A Mission-Centric Framework for Cyber Situational Awareness

Sushil Jajodia
George Mason University

SECRYPT 2012
Motivation

- An ever increasing number of critical applications and services rely on Information Technology infrastructures
  - Increased risk of cyber attacks
  - Increased negative impact of cyber attacks
- Attackers can exploit network configurations and vulnerabilities (both known and unknown) to incrementally penetrate a network and compromise critical systems
  - Manual analysis is labor-intensive and error-prone
  - Vulnerabilities are often interdependent, making traditional point-wise vulnerability analysis ineffective
  - Services and machines on a network are interdependent
- Need for tools that provide analysts with a “big picture” of the cyber situation
Current situation. Is there any ongoing attack? If yes, where is the attacker?

Impact. How is the attack impacting the enterprise or mission? Can we assess the damage?

Evolution. How is the situation evolving? Can we track all the steps of an attack?

Behavior. How are the attackers expected to behave? What are their strategies?

Forensics. How did the attacker create the current situation? What was he trying to achieve?

Information. What information sources can we rely upon? Can we assess their quality?

Prediction. Can we predict plausible futures of the current situation?

Scalability. How can we ensure that solutions scale well for large networks?
CSA Framework Architecture

- Vulnerability Databases
  - NVD
  - CVE
  - OSVD

- Topological Vulnerability Analysis
  - Cauldron
  - Switchwall

- Monitored Network

- Index & Data Structures
  - NSDMiner

- Stochastic Attack Models

- Dependency Analysis
  - Generalized Dependency Graphs

- Scenario Analysis & Visualization
  - Unexplained Activities
  - Adversarial modeling
  - Network Hardening

- Analyst

- Graph Processing and Indexing

Alerts/Sensory Data
Network Vulnerability Analysis

Attack Graphs
Motivation for Network Vulnerability Analysis

- Current security measures largely independent
  - Generate isolated vulnerability data
  - Manual process requiring high expertise
  - Administrators must make sense of this, then respond appropriately and quickly
  - Error prone due to complexity, volume, and frequent changes in security data and network configurations

- Establishing and understanding the context is a mandatory and necessary first step for successful cyber incident response
Vulnerability Scanner

Legend:
- Symbol Count: Description

Oracle DB Server
W2K Web Server
Firewall

Server LAN
160 Vulns
158 Vulns

Client LAN
107 Vulns
60 Vulns

External Attacker
47 Vulns

Attack Target

External

DMZ

W2K Web Server
W2K Exchange Server
Linux Mail Server

WinXP Client
W2K Pro Client

5 Server
3 Router
2 Firewall
2 PC

Oracle DB Server

192.168.1.5

smrsh (supplied by Sendmail) is designed to prevent the execution of commands outside of the restricted environment. However, when files are entered using either double pipes (||) or a mixture of dot and slash characters, a user may be able to bypass the checks performed by smrsh. This can lead to the execution of commands outside of the restricted environment.

Solution: upgrade to the latest version of Sendmail (or at least 6.8.28).
Risk factor: Medium
CVE: CAN-2002-1165
BID: 5845

The remote sendmail server, according to its version number, may be vulnerable to a remote buffer overflow allowing remote users to gain root privileges.

Sendmail versions from 5.79 to 8.12.8 are vulnerable.
Solution: upgrade to Sendmail version 8.12.9 or newer. The use of
--with-sendmailcap or similar might also prevent this vulnerability.
Limitations of Vulnerability Scanners

- Generate overwhelming amount of data
- Example Nessus scan
  - Elapsed time: 00:48:07
  - Total security holes found: 588
  - High severity: 120
  - Low severity: 370
  - Informational: 98
- No indication of how vulnerabilities can be combined
- Can an outside attacker obtain access to the DB server?
- Where does a security administrator start?
Limitations of IDSs

- Generate overwhelming number of alerts
- Many false alerts – normal traffic or failed attacks
- Alerts are isolated
- Incomplete alert information
- No indication of how alerts can be combined
- Where does a security administrator start?
- Is the attacker trying to obtain access to DB server?
- Require extensive human intervention
The reality – security concerns are highly interdependent.

Simply Listing Problems Misses the Big Picture!
Attack Graphs

- An attacker breaks into a network through a chain of exploits where each exploit lays the groundwork for subsequent exploits.
- Chain is called an attack path.
- Set of all possible attack paths form an attack graph.
- Generate attack graphs to mission critical resources.
- Report only those vulnerabilities associated with the attack graphs.
Related Work

- Phillips and Swiler NSPW 1998
- Templeton and Levitt NSPW 2000
- Ritchey and Ammann S&P 2000
- Wing, Jha et al. CSFW 2002
- Ammann et al CCS 2002
- Ou et al. CCS 2006
- Sawilla and Ou ESORICS 2008
Even small networks can yield complex attack graphs!
Network Hardening

Attack graph $G = (E \cup C, R_r \cup R_i)$
- $E$: set of exploits
- $C$: set of conditions
- $C_i \subseteq C$: set of initial conditions
- $R_r$: requires relationship
- $R_i$: implies relationship

**Hardening Solutions**
1) $\{ftp(0,2), ftp(0,1)\}$
2) $\{ftp(0,1), ftp(0,2), sshd(0,1)\}$
Cauldron has Numerous Applications

Network Hardening

Security Metrics

Alarm Correlation And Attack Response

Sensor Placement

Risk Model

Metrics

Scored Risks
Related Publications


Dependency Analysis

Generalized Dependency Graphs
Limitations of Attack Graphs

- Do not encode enough information about the attacker’s behavior
- Do not provide a mechanism to evaluate impact of each attack pattern on the enterprise
- Scalability issues have not been fully addressed - Ideally, attacks must be recognized in real-time
Example of Decision Making

Current Situation: The Mobile App Server has been compromised.

Possible futures:
1) The attacker will exploit the local DB Server with probability 70%.
2) The attacker will exploit the Order Processing Server with probability 30%.

Possible courses of actions:
1) Based on the probability of individual outcomes, we could be tempted to patch the vulnerability that has the highest probability of being exploited next.
2) However, protecting the Local DB Server will not reduce our expected future damage assessment, since the Mobile Order Tracking service is already compromised.
3) Protecting the Order Processing Server would instead guarantee that the other service is not compromised.
Motivation for Dependency Analysis

- A network application usually depends on several other network services to function correctly
  - These network services may depend on other services
- It is critical to know network service dependencies for
  - Fault diagnosis/isolation
  - Cyber situation awareness
  - Response to cyber attacks
- Challenges
  - An enterprise network is usually complex and dynamic
  - Manual analysis is error-prone and impractical
- It is desirable to automatically discover network service dependencies
Generalized Dependency Graphs

**Dependency functions**

\[ f_d(l_1, \ldots, l_n) = \frac{1}{n} \sum_{i=1}^{n} l_i \]

\[ f_s(l_1, \ldots, l_n) = \begin{cases} 1, & \text{if } (\forall i \in [1, n]) l_i = 1 \\ 0, & \text{otherwise} \end{cases} \]

![Diagram showing dependency functions and server states](image)

- **Fully working** (1)
- **Degraded performance**
- **Unusable** (0)

- **Online Shopping**
- **Mobile Order Tracking**
An Introductory Example

- **Web Server**
  - Depends on Authentication Server to authenticate the client
  - Depends on DB Server for data

- **Client**
  - Depends on DNS to resolve Web server’s IP address

---

INFOCOM 2012 Peng Ning

03/29/12
Previous Solutions

- Host-based schemes (e.g., Magpie [OSDI ’04], Pinpoint [NSDI ’04], Macroscope [CoNEXT ’09])
  - Effective
  - Intrusive → host agent/middleware
  - Some require application semantics
  - Not desirable due to the required changes on hosts

- Network-based schemes (Sherlock [SIGCOMM ’06], eXpose [SIGCOMM ’08], Orion [OSDI ’08])
  - Non-intrusive
  - Application independent
  - False positive and false negative are both high
Performance

High false positive rate with nominal detection rate

* From Orion Publication
Our Contribution

- NSDMiner: New approach for automated discovery of network service dependencies
  - Network-based
  - Passive
  - Focused on dependencies on the server side
  - Superior to previous network-based approaches
    - Significant reduction in false positives
    - Higher detection rate
Key observation

- Most outgoing connections that a server depends on happen during serving the request.
NSDMiner – A Timeline View

Client
$\rightarrow$ Web server
$t = 500$

Request
$t = 500.5$

Web server
$t = 501$

Request
$t = 501.2$

Kerberos server
$t = 502$

Reply
$t = 502.3$

Database server

Request

Reply

INFOCOM 2012
NSDMiner Algorithm

- NSDMiner: Analyze network traffic to correlate flows
- Input
  - TCP and UDP flows
    - UDP flow: A sequence of UDP packets between two endpoints where the delay between any two consecutive packets is less than a threshold
      
      \[(\text{StartTime}, \text{EndTime}, \text{Proto}, \text{SrcIP}, \text{SrcPort}, \text{DestIP}, \text{DestPort})\]

- Basic idea
  - Process each flow record in increasing order of \textit{StartTime}
    - Check previous flow records for potential dependencies
    - A previous flow potentially depends on the current flow if
      - If the current flow is from the destination host in the previous flow, and
      - The current flow occurs during the previous flow
Conclusion

- NSDMiner: A simple but effective method to identify local-remote service dependencies
  - Network based
  - Non-intrusive
  - Better performance than existing solutions

- Limitations
  - Rely on network activities
  - Limited to local-remote dependencies

- Future work
  - Handle unknown service clusters
  - Improve detection rate
Related Publication


Adversary Modeling and Scalability

Stochastic Attack Graphs
Detection Algorithm
Incorporating Attacker’s Behavior

- **Stochastic Attack Graphs**
  - Our goal is to incorporate knowledge of them (the attackers) into the attack model.

- **Our assumptions**
  - Different attack paths have different probabilities of being observed.
    - Vulnerabilities that are easier to exploit will be exploited more frequently.
  - There is a lower and upper bound on the time that can elapse between two consecutive exploits of an attack path.
### Occurrences & Probability Computation

When an alert for $V_C$ is received, 2 time units have elapsed since an alert for $V_A$ was received.

An occurrence of $A$ in $O$ is a sequence $O^* = \langle o_1, \ldots, o_k \rangle \subseteq O$.

Independence assumption:

$$\text{prob}(\langle o_1, \ldots, o_k \rangle, A) = \prod_{i \in [1,k-1]} \tau_i(x_i, y_i)$$

2 falls between lower and upper bound.

Probability = 0.3
Index Update/Detection Algorithm

- Algorithm **updateIndex** updates an index when a new alert is received.

  - $o_{new}$
  - 10 33 ip360.534 h_x h_y

- Algorithm **updateIndex** can be used iteratively for processing an entire observation sequence at once (**bulkUpdate**).

  - Check if the $exploit(o_{new})$ is a start node for an attack.
  - Check index tables associated with predecessors of $exploit(o_{new})$.
  - Check if the $exploit(o_{new})$ is an end node.

- Assess marginal damage of $exploit(o_{new})$. 
Under the Time Frame pruning strategy, for each new alert $o_{\text{new}}$, algorithm $updateIndex$ avoids scanning records that cannot be linked to $o_{\text{new}}$.

<table>
<thead>
<tr>
<th>curr</th>
<th>act</th>
<th>$D$</th>
<th>$t_0$</th>
<th>$prob$</th>
<th>$\Delta d$</th>
<th>prev</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td>A₁</td>
<td>2</td>
<td>0.6</td>
<td>12</td>
<td></td>
<td></td>
<td>•</td>
<td>⊥</td>
</tr>
<tr>
<td>A₂</td>
<td>1</td>
<td>0.7</td>
<td>2</td>
<td></td>
<td></td>
<td>•</td>
<td>⊥</td>
</tr>
<tr>
<td>A₁</td>
<td>4</td>
<td>0.8</td>
<td>13</td>
<td></td>
<td></td>
<td>•</td>
<td>⊥</td>
</tr>
<tr>
<td>A₃</td>
<td>7</td>
<td>0.9</td>
<td>8</td>
<td></td>
<td></td>
<td>•</td>
<td>⊥</td>
</tr>
<tr>
<td>A₄</td>
<td>9</td>
<td>0.5</td>
<td>18</td>
<td></td>
<td></td>
<td>•</td>
<td>⊥</td>
</tr>
</tbody>
</table>
Attack Scenario Analysis
Damage Assessment

- Each service/mission has an associated **utility**
  - A service/mission depends on one or more network components
    - If any of them is compromised, the service/mission is affected and its utility is reduced

- The **damage** caused by a cyber attack is proportional to the total loss of utility of services/missions affected by the attack

- For each possible future of the current situation, we can assess damage
  - **Attack graphs** indicate which assets might be directly compromised
  - **Dependency graphs** indicate which assets/services are affected as a consequence
\[ \Delta \text{damage} = \sum_{h \in H} (s_{i-1}(h) - s_i(h)) \cdot u(h) \]
Algorithm `rankFutureScenarios`, predicts possible futures – of length \( k \) or less – of the current situation and assesses their likelihood and marginal damage.

- Predicted scenarios are ranked by a measure of criticality accounting for both probability and marginal damage.

A criticality function can be defined as any function of the form \( f : [0, 1] \times \mathbb{R} \to \mathbb{R} \) that satisfies the following monotonicity axioms:

- \( (\forall \Delta d \in \mathbb{R}) \ p_1 \geq p_2 \Rightarrow f(p_1, \Delta d) \geq f(p_2, \Delta d) \)
- \( (\forall p \in [0, 1]) \ \Delta d_1 \geq \Delta d_2 \Rightarrow f(p, \Delta d_1) \geq f(p, \Delta d_2) \)

In the simplest case, we can estimate the criticality of a future as the product of its probability and marginal damage.
Experimental Results

- Experiments were conducted on both real (786 nodes) and synthetic (up to 300 thousand nodes) attack graphs.
  - In both cases we used the graphs to simulate a number of attack occurrences and generate a stream of 3 million alerts.

- We measured:
  - The time to build the index
  - The consumption of memory
  - The time to compute future scenarios
Real Attack Graph Used in Experiments

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of machines</td>
<td>64</td>
</tr>
<tr>
<td>Total number of exploits</td>
<td>786</td>
</tr>
<tr>
<td>Number of inter-domain edges</td>
<td>182</td>
</tr>
<tr>
<td>Number of edges in fully exploded graph</td>
<td>266,770</td>
</tr>
</tbody>
</table>
The time to build the index increases linearly with the number of alerts.

The algorithm can process between 20 and 30 thousands alerts per second.

There is no significant difference between results on real and synthetic attack graphs.

The size of the graphs does not significantly affect the index building time.
Index Building Time vs. Graph Size

- When the size of the merged graph changes by orders of magnitude, the processing time increases slightly.
- The slight increase can be attributed to the overhead of managing a larger number of tables.
Index Size vs. Graph Size

- Memory occupancy increases linearly with the number of alerts processed.
- Memory occupancy is independent of the size of the graphs.
As $k$ increases, processing time increases exponentially, but becomes stable as $k$ becomes comparable with the length of individual attack patterns.

Processing time is not significantly affected by the size of the graphs when $k$ is small.
Questions?