An Attack Surface Metric

Pratyusa K. Manadhata
Carnegie Mellon University
Context: Security Metrics

Software vendors are spending big on security.

How secure is our software?
Are we better-off than before?

Which is more secure: XP or Vista? Ubuntu or Fedora?

Gauging progress is critical for secure software development. We need measurements and metrics.
We Need Metrics Now!

• A long standing research challenge
  [ACSAC 01, CRA 03, DIMACS 03, CSTB 07]

Toward a Safer and More Secure Cyberspace
  [CSTB 2007]:

“..though many benefits would flow from the invention of good metrics, the challenge in this cybersecurity research area is particularly great, and some very new ideas will be needed if cybersecurity metricians are to make more progress.”
Our Approach: Attack Surface Measurement (ASM)

How can we quantify a software system’s security?

Measure the system’s attack surface
Motivation: ASM is Useful to both Industry and Consumers

A guide in consumers’ decision making process

A tool in the software development lifecycle to improve security
  • design, implementation, testing, deployment, and maintenance
Attack Surface Reduction (ASR) Mitigates Risk

Traditional industry approach: code quality improvement

Software will ship with known and future vulnerabilities

Reduce attack surface to increase the difficulty and decrease the impact of future exploitation
Code Quality and ASR Complement Each Other

- Code Quality:
  - Good
  - Medium
  - Bad

- Security Risk:
  - Low Security Risk
  - Medium Security Risk
  - High Security Risk

- Attack Surface Measurement:
  - Low
  - High

The diagram illustrates the relationship between code quality and attack surface measurement, showing that combinations of low quality and low measurement result in low security risk, while high quality and high measurement result in high security risk.
Inspiration: Relative Attack Surface Quotient for 7 Versions of Windows [HPW03]

Windows Server 2003 is “more secure” than previous versions.
## Linux Attack Surface Measurements

<table>
<thead>
<tr>
<th>Attack Vector</th>
<th>Debian</th>
<th>RH Default</th>
<th>RH Facilities</th>
<th>RH Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open socket</td>
<td>15</td>
<td>12</td>
<td>40</td>
<td>41</td>
</tr>
<tr>
<td>Open RPC endpoint</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Services running as root</td>
<td>21</td>
<td>26</td>
<td>29</td>
<td>30</td>
</tr>
<tr>
<td>Services running as nonroot</td>
<td>3</td>
<td>6</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Setuid root programs</td>
<td>54</td>
<td>54</td>
<td>72</td>
<td>72</td>
</tr>
<tr>
<td>Local user accounts</td>
<td>21</td>
<td>25</td>
<td>33</td>
<td>34</td>
</tr>
<tr>
<td>User id = root accounts</td>
<td>0</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Unpassworded accounts</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Nobody account</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Weak file permission</td>
<td>7</td>
<td>7</td>
<td>21</td>
<td>37</td>
</tr>
<tr>
<td>Scripts enabled</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Confirms **perception** that Debian is more secure than RedHat.
Lessons Learned from Windows and Linux Measurements

- Measurement method is ad-hoc

- Requires a security expert

- Focus is on measuring the attack surfaces of operating systems
Research Goals

• **Formalize** the notion of attack surface

• **Introduce a systematic** attack surface measurement method
  – Anyone, anywhere, anything

• **Validate** the method

• **Demonstrate** the uses of the method
Intuition Behind Attack Surfaces

A system’s attack surface is the ways in which an adversary can enter the system and potentially cause damage.

Hence we define a system’s attack surface in terms of the system’s resources (i.e., methods, channels, and data items).
Model of a System and its Environment

A system, $s$, and its environment, $E_s = \langle U, D, T=\{t_1, t_2\} \rangle$.

Formal model uses I/O automata [LT89].
Not All Resources Are Part of the Attack Surface

- Only those resources that the attacker can use to send data into or receive data from the system are relevant.

- We introduce the formal entry point and exit point framework to identify the relevant resources.
Entry Point and Exit Point Framework

• Entry Points/Exit Points
  – Direct (input/output action)
  – Indirect (internal action)

• Channels (e.g., sockets and pipes)
  – $c \in \text{Res}(m.\text{pre})$

• Untrusted Data Items (e.g., files)
  – $d \in \text{Res}(m.\text{post}), d \in \text{Res}(m.\text{pre})$
Attack Surface Definition

- **Definition**
  - $M$: set of entry points and exit points
  - $C$: set of channels
  - $I$: set of untrusted data items.

$$\text{attack surface} = \langle M, C, I \rangle$$

**Theorem:** Given an environment, $E$, if $AS(A) \geq AS(B)$, then $\text{Attacks}(A \parallel E) \supseteq \text{Attacks}(B \parallel E)$. 
Not All Resources Contribute Equally to the Attack Surface

- Contribution \( \propto \) Damage Potential
  
  Contribution \( \propto (\text{Attacker Effort})^{-1} \)

- Contribution = \( \frac{\text{Damage Potential}}{\text{Attacker Effort}} \)

Higher Damage Potential \( \Rightarrow \) Stronger m.post
  
  \( \Rightarrow \) more methods can follow m

Lower Attacker Effort \( \Rightarrow \) Weaker m.pre
  
  \( \Rightarrow \) m can follow more methods
Attack Surface Measurement (ASM)

- ASM(A) ≥ ASM(B) if there exists a nonempty set, R, of resources s.t.
  \[ \forall r \in R. \text{contribution}(r, A) \geq \text{contribution}(r, B). \]

Theorem: Given an environment, E, if ASM(A) ≥ ASM(B), then Attacks(A || E) ⊇ Attacks(B || E).
Quantitative Attack Surface Measurement

• Assume $\text{der}$: method $\rightarrow Q$.
  – Similarly, for channel and data.

$$\text{ASM} = \left( \sum_{m \in M} \text{der}(m), \sum_{c \in C} \text{der}(c), \sum_{d \in I} \text{der}(d) \right)$$

• Analogous to risk modeling

$$\sum_{m \in M} p(m)\text{der}(m)$$

probability = 1

consequence
Abstract Measurement Method

1. **Identify** a set, $M$, of entry points and exit points, a set, $C$, of channels, and a set, $I$, of untrusted data items.

2. **Estimate** each relevant resource’s damage potential-effort ratio, $\text{der}$.

3. **Compute** Attack Surface Measurement $= \left\langle \sum_{m \in M} \text{der}(m), \sum_{c \in C} \text{der}(c), \sum_{d \in I} \text{der}(d) \right\rangle$. 
C Measurement Method and Examples

• FTP Servers
  – ProFTP 1.2.10, Wu-FTP 2.6.2

• IMAP Servers
  – Courier 4.0.1, Cyrus 2.2.10
Step 1: Identify Relevant Resources

• Entry Points and Exit Points
  – Static analysis
  – \texttt{C library} methods (e.g., \texttt{read}) for data exchange
  – Call graph

• Channels and Untrusted Data Items
  – \texttt{Run time} monitoring
  – Open channels
  – Data read and written
Step 2: Damage Potential-Effort Ratio

<table>
<thead>
<tr>
<th>Resource</th>
<th>Damage Potential</th>
<th>Attacker Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Privilege</td>
<td>Access Rights</td>
</tr>
<tr>
<td>Channel</td>
<td>Protocol</td>
<td>Access Rights</td>
</tr>
<tr>
<td>Data Items</td>
<td>Type</td>
<td>Access Rights</td>
</tr>
</tbody>
</table>

Impose a total ordering among the values of the attributes and assign numeric values accordingly, e.g.,

root = 5 and auth = 3.
FTP Measurement Results

ProFTP = \langle 312.9, 1.0, 18.9 \rangle, \text{ Wu-FTP } = \langle 392.3, 1.0, 17.6 \rangle

Use domain knowledge to decide which dimension presents more risk and choose accordingly.
Validation

- Validating a software measure is hard [KPF97,....]
  - security metric is even harder

<table>
<thead>
<tr>
<th>Software measure</th>
<th>Attack surface</th>
<th>MS Bulletins, Expert Survey</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prediction System</td>
<td>Security Risk</td>
<td>IO Automata Model, Patch Analysis, Anecdotal Evidence</td>
</tr>
</tbody>
</table>

Liu and Traore independently validated our metric [LT07].
Validating the Measurement Method

Key Assumptions

• Three **dimensions** of the attack surface
• Damage potential-effort ratio
• Six **attributes**
  – method privilege, method access rights, channel protocol, channel access rights, data item type, and data item access rights
Statistical Analysis of Microsoft Security Bulletins (MSB)

• An MSB mentions a vulnerability and resources needed for exploitation
• Are methods, channels, and data used in the exploitation?
• Analyzed MSBs from 2004-2006

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods</td>
<td>✅</td>
</tr>
<tr>
<td>Channels</td>
<td>✅</td>
</tr>
<tr>
<td>Data</td>
<td>✅</td>
</tr>
</tbody>
</table>
## Results: The Attributes are Indicators of Damage Potential and Effort

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Significance</th>
<th>Correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Privilege</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Method Access Rights</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Protocol</td>
<td>✔️</td>
<td>?</td>
</tr>
<tr>
<td>Channel Access Rights</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Type</td>
<td>✔️</td>
<td>?</td>
</tr>
<tr>
<td>Data Access Rights</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>
Expert Linux System Administrator Survey

• MSB has no data relevant to a resource’s attackability
  – Could not validate damage potential-effort ratio
• Surveys are widely used to collect a wide range of data
  – Prior work uses surveys to validate measures [K87, ....]
  – Feedback from one target user group (Industrial collaboration for other target user group)
  – W.r.t. Linux (MSB w.r.t. Windows)
Results: A Majority of the Subjects Agree With Our Measurement Method

<table>
<thead>
<tr>
<th>Methods</th>
<th>✓</th>
<th>Privilege</th>
<th>✓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channels</td>
<td>✓</td>
<td>Method Access Rights</td>
<td>✓</td>
</tr>
<tr>
<td>Data</td>
<td>✓</td>
<td>Protocol</td>
<td>?</td>
</tr>
<tr>
<td>Damage Potential-Effort Ratio</td>
<td>✓</td>
<td>Channel Access Rights</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Type</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Data Access Rights</td>
<td>✓</td>
</tr>
</tbody>
</table>
Validating the Prediction System

• Show that if system A is more secure than system B, then $\text{ASM}(A) < \text{ASM}(B)$

• Assumption: Vulnerability patches improve software security
  – $\text{ASM}(\text{After Patch}) < \text{ASM}(\text{Before Patch})$

Patches **reduce** attack surface measurement
## Results: A Majority of the Patches Reduce ASM

<table>
<thead>
<tr>
<th>Software</th>
<th>Percentage of Patches that reduce ASM</th>
<th>Significance (p&lt; 0.05)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firefox 2.0</td>
<td>67%</td>
<td>✓</td>
</tr>
<tr>
<td>ProFTP (all)</td>
<td>70%</td>
<td>✓</td>
</tr>
<tr>
<td>All NVD Bulletins</td>
<td>76.9%</td>
<td>✓</td>
</tr>
</tbody>
</table>
Anecdotal Evidence from Industry

• Microsoft
  – Sasser Worm
  – Nachi Worm
  – Zotob Worm

• Firefox 2.0
  – SSL buffer overflow
Collaboration with SAP

• SAP is world’s largest provider of enterprise-scale software
  – Complex technology platforms and business applications

• Demonstrate that the measurement method scales to enterprise-scale software

• Receive feedback from software architects and developers
Java Measurement Tool Screenshot
Results

• Measured the attack surface of a key component of SAP component
  – Measurement results conform to expectation
  – Detailed tool output, incremental analysis, and what-if scenarios are useful for attack surface reduction
  – Lessons learned
ASM in Software Development

LifeCycle

Use ASM to guide testing and code inspection
[MuSecurity, SAP]

Use ASM to choose a secure configuration

Use ASM in patch implementation

Compare and reduce ASM from version to version
[Microsoft, Firefox, OpenSSH]
Future Work: Software Development

- **Range analysis**

- **Other uses**
  - "Safe" software composition
  - Testing, deployment, maintenance
Future Work: Software Consumers

• Attack surface measurement in the absence of source code
  – Components as Entry/Exit points
  – Channels and Data as before

• Multiple metrics are needed for decision support

How do we combine multiple measures?
Related Work-1

• Prior work assumes the knowledge of vulnerabilities [AB95, VGMCM96, ODM99...]

• ASM is based on a system’s inherent properties
  – Formal framework encompasses past, present, and future vulnerabilities
  – Complementary to prior work
Related Work-2

• Prior work takes an attacker-centric approach [S99, MBFB05, LB08,..]

• ASM takes a system-centric approach
  – Depends on a system’s design
  – No assumptions about the attacker
  – Can be used as a tool in software development
Related Work-3

• Prior work is conceptual in nature and haven’t been applied to real systems [AB95, MGVT02, S04,..]

• We measured the attack surfaces of real-world software
  – FTP servers, IMAP servers
  – SAP business applications
Summary

• Introduced a **pragmatic** approach for security measurement
  – **Software industry** found it useful [Microsoft, Firefox, OpenSSH, MuSecurity, SAP, ..]

• **First step** in the grander challenge of security metrics
  – Understanding over time will lead to more meaningful metrics

Acknowledgements: Jeannette Wing, Roy Maxion, Virgil Gligor, Mike Reiter, Yuecel Karabulut, Effrat Keren, Dilsun Kaynar, Kymie Tan, Gourav Kataria, Miles McQueen, Mark Flynn, Michael Howard, Paul Hoffman, Mary Shaw, and Survey Participants.
Backups
I/O Automata [LT89]

• Action Signature
  – Input, Output, Internal actions
  – Pre and Post conditions
    m.pre and m.post

• Composition
  – \( E_s = (U_{io} \parallel D_{io} \parallel ( \parallel t_{io} )) \)
  – \( P = s_{io} \parallel E_s \quad t_{io} \in T_{io} \)
Validation of the Attributes

- An MSB has a severity rating and mentions the six resources attributes

<table>
<thead>
<tr>
<th>Impact</th>
<th>Damage Potential</th>
</tr>
</thead>
<tbody>
<tr>
<td>Difficulty</td>
<td>Attacker Effort</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Significant Predictor</th>
<th>Two sided Z-test (p &lt; 0.05)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correlation</td>
<td>Sign of Coefficient in Ordered Logistic Regression</td>
</tr>
</tbody>
</table>
Inspiration: Howard’s Relative Attack Surface Quotient (RASQ) [H03]

• Howard’s informal RASQ Measurement Method
  – Identify a system’s **attack vectors**
  – Assign **weights** to the attack vectors to reflect their **attackability**
  – RASQ = **sum of the weighted counts** of the attack vectors
Direct Entry Points

Methods that directly receive data.

direct entry point: an input action with a matching output action
Indirect Entry Points

Methods that \textit{indirectly} receive data.

\textbf{Indirect entry point: internal action}

\[ (m1.\text{post} \Rightarrow m.\text{post}) \land (d \in \text{Res}(m1.\text{post}) \land d \in \text{Res}(m.\text{pre})) \]
Channels and Data

Channels (e.g., sockets and pipes)
  • $c \in \text{Res}(m.\text{pre})$

Untrusted Data Items (e.g., files)

$d \in \text{Res}(m.\text{post})$

$d \in \text{Res}(m.\text{pre})$
Definition of An Attack

\[ \text{Attacks } (s_{io}) = \text{Set of executions of } (s_{io} \parallel E_s) \text{ that contain either an input action or output action of } s_{io}. \]
Not All Resources Contribute Equally to the Attack Surface

- contribution $\alpha$ damage potential
  
  $\alpha \frac{1}{\text{attacker effort}}$

- $r_1 \geq r_2$ if higher damage potential and/or lower attacker effort

\[
m(\text{MA, CA, DA, MB, CB, DB})
\]

**pre:** $P_{\text{pre}} \land (\text{MA} \geq m.\text{ef}) \land (\text{CA} \geq c.\text{ef}) \land (\text{DA} \geq d.\text{ef})$

**post:** $P_{\text{post}} \land (\text{MB} \geq m.\text{dp}) \land (\text{CB} \geq c.\text{dp}) \land (\text{DB} \geq d.\text{dp})$
Damage Potential-Effort Ratio

• Contribution $\propto$ Damage Potential

Contribution $\propto (\text{Attacker Effort})^{-1}$

• Contribution $=$ \[
\frac{\text{Damage Potential}}{\text{Attacker Effort}}
\]
C Measurement Method

Source Code → Annotation

Annotation → Annotated Sourcecode → Callgraph Generator And Analyzer

C Library Methods

Callgraph Generator And Analyzer → Entry/Exit Points

Entry/Exit Points → AS Computation

AS Computation → Numeric Values

AS Measurements

AS Computation → Channels

Channels → Data Items

Data Items → Runtime Monitoring

Runtime Monitoring → Running Process

Running Process → Compilation And Execution

Compilation And Execution → Source Code
Survey Methodology

• Email survey of experienced Linux system administrators
  – Diverse background and geographic location

• Questions on a five point Likert scale [L32]
  – Pretesting and interviewing to avoid bias
  – Self-selection bias

• Descriptive analysis techniques
  – Central tendency bias
  – %age of agreement, disagreement, and neither
  – t-test ($p < 0.05$)
Not All Patches Are Relevant

• Heuristics: vulnerability type determines patch relevance
  – Use National Vulnerability Database (NVD) type information
  – Infer type if missing
• Not all relevant patches reduce the attack surface
• Consider local effect of a patch
Data Collection for Firefox 2.0

Relevant?

MITRE

Reduces ASM?
Java Measurement Method

• Focus on **method** dimension

• Entry Points and Exit Points
  – Call graph
  – **Interface** methods, methods invoking other systems’ interfaces and Java I/O library methods

• Damage Potential-Effort Ratio
  – Use SAP’s **threat modeling** process to assign numbers
Tool Usage in Software Development

• Tool produces **detailed output**
  – Guides attack surface reduction

• **Incremental** analysis

• **What-If** scenarios
  – Addition of a new feature
  – Removal of a feature
FTP Daemons (method)

1. Access rights **don't** matter.
2. **proftpd** privilege level contributes more than auth. s.
Tool Output

Measurements

Entry (Exit) Point

Contribution

Reason


References- 2


References- 3


