree of vulnerability exploitability

<table>
<thead>
<tr>
<th>Vulnerability</th>
<th>Exploitability</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVE-2006-0586</td>
<td>9.9</td>
</tr>
<tr>
<td>CVE-2005-0001</td>
<td>4.9</td>
</tr>
<tr>
<td>CVE-2006-0541</td>
<td>3.9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Node</th>
<th>Likelihood</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_1</td>
<td>0.43097</td>
</tr>
<tr>
<td>S_2</td>
<td>0.14876</td>
</tr>
<tr>
<td>S_3</td>
<td>0.32414</td>
</tr>
<tr>
<td>S_4</td>
<td>0.37017</td>
</tr>
</tbody>
</table>

More complex attack model
Conclusions

• Current state of security practices help guard against
  • illegitimate network entry access
  • network intrusion and network infection
  BUT attackers can still attack the network by exploiting network vulnerabilities (due to configuration or software errors)
• One remedy is to aim to prevent all possible attacks from these vulnerabilities (not just entry points)
• We give an example of how
  • Attack model can be automatically constructed and used for security management
• Scalability is a concern that requires further work

References

• Schiffman, Cisco CAIG, A Complete Guide to the Common Vulnerability Scoring System (CVSS), Forum Incident Response and Security Teams (http://www.first.org/)