Quantitative Security

Colorado State University

Yashwant K Malaiya

CS 559

L13

CSU Cybersecurity Center
Computer Science Dept
Today’s Outline

• Some thoughts
• Review
• Seasonality
• Multi-version software
• Long term effects
Is hacking legal?

• That depends on what you mean by hacking.
  – Original meaning (MIT, 1960): informal programming
  – Unauthorized access of computing systems is illegal.
    • Can be done by people with limited expertise.
  – Discovering vulnerabilities in software/systems one owns is not illegal.
    • May take significant skill.
  – Scanning for security holes in systems you don’t own, is not legal.
  – Paying ransom for data (ransomware) is not legal in USA.
  – Disclosure/selling of zero-day vulnerabilities may be controlled by governments.
Dimensions and Approximations

- For real problems, proper approximations are essential.
  - \((1,000,001 - 1,000,000)\) may not be approximated as 0.
- Note the distinction between K \(10^3\) and M \(10^6\). You must convert numbers appropriately.
- You need to keep dimensions in mind.
  - Fort Collins to Denver is ______ miles.
  - Windows 10 is about 50 Million lines of code.
  - Documented smallest software defect density is 0.1/KLOC (space shuttle software).
  - In OS, the vulnerabilities are about 1% of the defects.
What you should question

• A claim should probably be tentatively accepted if
  – It is consistent with well established, carefully researched observations
  – Credibility of the researchers and publication

• Question a claim if
  – You think you can come up with a better idea

• Researchers (unlike managers) do not claim they know everything.
Term Research Project

• Select your topic idea asap.

• Project Proposal & Sources: due Oct 10
  – See requirements.

• Semi-final report: due Nov 7
  – Lit review done, some preliminary results

• Slides/Presentation: Nov 18, Nov 19-Dec 8
  – interactive

• Final report: due Dec 9
  – Possible publication

• Critical Peer reviews: due Dec 10
Quantitative Security
Colorado State University
Yashwant K Malaiya
CS 559
Vulnerability Discovery Models

CSU Cybersecurity Center
Computer Science Dept
Time–vulnerability Discovery model

3 phase model S-shaped model.

• Phase 1:
  • Installed base – low.

• Phase 2:
  • Installed base–higher and growing/stable.

• Phase 3:
  • Installed base–dropping.

\[
\frac{dy}{dt} = Ay(B - y)
\]

\[
y = \frac{B}{BCe^{-ABt} + 1}
\]

Time–based model: Windows 98

<table>
<thead>
<tr>
<th>Windows 98</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.004873</td>
</tr>
<tr>
<td>B</td>
<td>37.7328</td>
</tr>
<tr>
<td>C</td>
<td>0.5543</td>
</tr>
<tr>
<td>$X^2$</td>
<td>7.365</td>
</tr>
<tr>
<td>$X^2_{critical}$</td>
<td>60.481</td>
</tr>
<tr>
<td>P-value</td>
<td>1 - 7.6x10^{-11}</td>
</tr>
</tbody>
</table>
Usage – vulnerability Discovery model

- The data:
  - The global internet population.
  - The market share of the system during a period of time.

- **Equivalent effort**
  - The real environment performs an intensive testing.
  - Malicious activities is relevant to overall activities.
  - Defined as

\[
E = \sum_{i=0}^{n} (U_i \times P_i)
\]
Software Reliability Modeling

• Applicable to general software bugs

• Key Static software metrics
  – Software size (without comments, KLOC)
  – Defect density (total defects/size)
    • Typical range Range 16 -0.1 /KLOC
    • Software evolution/reuse, requirement volatility
    • Team capabilities, extent of testing
  – Defect finding efficiency

 0.1/KLOC  Space Shuttle
Exponential Reliability Growth Model

• Assumption: rate of finding and removing bugs proportional to the number of bugs present at time t.

\[- \frac{dN(t)}{dt} = \beta_1 N(t)\]

Which yields

\[N(t) = N(0)e^{-\beta_1 t}\]

• Cumulative number of defects found is

\[N(0)(1 - e^{-\beta_1 t})\]

• Defect finding rate is

\[N(0)e^{-\beta_1 t}\]

• \(N(0)\) may be estimated using defect density and size

• \(\beta_1\) depends on defect finding efficiency
Usage – vulnerability Discovery model

• The model: growth with effort.
• Growth model based on the exponential SRGM
• Time is eliminated.
• \( y = N(0)(1 - e^{-\beta_1E}) \)
Vulnerability density and defect density

• Defect density
  – Valuable metric for planning test effort
  – Used for setting release quality target
  – Some data is available
  – Depends on various factors, may be stable for a team/process

• Vulnerabilities are a class of defects
  – Vulnerability data is in the public domain.
  – Is vulnerability density a useful measure?
  – Is it related to defect density?
    • Vulnerabilities = 5% of defects [Longstaff]?
    • Vulnerabilities = 1% of defects [Anderson]?

• Can be a major step in measuring security.
**Vulnerability density and defect density**

- **Vul dens**: 95/98: 0.003-0.004, NT/2000/XP: 0.01-0.02, Apache 0.04
- **V_{KD}/D_{KD}**: about 1% for client OSs, Higher for HTTP servers, server OSs

<table>
<thead>
<tr>
<th>System</th>
<th>MSLOC</th>
<th>Known Defects (1000s)</th>
<th>D_{KD} (/Kloc)</th>
<th>Known Vulnerabilities</th>
<th>V_{KD} (/Kloc)</th>
<th>Ratio V_{KD}/D_{KD}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Win 95</td>
<td>15</td>
<td>5</td>
<td>0.33</td>
<td>46</td>
<td>0.0031</td>
<td>0.92%</td>
</tr>
<tr>
<td>NT 4.0 server</td>
<td>16</td>
<td>10</td>
<td>0.625</td>
<td>162</td>
<td>0.0101</td>
<td>1.62%</td>
</tr>
<tr>
<td>Win 98</td>
<td>18</td>
<td>10</td>
<td>0.556</td>
<td>84</td>
<td>0.0047</td>
<td>0.84%</td>
</tr>
<tr>
<td>Win 2000</td>
<td>35</td>
<td>63</td>
<td>1.8</td>
<td>508</td>
<td>0.0145</td>
<td>0.81%</td>
</tr>
<tr>
<td>Win XP</td>
<td>40</td>
<td>106.5*</td>
<td>2.66*</td>
<td>728</td>
<td>0.0182</td>
<td>0.68%*</td>
</tr>
<tr>
<td>Apache HTTP 2006 (Unix)</td>
<td>227</td>
<td>4148</td>
<td>18.27</td>
<td>96</td>
<td>0.423</td>
<td>2.32%</td>
</tr>
<tr>
<td>Firefox</td>
<td>2.5</td>
<td>24,027</td>
<td>9.61</td>
<td>134</td>
<td>0.0536</td>
<td>0.557%</td>
</tr>
</tbody>
</table>

MS Thesis Woo, 2006
### Vulnerability Discovery Models

<table>
<thead>
<tr>
<th>Model</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NHPP Power-law (Movahedi et al., 2018)</td>
<td>( \Omega(t) = (\beta^{-\alpha})t^\alpha )</td>
</tr>
<tr>
<td>Gamma-based VDM (Joh and Malaiya, 2014)</td>
<td>( \Omega(t) = \int_{t=0}^{t_0} \frac{\gamma}{\Gamma(\alpha)\beta^\alpha} t^{\alpha-1} e^{-\frac{t}{\beta}} dt )</td>
</tr>
<tr>
<td>Weibull-based VDM (Kim et al., 2007)</td>
<td>( \Omega(t) = \gamma { 1 - e^{-\left(\frac{t}{\beta}\right)^\alpha} } )</td>
</tr>
<tr>
<td>AML VDM (Alhazmi and Malaiya, 2005)</td>
<td>( \Omega(t) = \frac{\beta \gamma}{B \text{e}^{-\beta t} + 1} )</td>
</tr>
<tr>
<td>Normal-based VDM (Joh and Malaiya, 2014)</td>
<td>( \Omega(t) = \frac{\gamma}{1 + e^{-\left(\frac{t}{\beta}\right)}} )</td>
</tr>
<tr>
<td>Rescorla Exponential (RE) (Rescorla, Jan. 2005)</td>
<td>( \Omega(t) = \gamma (1 - e^{-\lambda t}) )</td>
</tr>
<tr>
<td>Rescorla Quadratic (RQ) (Rescorla, Jan. 2005)</td>
<td>( \Omega(t) = \frac{At^2}{2} + Bt )</td>
</tr>
<tr>
<td>Younis Folded (YF) (Younis et al., 2011)</td>
<td>( \Omega(t) = \frac{\gamma}{2} \left{ \text{erf}(\frac{t - \mu}{\sqrt{2\sigma}}) + \text{erf}(\frac{t + \mu}{\sqrt{2\sigma}}) \right} )</td>
</tr>
<tr>
<td>Linear Model (LM) (Alhazmi and Malaiya, 2006)</td>
<td>( \Omega(t) = At + B )</td>
</tr>
</tbody>
</table>

Table of models and their equations

Seasonality in Vulnerability Discovery
Seasonality in Vulnerability Discovery

• **Vulnerability Discovery Model (VDM):**
  – a probabilistic methods for modeling the discovery of software vulnerabilities \[Ozment\]
  – Spans a few years: introduction to replacement

• **Seasonality:** periodic variation
  – well known statistical approach
  – quite common in economic time series
    • Biological systems, stock markets etc.

*Halloween indicator:* Low returns in May-Oct.
Examining Seasonality

• Is the seasonal pattern statistically significant?
• Periodicity of the pattern
• Analysis:
  – Seasonal index analysis with test
  – Autocorrelation Function analysis
• Significance
  – Enhance VDMs’ predicting ability
• Annual and Weekly seasonality
## Annual: Prevalence in Month

### Vulnerabilities Disclosed

<table>
<thead>
<tr>
<th></th>
<th>WinNT '95~'07</th>
<th>IIS '96~'07</th>
<th>IE '97~'07</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan</td>
<td>42</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Feb</td>
<td>20</td>
<td>10</td>
<td>32</td>
</tr>
<tr>
<td>Mar</td>
<td>12</td>
<td>2</td>
<td>22</td>
</tr>
<tr>
<td>Apr</td>
<td>13</td>
<td>11</td>
<td>29</td>
</tr>
<tr>
<td>May</td>
<td>18</td>
<td>12</td>
<td>41</td>
</tr>
<tr>
<td>Jun</td>
<td>24</td>
<td>17</td>
<td>45</td>
</tr>
<tr>
<td>Jul</td>
<td>18</td>
<td>11</td>
<td>53</td>
</tr>
<tr>
<td>Aug</td>
<td>17</td>
<td>7</td>
<td>42</td>
</tr>
<tr>
<td>Sep</td>
<td>11</td>
<td>6</td>
<td>26</td>
</tr>
<tr>
<td>Oct</td>
<td>14</td>
<td>6</td>
<td>20</td>
</tr>
<tr>
<td>Nov</td>
<td>18</td>
<td>7</td>
<td>26</td>
</tr>
<tr>
<td>Dec</td>
<td>51</td>
<td>28</td>
<td>93</td>
</tr>
<tr>
<td>Total</td>
<td>258</td>
<td>132</td>
<td>444</td>
</tr>
<tr>
<td>Mean</td>
<td>21.5</td>
<td>11</td>
<td>37</td>
</tr>
<tr>
<td>s.d.</td>
<td>12.37</td>
<td>6.78</td>
<td>20.94</td>
</tr>
</tbody>
</table>

### Percentage of Vuln. for Month

The bar graph shows the percentage of vulnerabilities disclosed by month for WinNT, IIS, and Internet Explorer. The x-axis represents the months from January to December, and the y-axis represents the percentage of vulnerabilities. The graph highlights the variation in prevalence across different months.
Seasonal Index

<table>
<thead>
<tr>
<th></th>
<th>WinNT</th>
<th>IIS</th>
<th>IE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan</td>
<td>1.95</td>
<td>1.36</td>
<td>0.41</td>
</tr>
<tr>
<td>Feb</td>
<td>0.93</td>
<td>0.91</td>
<td>0.86</td>
</tr>
<tr>
<td>Mar</td>
<td>0.56</td>
<td>0.81</td>
<td>0.59</td>
</tr>
<tr>
<td>Apr</td>
<td>0.60</td>
<td>1.00</td>
<td>0.78</td>
</tr>
<tr>
<td>May</td>
<td>0.84</td>
<td>1.09</td>
<td>1.11</td>
</tr>
<tr>
<td>Jun</td>
<td>1.12</td>
<td>1.55</td>
<td>1.22</td>
</tr>
<tr>
<td>Jul</td>
<td>0.84</td>
<td>1.00</td>
<td>1.43</td>
</tr>
<tr>
<td>Aug</td>
<td>0.79</td>
<td>0.64</td>
<td>1.14</td>
</tr>
<tr>
<td>Sep</td>
<td>0.51</td>
<td>0.55</td>
<td>0.70</td>
</tr>
<tr>
<td>Oct</td>
<td>0.65</td>
<td>0.55</td>
<td>0.54</td>
</tr>
<tr>
<td>Nov</td>
<td>0.84</td>
<td>0.64</td>
<td>0.70</td>
</tr>
<tr>
<td>Dec</td>
<td>2.37</td>
<td>2.55</td>
<td>2.51</td>
</tr>
<tr>
<td>$\chi^2_C$</td>
<td>19.68</td>
<td>19.68</td>
<td>19.68</td>
</tr>
<tr>
<td>$\chi^2_S$</td>
<td>78.37</td>
<td>46</td>
<td>130.43</td>
</tr>
<tr>
<td>p-value</td>
<td>3.04e-12</td>
<td>3.23e-6</td>
<td>1.42e-6</td>
</tr>
</tbody>
</table>

- **Seasonal index**: measures how much the average for a particular period tends to be above (or below) the expected value.
- **$H_0$**: no seasonality is present. We will evaluate it using the monthly seasonal index values given by [4]:

  \[
  s_i = \frac{d_i}{d}
  \]

  where, $s_i$ is the seasonal index for $i^{th}$ month, $d_i$ is the mean value of $i^{th}$ month, $d$ is a grand average.

Autocorrelation function (ACF)

- Plot of autocorrelations function values
- With time series values of $z_b, z_{b+1}, ..., z_n$, the ACF at lag $k$, denoted by $r_k$, is [5]:

$$r_k = \frac{\sum_{t=b}^{n-k}(z_t - \overline{z})(z_{t+k} - \overline{z})}{\sum_{t=b}^{n}(z_t - \overline{z})^2}$$

, where

- Measures the linear relationship between time series observations separated by a lag of time units
- Hence, when an ACF value is located outside of confidence intervals at a lag $t$, it can be thought that every lag $t$, there is a relationships along with the time line

Autocorrelation (ACF): Results

- Expected lags corresponding to 6 months or its multiple would have their ACF values outside confidence interval.
- Upper/lower dotted lines: 95% confidence intervals.
- An event occurring at time $t + k$ ($k > 0$) lags behind an event occurring at time $t$.
- Lags are in month.
Why seasonality?

Weekly Seasonality

Figure 1. Run charts for unpatched critical vulnerabilities in 2008 and Exploitation with their corresponding ACFs. The upper two plots are normalized using the maximum value as 100%. In the bottom two plots, legs are in day.

Halloween Indicator

- “Also known as “Sell in May and go away”
- Global (1973-1996):
  - Nov.-April: 12.47% ann., st dev 12.58%
  - 12-months:10.92%, st. dev. 17.76%
- 36 of 37 developing/developed nations
- Data going back to 1694
- “No convincing explanation”

Quantitative Security

Colorado State University
Yashwant K Malaiya
CS 559
Multi-version Systems

CSU Cybersecurity Center
Computer Science Dept
• Motivation
• Software Evolution
• Multi-version Software Discovery Model
  – Apache, Mysql and Win XP data
Motivation for Multi-version VDMs

- Superposition effect on vulnerability discovery process due to shared code in successive versions.
- Examination of software evolution: impact on vulnerability introduction and discovery.
- Other factors impacting vulnerability discovery process not considered before.
Software Reuse

• New software projects use both new and reused blocks.
  – New blocks have a higher defect density because they have undergone less testing.
  – Reused blocks are more reliable.
  – Some defects may be introduced at the new/reused block interface.
  – Overall defect density is weighted average of the two.
  – Encounter rate during execution depends on weighted usage
Software Evolution

• The modification of software during maintenance or development:
  – fixes and feature additions.
  – Influenced by competition

• Code decay and code addition introduce new vulnerabilities

• Successive version of a software can share a significant fraction of code.

Software Evolution: Apache & Mysql

Modification: Apache 43%, Mysql 31%

Some vulnerabilities are in added code, many are inherited from precious versions.
• **Observation**
  - Vulnerability increases after saturation in AML modeling

• **Accounting for Superposition Effect**
  - Shared components between several versions of software
## Multi-version Vulnerability Discovery

![Graph of Multiple Software Vulnerability Discovery Trend](image)

The probability of vulnerability discovery, \( \Omega(t) \), can be calculated using the following equation:

\[
\Omega(t) = \frac{B}{BCe^{-ABt} + 1} + \alpha \frac{B'}{B'C'e^{-A'B'(t-\varepsilon)} + 1}
\]

<table>
<thead>
<tr>
<th>Software</th>
<th>Previous Version</th>
<th>Next Version</th>
<th>Shared Code Ratio ( \alpha )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache</td>
<td>1.3.24 (3-21-2002)</td>
<td>2.0.35 (4-6-2002)</td>
<td>20.16%</td>
</tr>
<tr>
<td>Mysql</td>
<td>4.1.1 (12-1-2003)</td>
<td>5.0.0 (12-22-2003)</td>
<td>83.52%</td>
</tr>
</tbody>
</table>
One vs Two Humps

One-humped Vulnerability Discovery Model

Superposition affect
Multi-version Vulnerability Discovery

- May result in a single hump with prolonged linear period
Gradually evolving software
Software evolves in each version.

• Existing code fixed
  – some vulnerabilities found and patched

• Code added for increasing functionality
  – New vulnerabilities injected
  – Total number of vulnerabilities may remain about the same

• Overall code size keeps increasing
  – Vulnerability discovery rate may remain stable
Linear model

- Because of nearly continuous evolution, the linear phase may get stretched.

- If the evolution rate is steady, the size of the pool of undiscovered vulnerabilities stays the same (vulnerabilities removal rate = injection rate)

- If the market share is steady, the number of vulnerability finders remains steady.

Joh’s thesis
• Four Windows releases: 500 vulnerabilities during July 1998-July 2009
• Size: 35-50 M LOC
• Slope = **about 45 vulnerabilities/year**
• Further investigation is needed.

Data from Joh’s thesis
Long Term Trends

- Long term Trends: Total vulnerabilities, Microsoft products
Long Term Trends

- Long term Trends: Microsoft products, Win XP, Win 10
Long Term Trends

- Size evolution: Linus kernel
Likely factors that affect long-term trends

• Better understanding of safer coding practices
  – Fewer vulnerabilities injected?

• Better vulnerability discovery tools (fuzzers) and more finders
  – Higher vulnerability discovery rates

• More software products
  – More vulnerabilities to be found
What factors impact risk?

• Not the vulnerabilities that have been found and patched

• Vulnerabilities that have been discovered but not patched
  – Before disclosure: black hat people/organizations
  – after disclosure: when patch development is taking time
  – Vulnerabilities with patches, but patches not applied

• Statistical modeling may be needed for assessing probability of breaches