Quantitative Security

Colorado State University
Yashwant K Malaiya

CS 559
L13



CSU Cybersecurity Center
Computer Science Dept

Today's Outline

- Dimensions
- Vulnerability discover models
- Seasonality
- Multi-version software
- Long term effects
- Security Metrics

Term Research Project etc .

- Select your topic idea by Oct 11.
- HW2a: Due Oct 15
- Project Proposal & Sources: due Oct 14
  - See requirements.
  - Midterm: On Campus: Nov 3 Online: Nov 4 (12:10 AM-11:50PM).
- Semi-final report: due Nov 11
  - Lit review done, some preliminary results
- Slides/Presentation: Nov 29, Dec 1, 3
  - interactive
- Final report: due Dec 9
  - Possible publication
- Critical Peer reviews: due Dec 15
- Optional Extra Credit: Prop 11/18/26, Report 12/16/26

Quantitative Security

Colorado State University
Yashwant K Malaiya

CS 559
Vulnerability Discovery Models



CSU Cybersecurity Center
Computer Science Dept

Goal: Modeling Vulnerability Discovery

- Developing a quantitative model to estimate vulnerability discovery.
  - Using *calendar time*.
  - Using *equivalent effort*.
- Validate these measurements and models.
  - Testing the models using available data
- Identify security Assessment metrics
  - *Vulnerability density*
  - *Vulnerability to Total defect ratio*

AML VDM

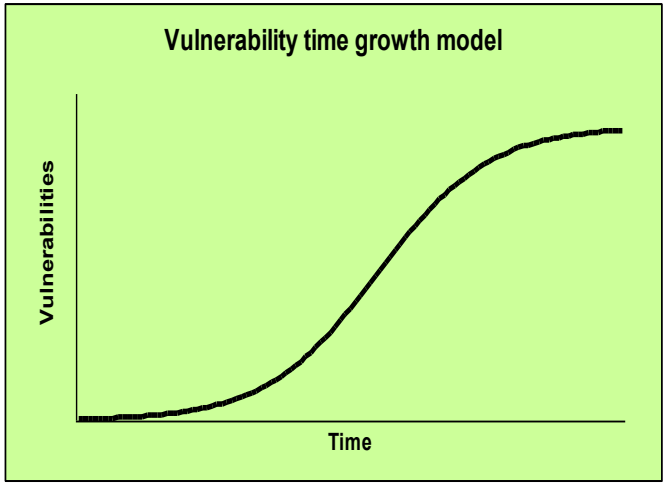
Alhazmi Malaiya Logistic (AML) vulnerability discovery model

$$\frac{d\Omega}{dt} = A\Omega(B - \Omega), \qquad (3)$$

where Ω is the cumulative number of vulnerabilities, t is the calendar time, and initially t=0. A and B are empirical constants determined from the recorded data.

By solving the differential equation we obtain

$$\Omega(t) = \frac{B}{BCe^{-ABt} + 1}, \qquad (4)$$



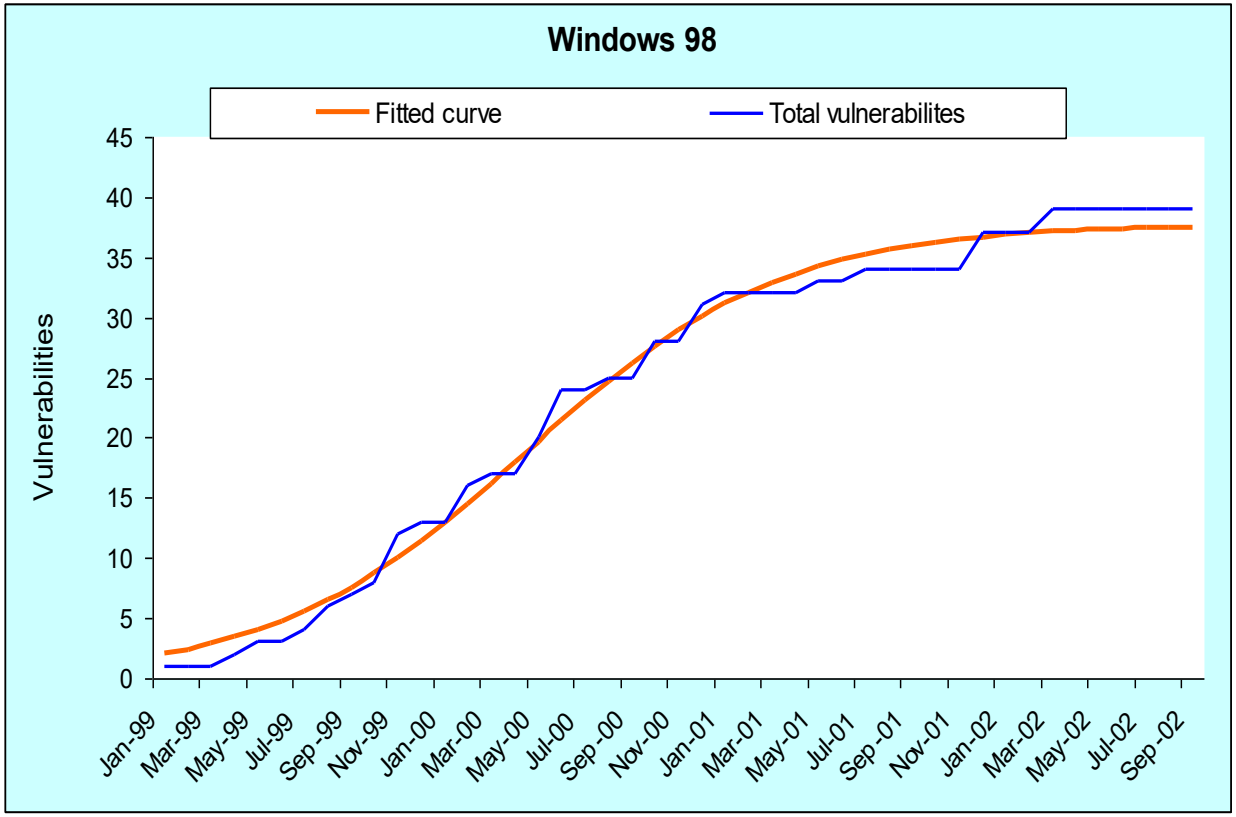
3 phase model S-shaped model.

- Phase 1: Installed base –low.
- Phase 2: Installed base–higher and growing/stable.
- Phase 3: Installed base–dropping.

O. H. Alhazmi and Y. K. Malaiya, "Quantitative Vulnerability Assessment of Systems Software Proc. Ann. IEEE Reliability and Maintainability Symp., 2005, pp. 615-620

Colorado State University

Time–based model: Windows 98



	Windows 98
A	0.004873
B	37.7328
C	0.5543
X^2	7.365
$X^2_{critial}$	60.481
P-value	$1- 7.6x10^{-11}$

Colorado State University

Usage –vulnerability Discovery model

The data:

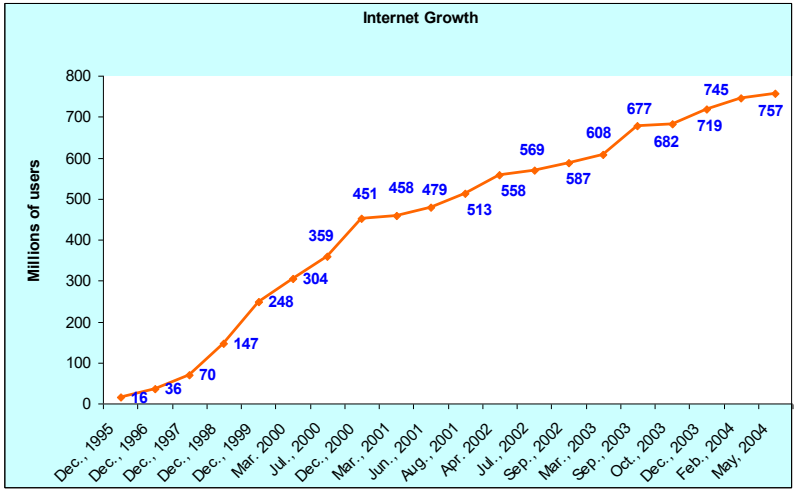
- The global internet population.
- The market share of the system during a period of time.

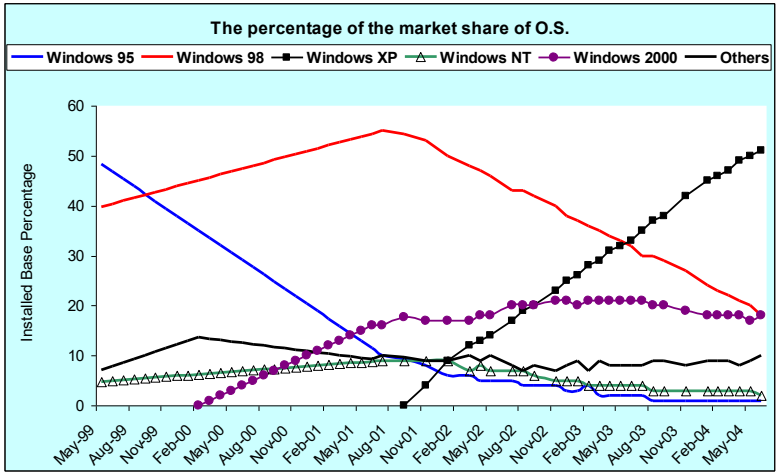
Equivalent effort

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

$$E = \sum_{i=0}^{n} (U_i \times P_i)$$

- U_i is total number of systems, P_i is percentage of systems using the specific package

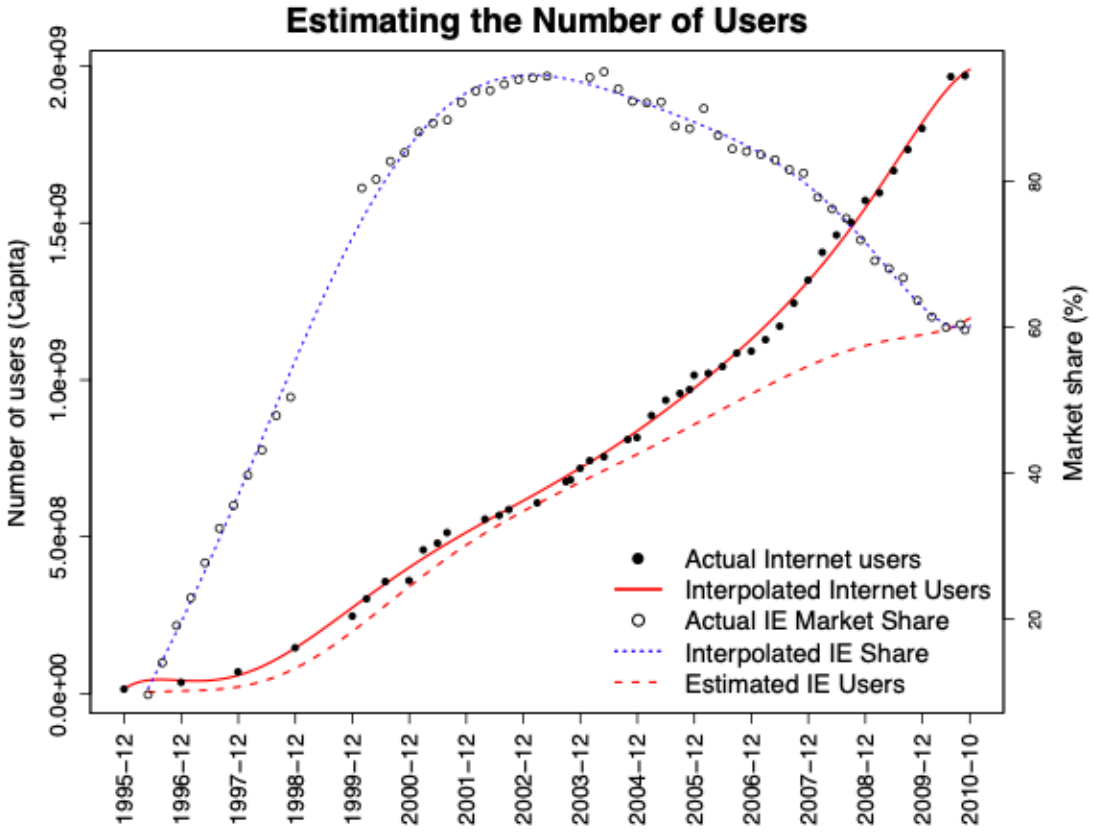




Woo, Alhazmi and Malaiya, "Assessing Vulnerabilities in Apache and IIS HTTP Servers", Proc. IEEE DASC'06

Colorado State University

Estimating number of users



Estimating the number of IE users (internet users x market share)

QUANTITATIVE ANALYSES OF SOFTWARE VULNERABILITIES, HyunChul Joh, PhD Dissertation 2011

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
- Software reliability growth models (SRGMs)

Exponential SRGM

Exponential Reliability Growth Model

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

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

Which yields

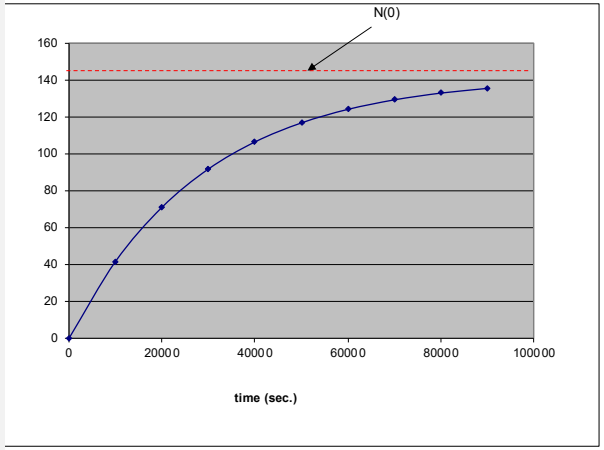
$$N(t) = N(0)e^{-\beta_1 t}$$

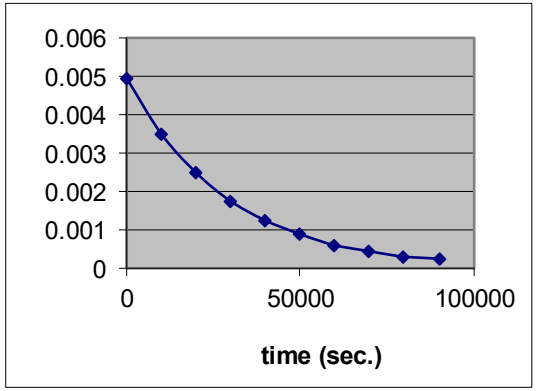
- Cumulative number of defects found is

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

- Defect finding rate is

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



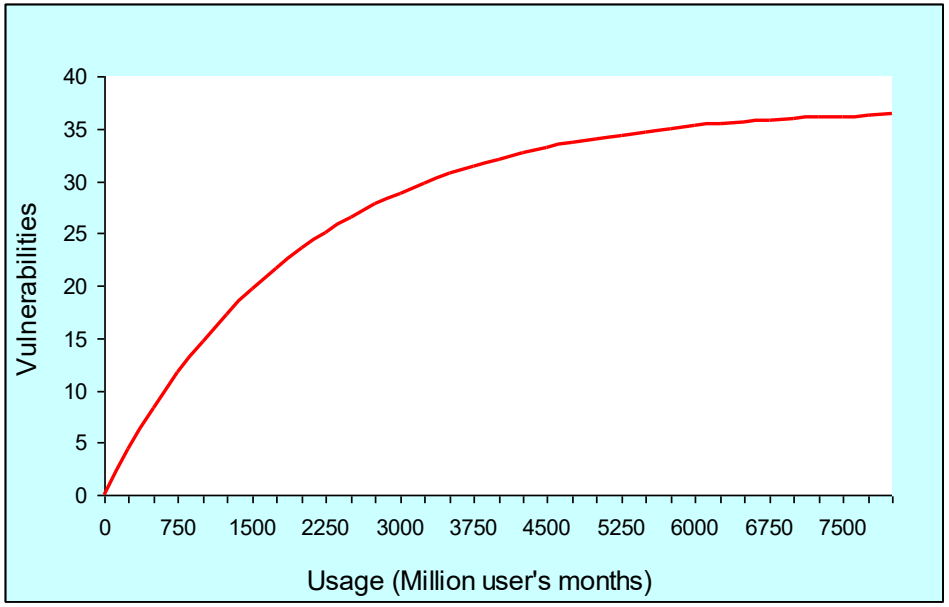


- N(0) may be estimated using defect density and size
- β_1 depends to defect finding efficiency

Colorado State University

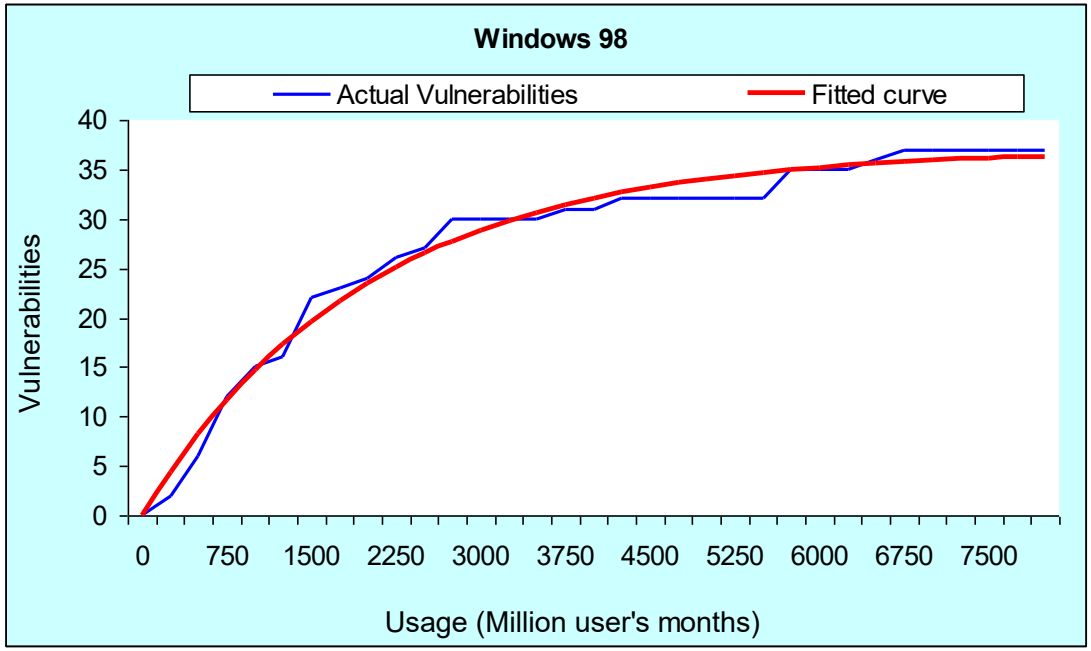
Usage –vulnerability Discovery model

- The model: growth with effort.
- Growth model based on the exponential SRGM [Musa].
- Time is eliminated.
- $y = N(0)(1 - e^{-\beta_1 E})$



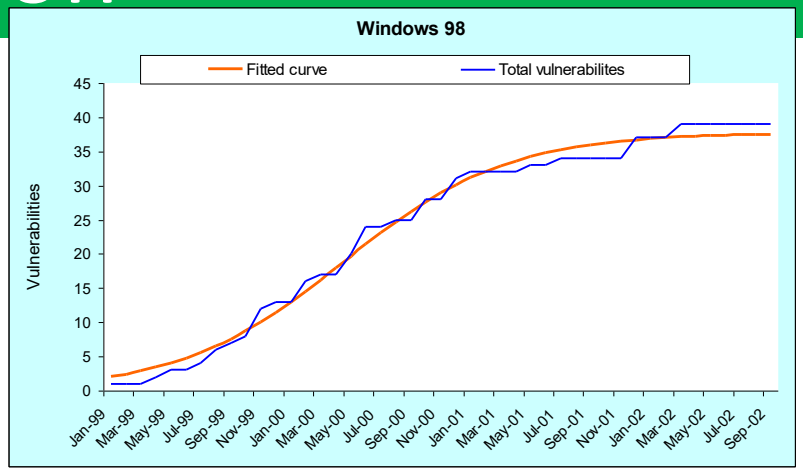
Colorado State University

Effort-based model: Windows 98



	Windows 98
B	37
λ_{vu}	0.000505
χ^2	3.510
$\chi^2_{critial}$	44.9853
P-value	1- 3.3×10^{-11}

Discussion



- Excellent fit for Windows 98 and NT 4.0.
- Model fits data for all OSs examined.
- Deviation from the model caused by overlap:
  - Windows 98 and Windows XP
  - Windows NT 4.0 and Windows 2000
- Vulnerabilities in shared code may be detected in the newer OS.
- Need: approach for handling such overlap

Quantitative Security

Colorado State University
Yashwant K Malaiya

CS 559
Multi-version Systems



CSU Cybersecurity Center
Computer Science Dept

Motivation for Multi-version VDMs

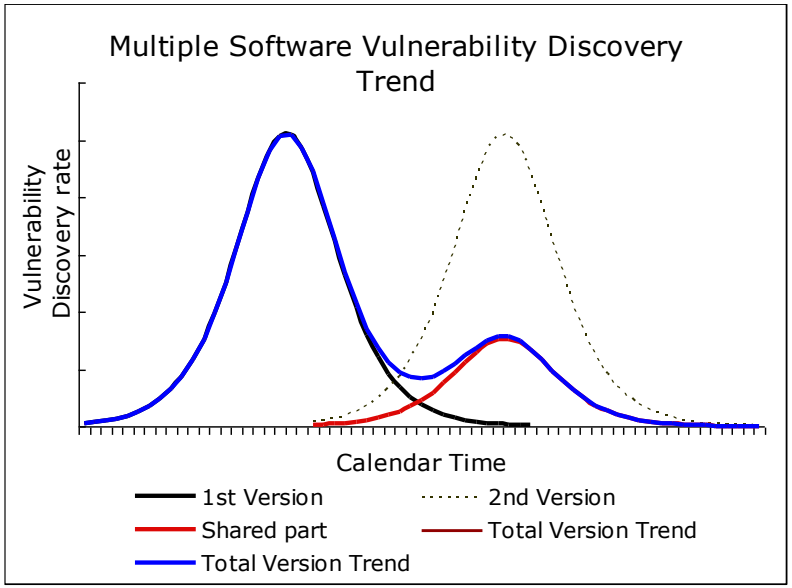
- Superposition effect on vulnerability discovery process due to shared code in successive versions.
- Examination of software evolution: impact on vulnerability introduction and discovery
- Other factors impacting vulnerability discovery process not considered before

16

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

Multi-version Vulnerability Discovery



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

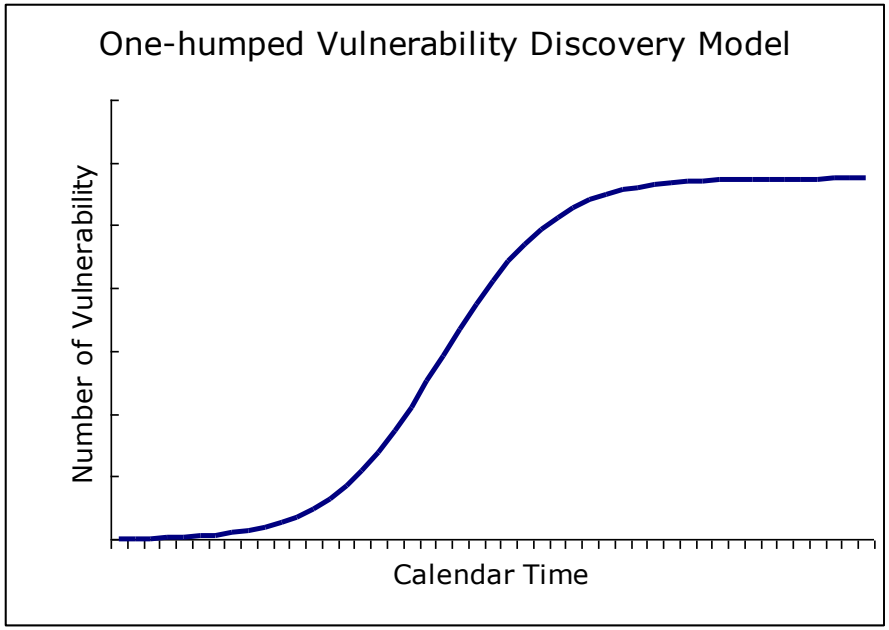
Parameter α indicates shared components such as shared code and shared functionality, and the parameter ε is the time lag between the release dates of the two versions.

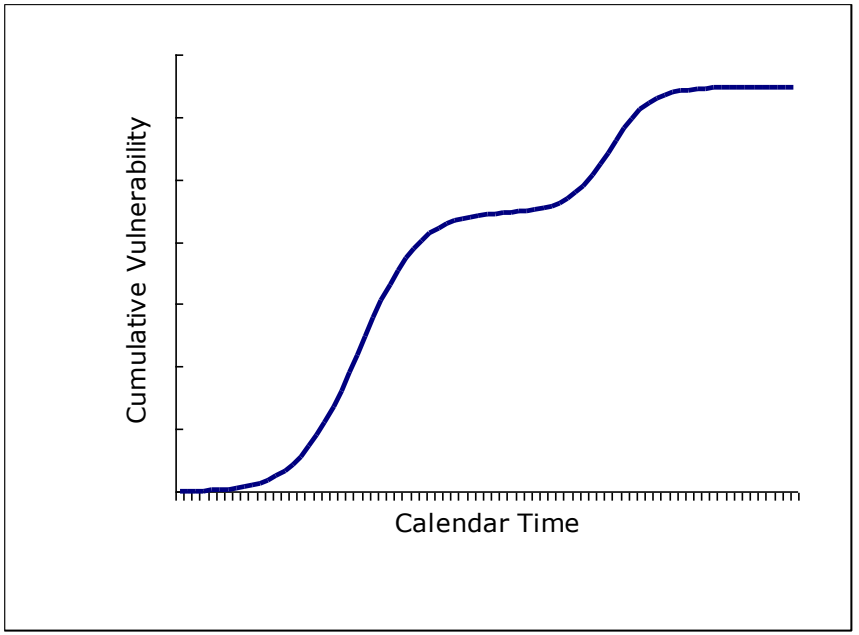
	Previous Version	Next Version	Shared Code Ratio α
Apache	1.3.24 (3-21-2002)	2.0.35 (4-6-2002)	79.84%
Mysql	4.1.1 (12-1-2003)	5.0.0 (12-22-2003)	83.52%

http://www.cs.colostate.edu/~malaiya/p/kim07.pdf

18

One vs Two Humps

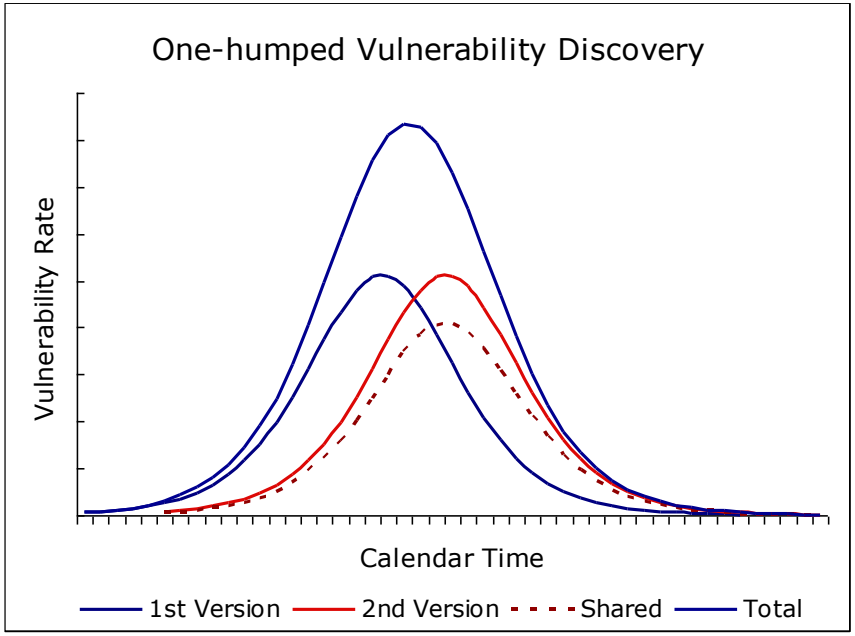


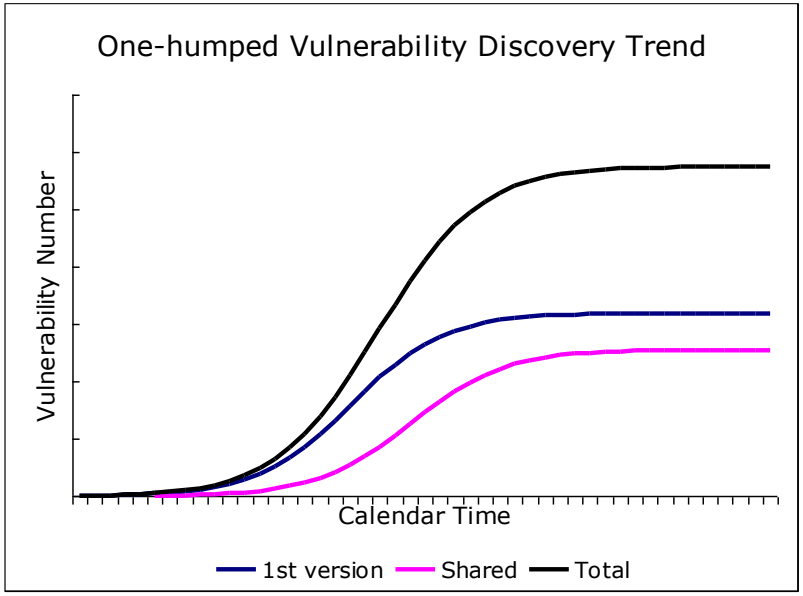


Superposition affect

19

Multi-version Vulnerability Discovery





- May result in a single hump with prolonged linear period

Evolving Programs

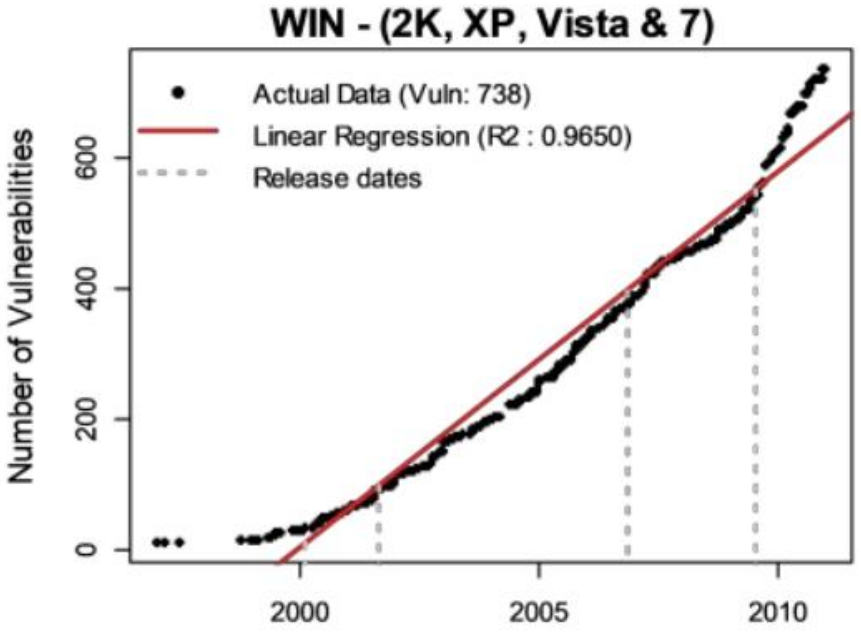
Gradually evolving software

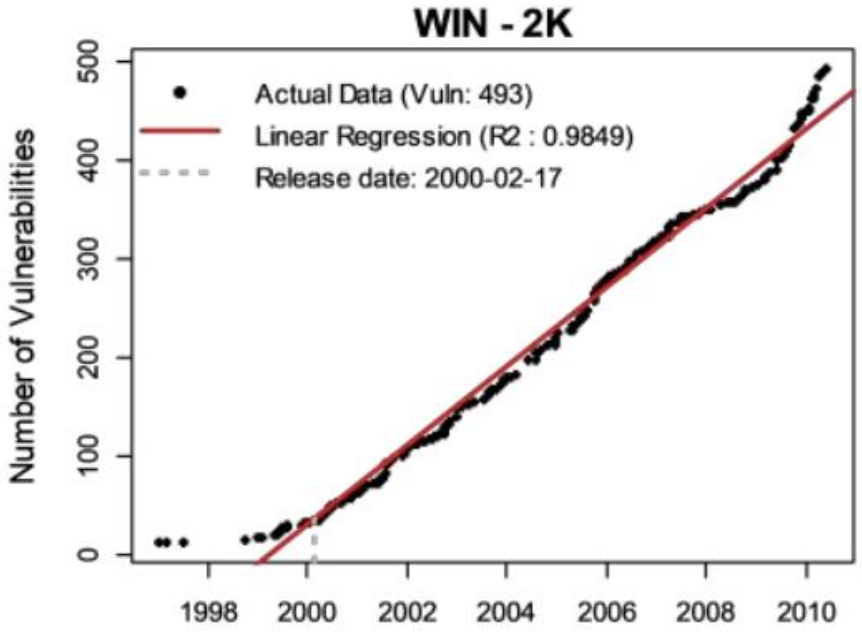
Software evolves in each version.

- 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.



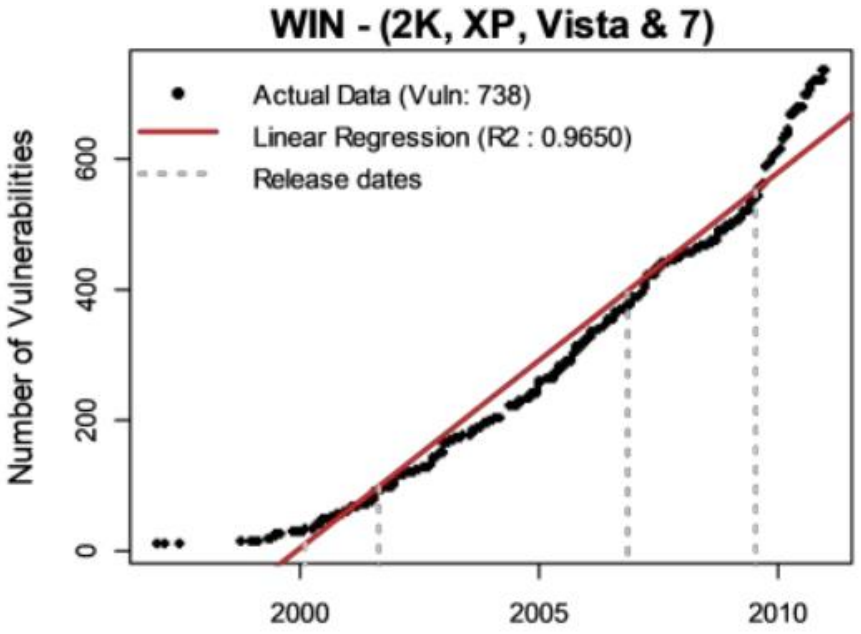


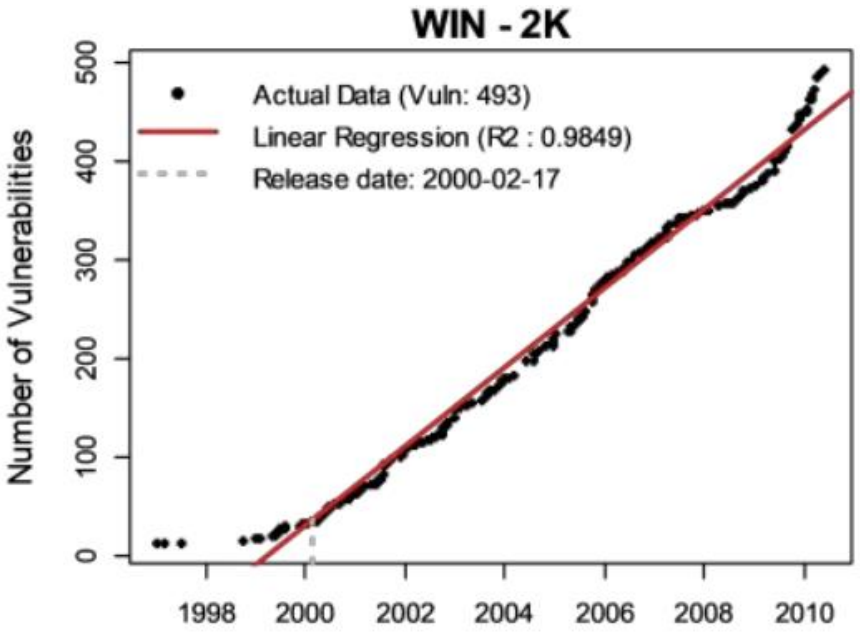
- If the evolution rate is steady, the size of the pool of undiscovered vulnerabilities stays the same (vulnerabilities removal rate = injection rate)
- If the market share is steady, the number of vulnerability finders remains steady.

Joh's thesis

22

Linear model





- 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

23

Examining Seasonality

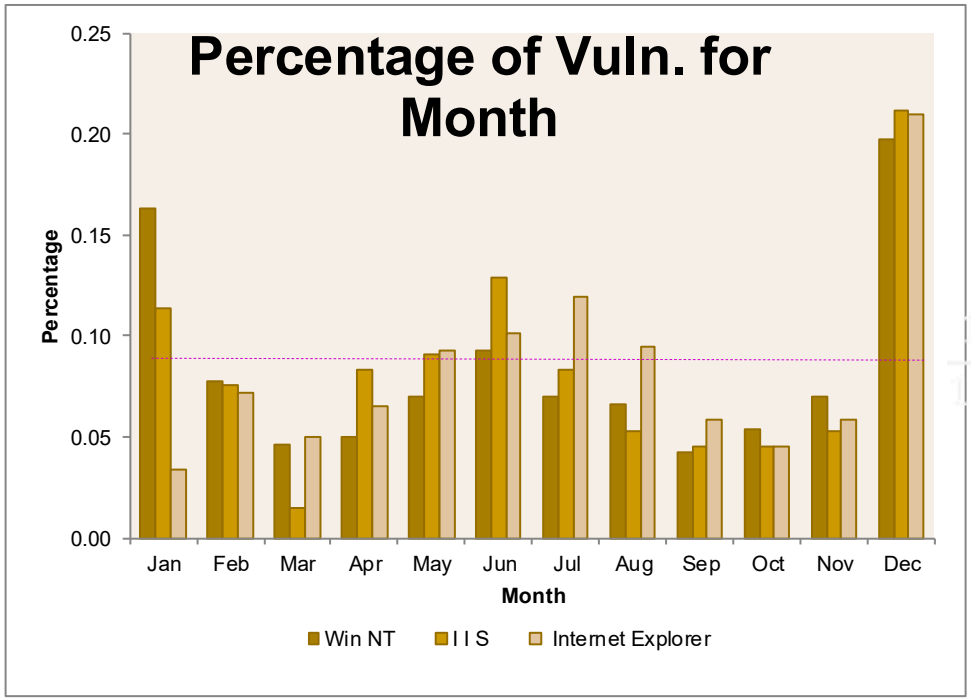
- Is the seasonal pattern statistically significant?
- Periodicity of the pattern
- Analysis:
  - Seasonal index analysis with test
  - Autocorrelation Function analysis
- Significance
  - Enhance VDMs' predicting ability
- Annual and Weekly seasonality

24

Annual: Prevalence in Month

Vulnerabilities Disclosed

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



25

Colorado State University

Why seasonality?

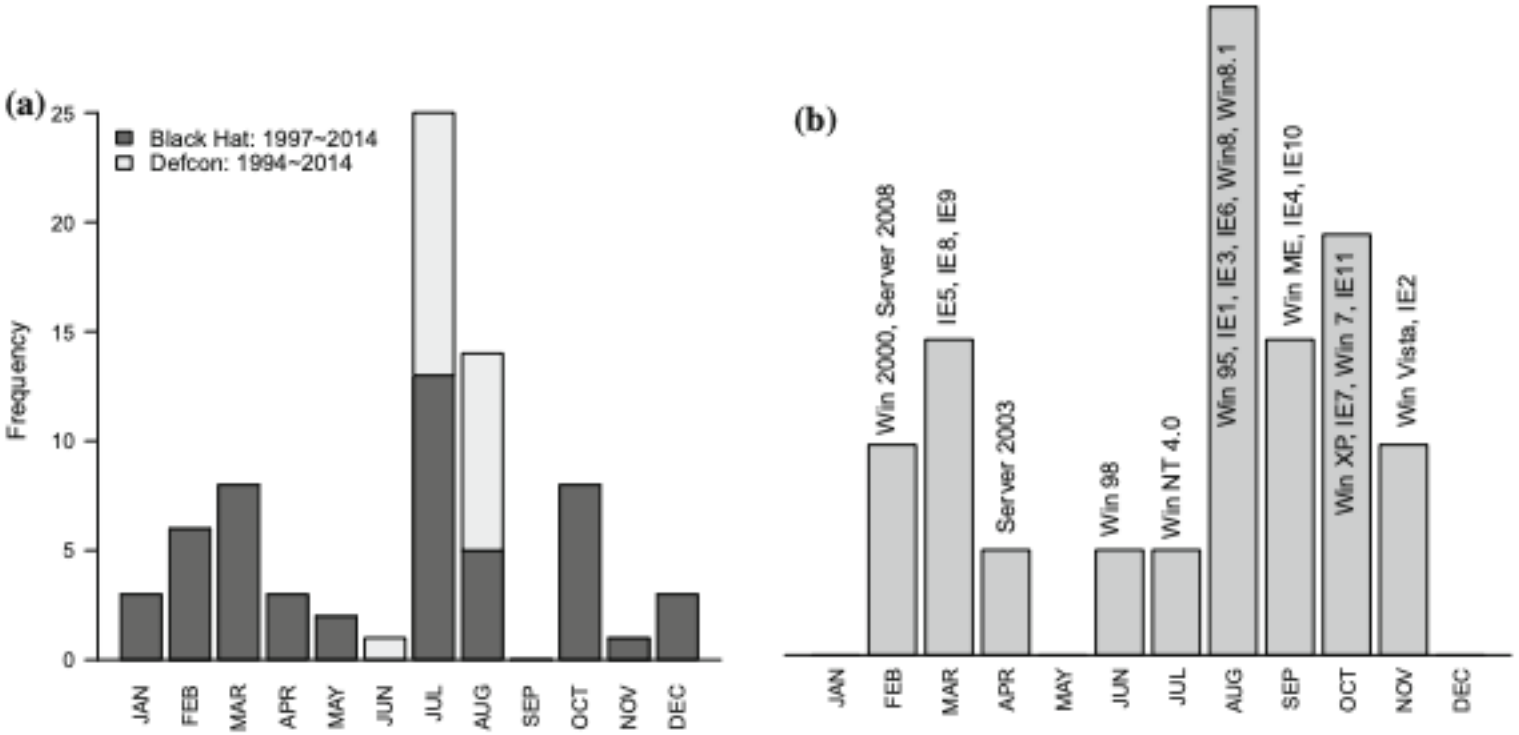


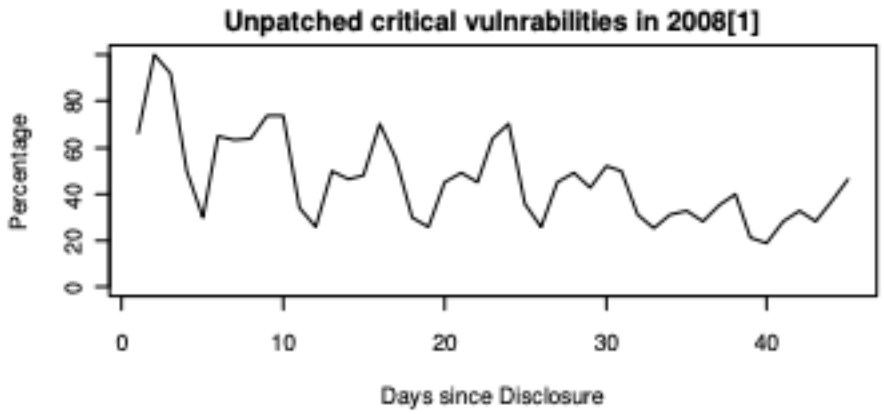
Fig. 6 Frequency of Black Hat and Defcon by month, and major Microsoft software system release time by month. **a** Black Hat and Defcon by month. **b** MS release by month

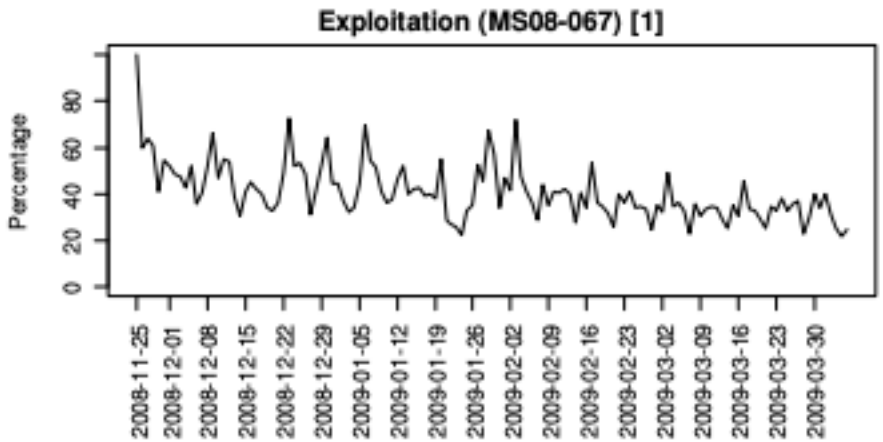
H. Joh and Y.K. Malaiya, "Periodicity in Software Vulnerability Discovery, Patching and Exploitation", International Journal of Information Security, July 2016, pp 1-18.

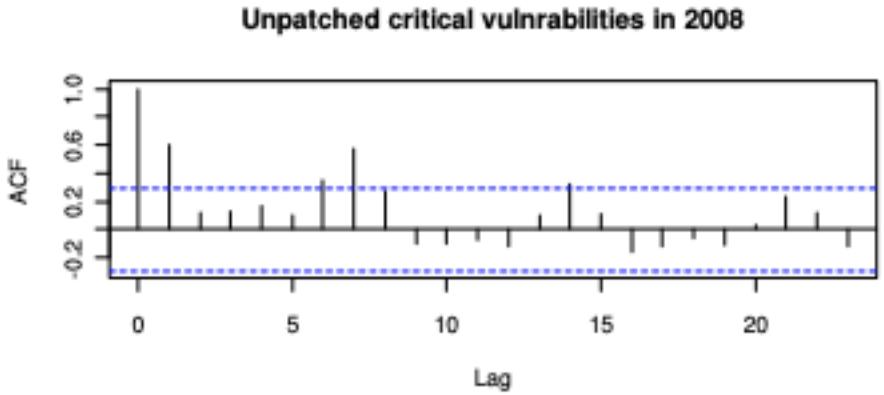
26

Colorado State University

Weekly Seasonality







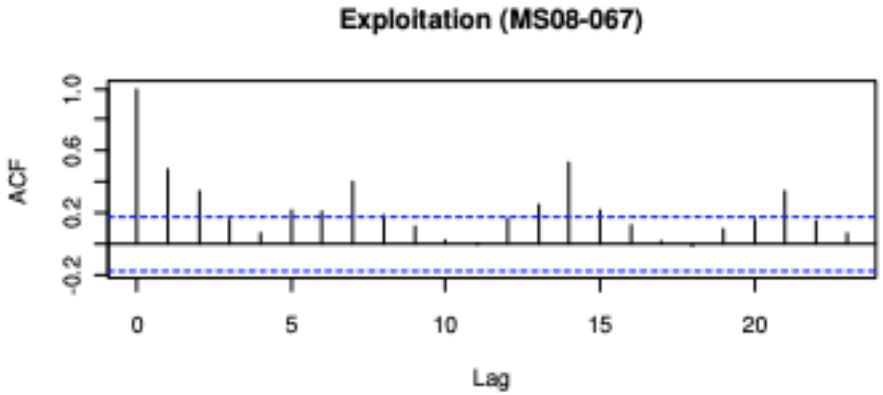
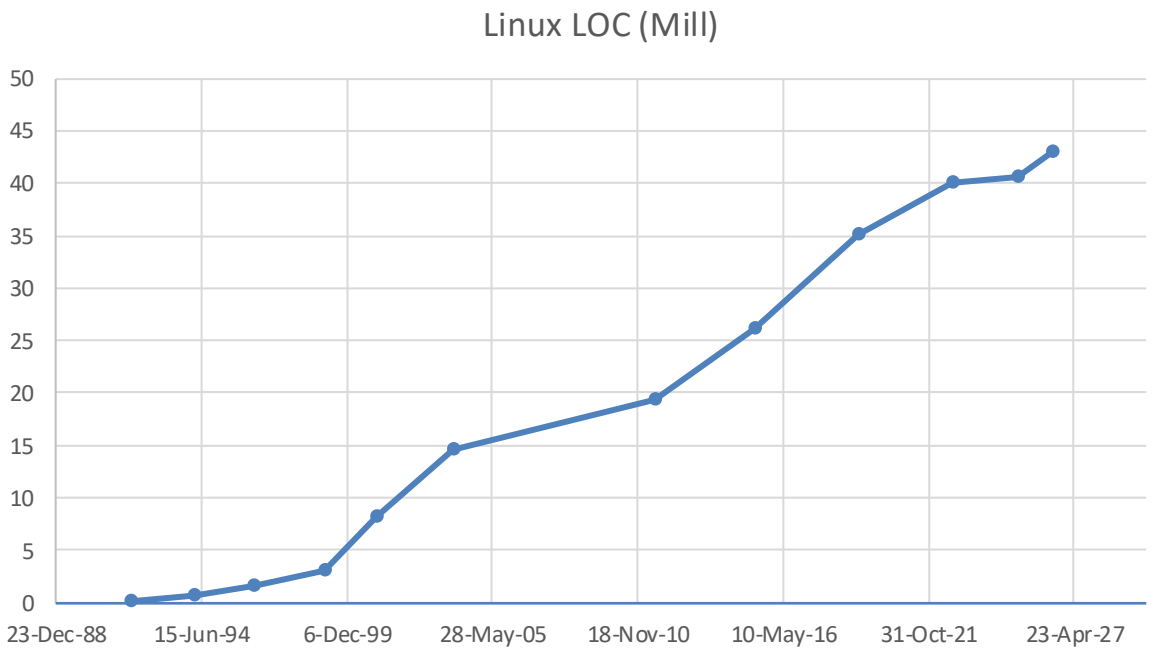


Figure 1. Run charts for unpatched critical vulnerabilities in 2008 and Exploitation with their corresponding ACFs. The upper two plots are normalized using the maximum value as 100%. In the bottom two plots, legs are in day.

H. Joh, S. Chaichana and Y. K. Malaiya, "Short-term Periodicity in Security Vulnerability Activity" Proc. Int. Symp. Software Reliability Eng. (ISSRE), FA, November 2010, pp. 408-409

27

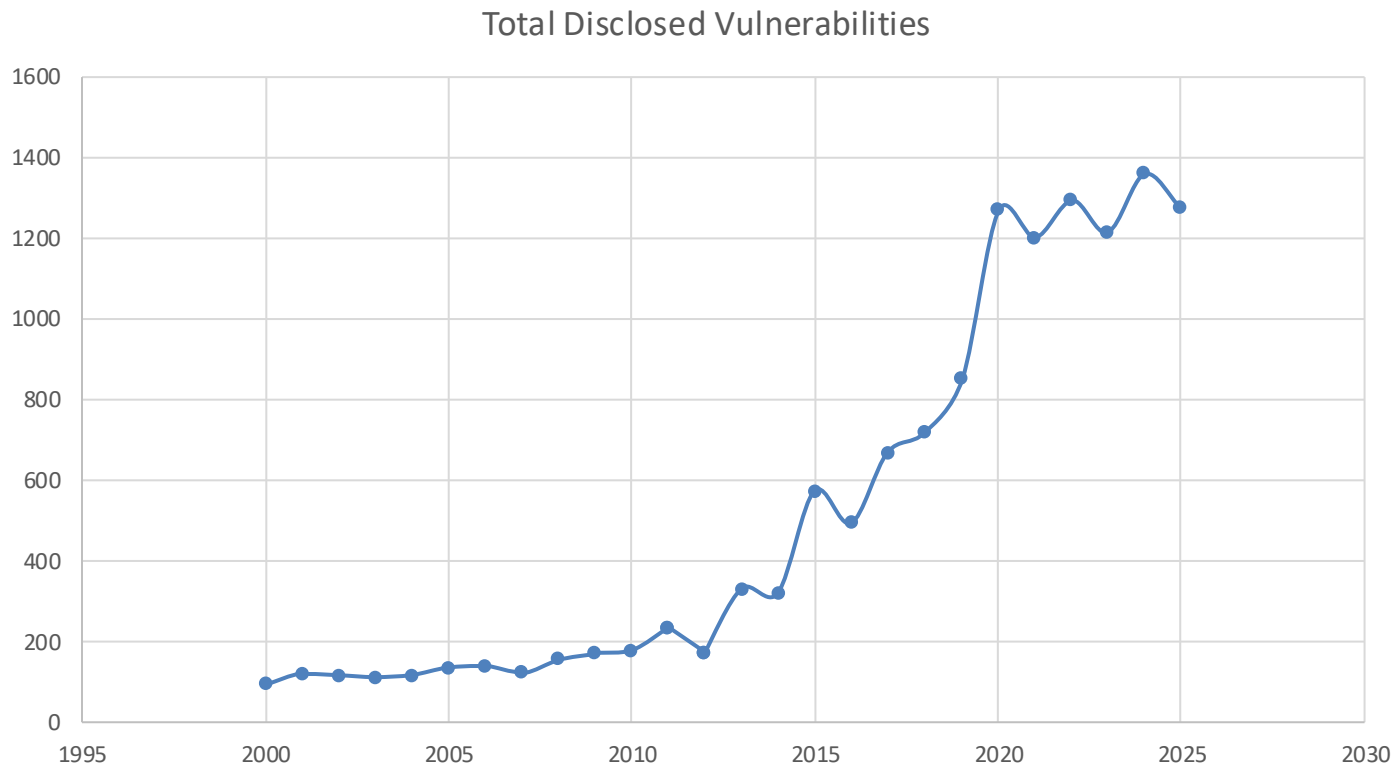
Long Term Trends: software sizes



Size evolution: Linux kernel size
Linux 1.0-7.2: Oct 1991-June 2026

28

Long Term Trends: Microsoft Vulnerabilites



Long term Trends: Total vulnerabilities, Microsoft products

Note: Covid19: became common in January 2020

29

Long term trends

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
- Increasing code sizes
  - Injection of new vulnerabilities

30

Vulnerability Discovery and Risks

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

31

Dimensions and Approximations

- For real problems, proper approximations are essential.
  - Elon Musk net worth is $269.8 Billion on Sept 30, 2024. Can be approximated as 270 Billion.
  - (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 65 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.

Quantitative Security

Colorado State University
Yashwant K Malaiya

CS 559
Security Metrics



CSU Cybersecurity Center
Computer Science Dept

To measure is to know.

James Clerk Maxwell, 1831-1879

Maxwell's equations for electromagnetism



Measurement motivates.

John Kenneth Galbraith. 1908-2006

Economist, Ambassador to India 1961-63



34

Metrics do matter

1. Metrics quantify the otherwise unquantifiable
2. Metrics can show trends and trends matter more than measurements do
3. Metrics can show if we are doing a good or bad job
4. Metrics can show if you have no idea where you are
5. Metrics establish where “You are here” really is
6. Metrics build bridges to managers
7. Metrics allow cross sectional comparisons
8. Metrics set targets
9. Metrics benchmark yourself against the opposition
10. Metrics create curiosity



Source: Andy Jaquith, Yankee Group, Metricon 2.0

35

Metrics

- Things that can be measures or predicted
  - Some metrics are more important than others
  - Some metrics can be correlated or redundant
    - Cyclomatic complexity and Lines of code
- Security metrics: contribute to determination of system security risk
- Classification by Pendleton et al
  - metrics of *system vulnerabilities*,
  - metrics of *defense strength*,
  - metrics of *attack (or threat) severity*,
  - metrics of *situation understanding*.

Marcus Pendleton, Richard Garcia-Lebron, Jin-Hee Cho, and Shouhuai Xu. 2016. A survey on systems security metrics. ACM Comput. Surv. 49, 4, Article 62 (December 2016), 35 pages. DOI: http://dx.doi.org/10.1145/3005714

Scales of Metrics

- **Nominal scale** is a naming scale, where variables are simply "named" or labeled, with no specific order.
  - Ex: blood types A, B, AB, O
- **Ordinal scale** has all its variables in a specific order, beyond just naming them.
  - Ex: “low income”, ”middle income”, ”high income”
- **Interval scale** offers labels, order, as well as a specific interval between each its variable options.
  - Ex: temperature (Fahrenheit)
- **Ratio scale** bears all the characteristics of an interval scale, in addition to that, it can also accommodate the value of "zero" on any of its variables.
  - Weight, money, temp. in Kelvin

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.



Colorado State University

Security Metrics

- Pendleton, Garcia-Lebron, Cho, and Xu. 2016. A survey on systems security metrics. Classification:
  - Enterprise system with n machines
  - *1. system vulnerabilities*, user, interface, software
  - *2. defense strength* proactive, reactive
  - *3. attack severity*, zero-day, targeted, botnets, malware etc
  - *4. situation (secure or compromised)* *internet, enterprise, breach rates etc*
  - Four sets of metrics is examined next. Search literature for details.

Terminology

- ***Enterprise systems*** refer to networked systems of multiple computers/devices, clouds, or even the entire cyberspace.
- ***Computer systems*** represent individual computers/devices. We interchangeably use the terms node, device, or computer to refer to a single entity.
- **Attackers:** These are *attacking entities* representing computers or IP addresses from which cyber attacks are launched against other normal entities.
- **Incident:** It represents a successful attack (e.g., malware infection or data breach).

Attack-defense interactions in an enterprise system at time *t*

At time *t*, the enterprise system consists of *n* entities (i.e., computers), denoted by the vector

$C(t) = \{c1(t), \ldots, cn(t)\}$

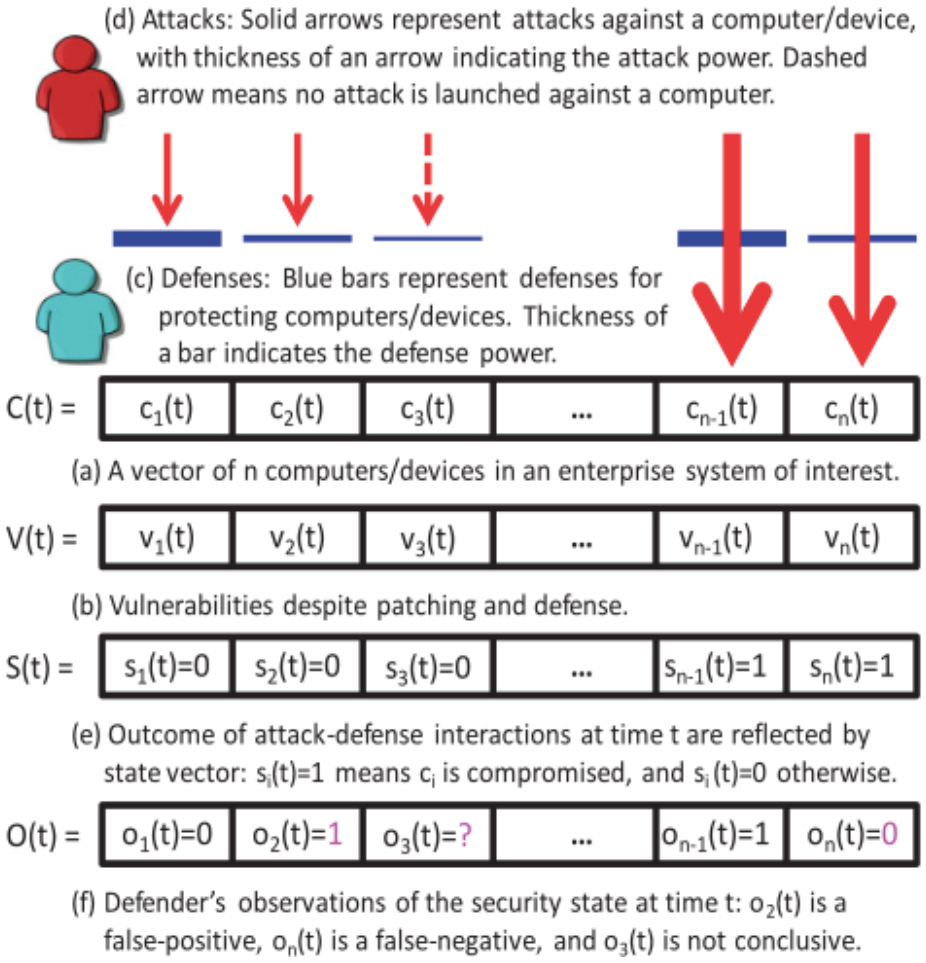
n may vary with time.

Each entity, $ci(t)$, has a vector $vi(t)$ of vulnerabilities, such as zero-day and/or some unpatched software vulnerabilities.

The outcome of the attack-defense interaction reflected by a global **security state vector**

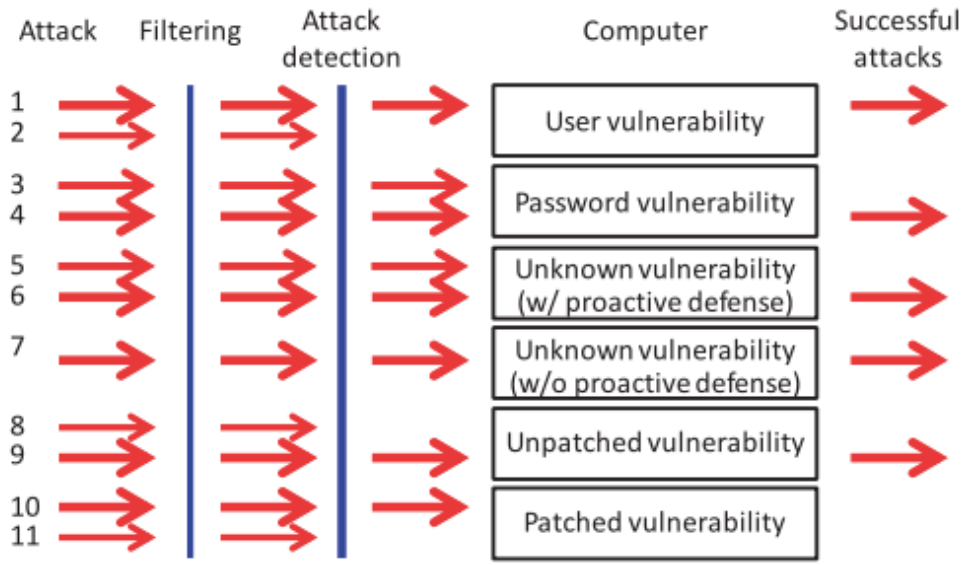
$S(t) = \{s1(t),\ldots,sn(t)\}$,

where $si(t) = 0$ means entity $ci(t)$ is secure at time *t* and $si(t) = 1$ means entity $ci(t)$ is compromised at time *t*



Marcus Pendleton, Richard Garcia-Lebron, Jin-Hee Cho, and Shouhuai Xu. 2016. A survey on systems security metrics. ACM Comput. Surv. 49, 4, Article 62 (December 2016), 35 pages. DOI: http://dx.doi.org/10.1145/3005714

Attack-defense interactions in a computer (or device), *ci* (*t*)



Filtering: mechanisms deployed at the enterprise system perimeter to block traffic from malicious or blacklisted IP addresses,
Use of some ***attack detection*** mechanisms to detect and block attacks before they reach *ci* (*t*),
Use of proactive defense mechanisms (e.g., address space randomization) to mitigate vulnerabilities exploitable by attackers (i.e., exploitability).

Situations

- *Situation*(t)= f(V(t), D(t), A(t))
  - where V(t) is a function of *vulnerabilities* at time t,
  - D(t) is a function of *defenses* at time t, and
  - A(t) is a function of *attacks* at time t.
- S(t) is naturally affected by V (t), D(t), and A(t) as well.

Security Metrics

- Most researchers have focused on a specific component of he overall risk and have used various metrics for their work.
  - A single component is not the whole risk.
  - $Risk_i = Likelihood_i \times Impact_i$
  - $Likelihood_i = P\{\text{A security hole}_i \text{ is exploited}\}$.

    $= P\{\text{hole}_i \text{ present}\}.\ P\{\text{exploitation} | \text{hole}_i \text{ present}\}$
- Pendleton, Garcia-Lebron, Cho, and Xu. 2016. A survey on systems security metrics.
  - *1. system vulnerabilities*,
  - *2. defense strength and 3. attack severity*,
  - *4. situation*
- Four set of metrics is examined next.