Computer Systems Security
CSE 628A

Pramod Subramanyan
Indian Institute of Technology Kanpur

ADMINISTRIVIA

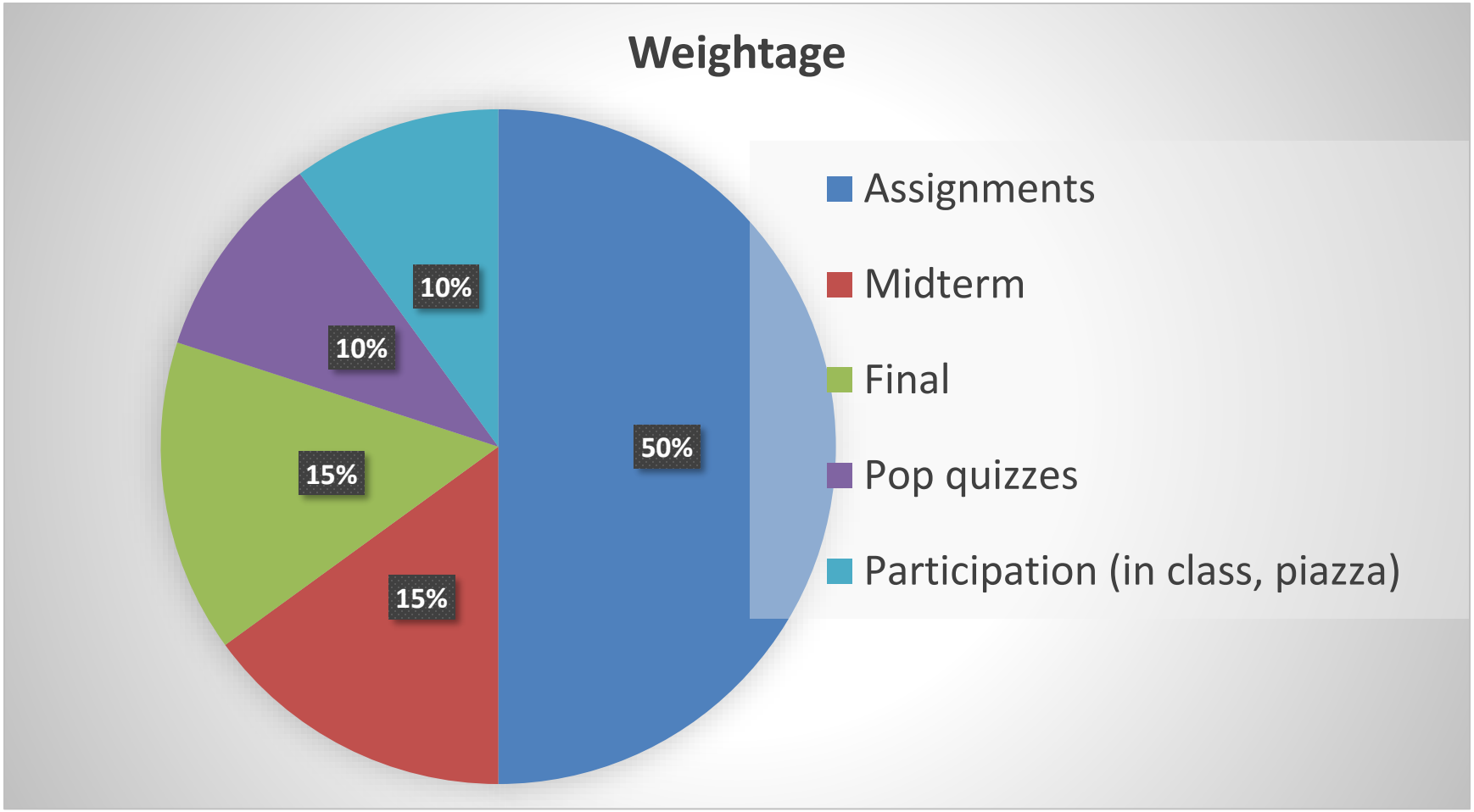
Team

- Instructor: Pramod Subramanyan (spramod@cse)
- Teaching Assistants:
  - Deepak Sirone J (dsirone@cse)
  - Fadadu Fenilkumar Chetanbhai (fenilk@cse)
  - Harsh Bhagwani (harshb@cse)
  - Mohd Arshalaan Hameed (ahameed@cse)
  - Santhosh Kumar T (santkum@cse)
  - Saurabh Kumar (skmtr@cse)

Links

- Piazza signup link:
  - https://piazza.com/iitk.ac.in/secondsemester2019/cs628
- Moodle course:
  - https://moodle.cse.iitk.ac.in/course/view.php?id=139
  - Assignments will be posted on moodle
- Course webpage:
  - https://web.cse.iitk.ac.in/users/spramod/courses/cs628-2019/
  - Slides and readings will be posted under schedule

Grading


Weightage
50%
15%
15%
10%
10%
Assignments
Midterm
Final
Pop quizzes
Participation (in class, piazza)

Expectations/Advice

1. Ensure you have the background knowledge
2. Come to the classes and participate
3. Study after each class
4. Post and answer questions on piazza
5. Actually read the readings
6. Start the homeworks early

Remember, your goal is to learn the material

Background/Preparation

- Computer organization (assembly language, TLBs, demand paging, privilege separation)
- Operating systems (processes, threads, heaps, stacks, page tables, permissions, etc.)
- Networks (IP, TCP, UDP, basic knowledge of SSL/TLS, BGP, etc.)
- Ability to read, write and understand programs

Pop Quiz 0

- Will be posted on Moodle tonight
- Deadline for submission in 24 hours
- Should not take you more than 10 minutes
- You get 100% of the points if you answer at least one question correctly

Module 0: Introduction

Context and Landscape

Acknowledgements

- Sandeep Shukla (IIT Kanpur)
- Arvind Narayanan (Princeton University)
- Dan Boneh (Stanford University)
- John C. Mitchell (Stanford University)
- Nicolai Zeldovich (MIT)
- Jungmin Park (Virginia Tech)
- Patrick Schaumont (Virginia Tech)
- Web Resources

The computer security problem

Two factors:

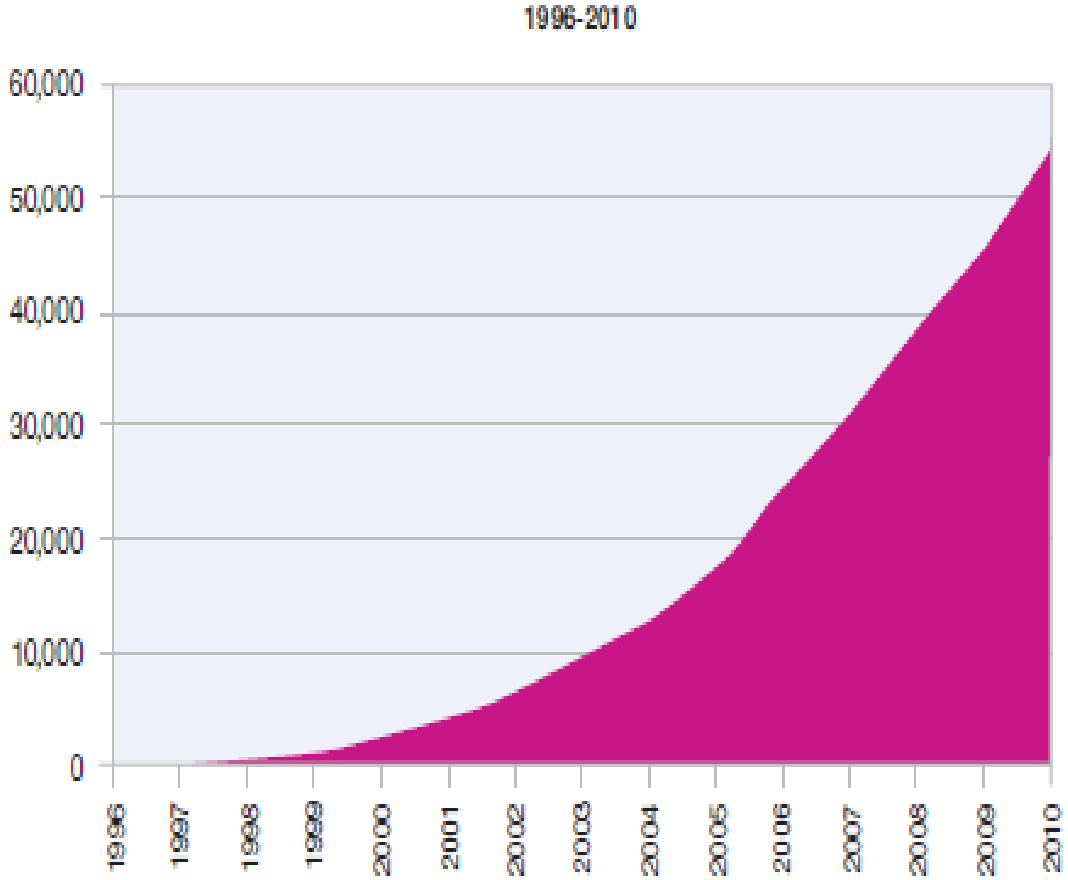
- **Lots of buggy software** (and gullible users)
- **Money can be made from finding and exploiting vulnerabilities**

1. Marketplace for vulnerabilities
2. Marketplace for owned machines
3. Methods to profit from owned client machines

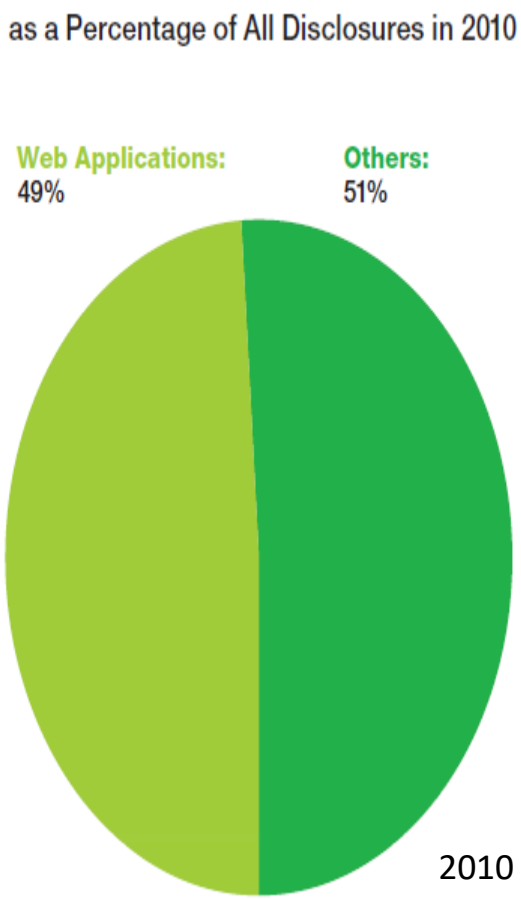
current state of computer security

MITRE tracks vulnerability disclosures

Cumulative Disclosures



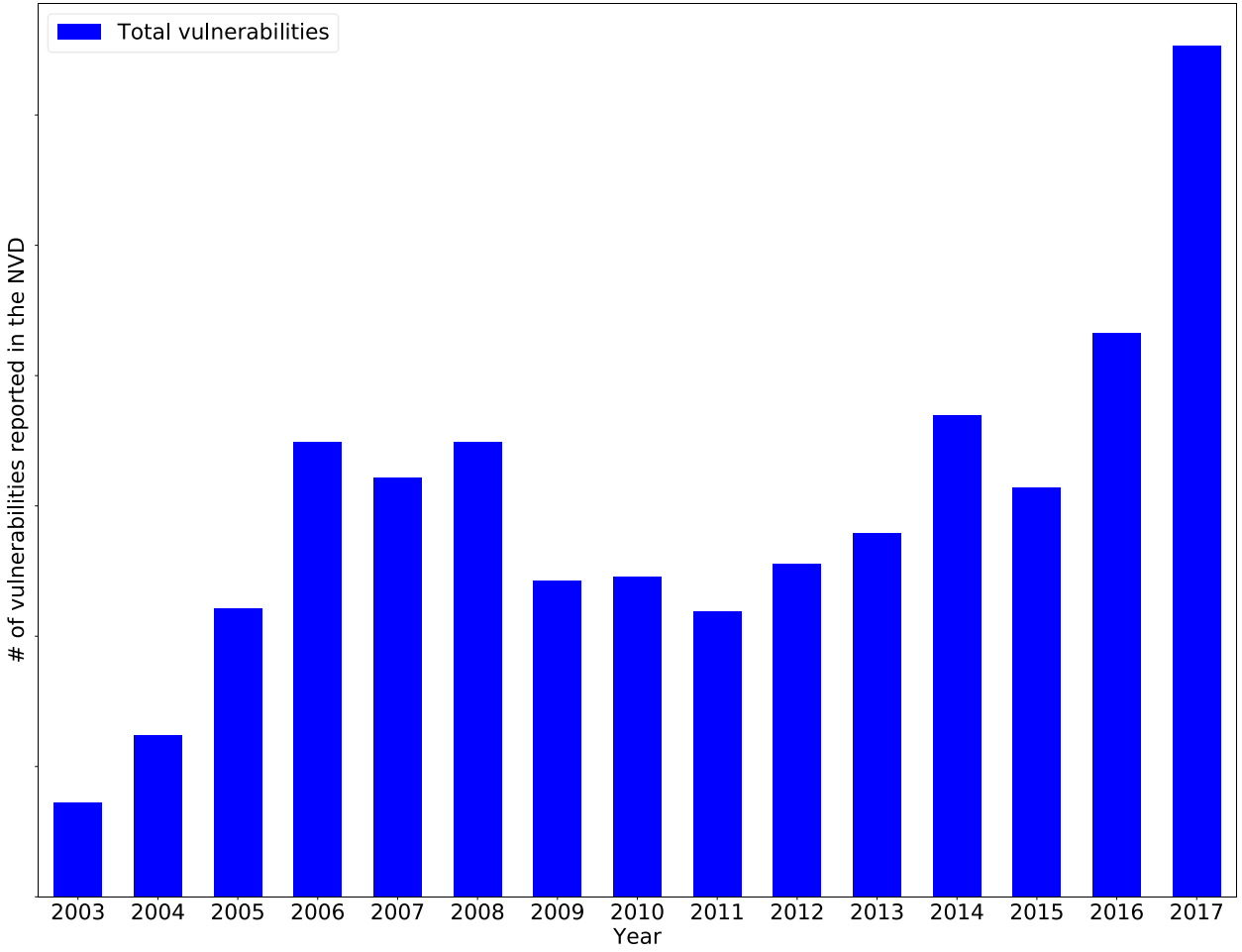
Percentage from Web applications



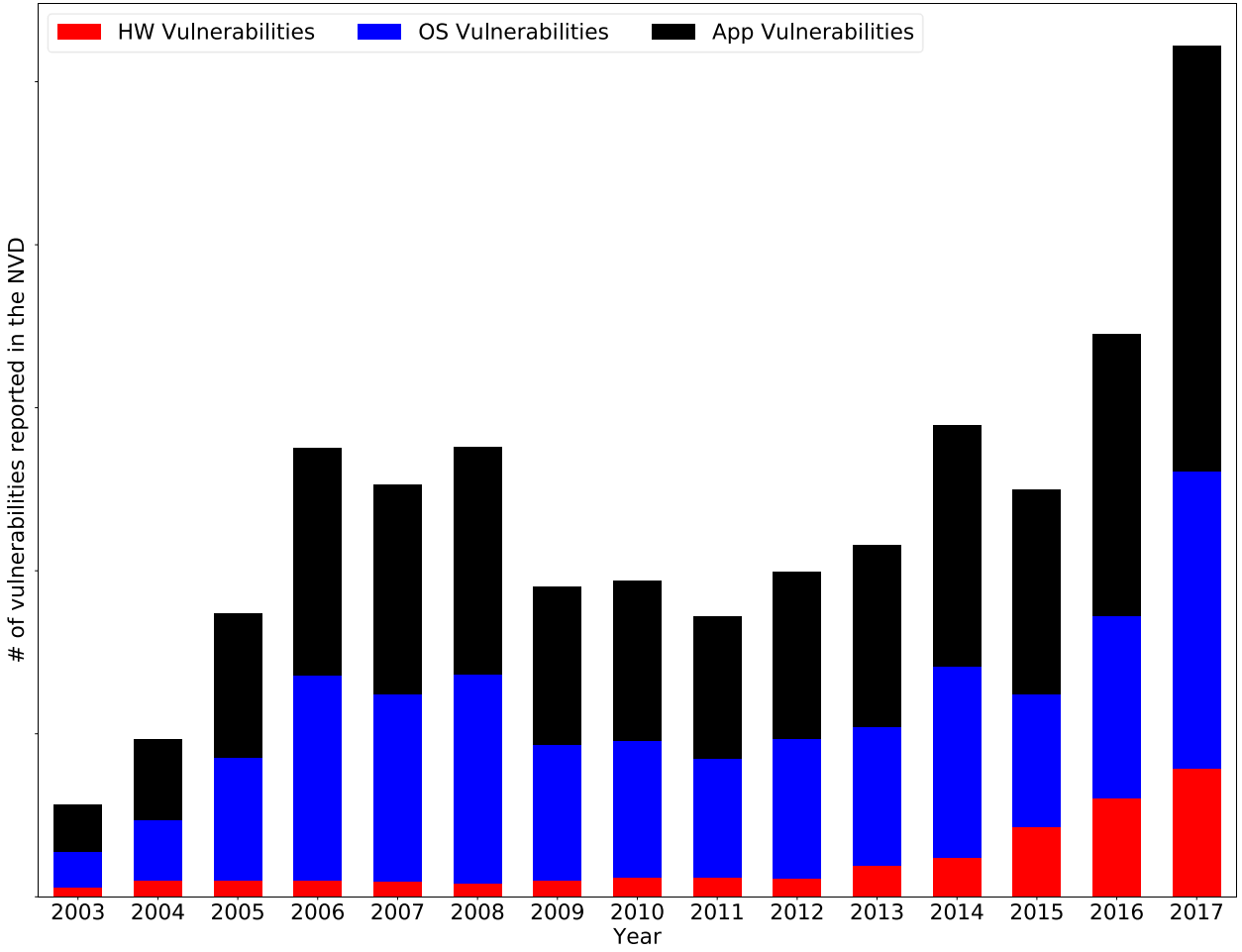
Source: IBM X-Force, Mar 2011

Data: http://cve.mitre.org/

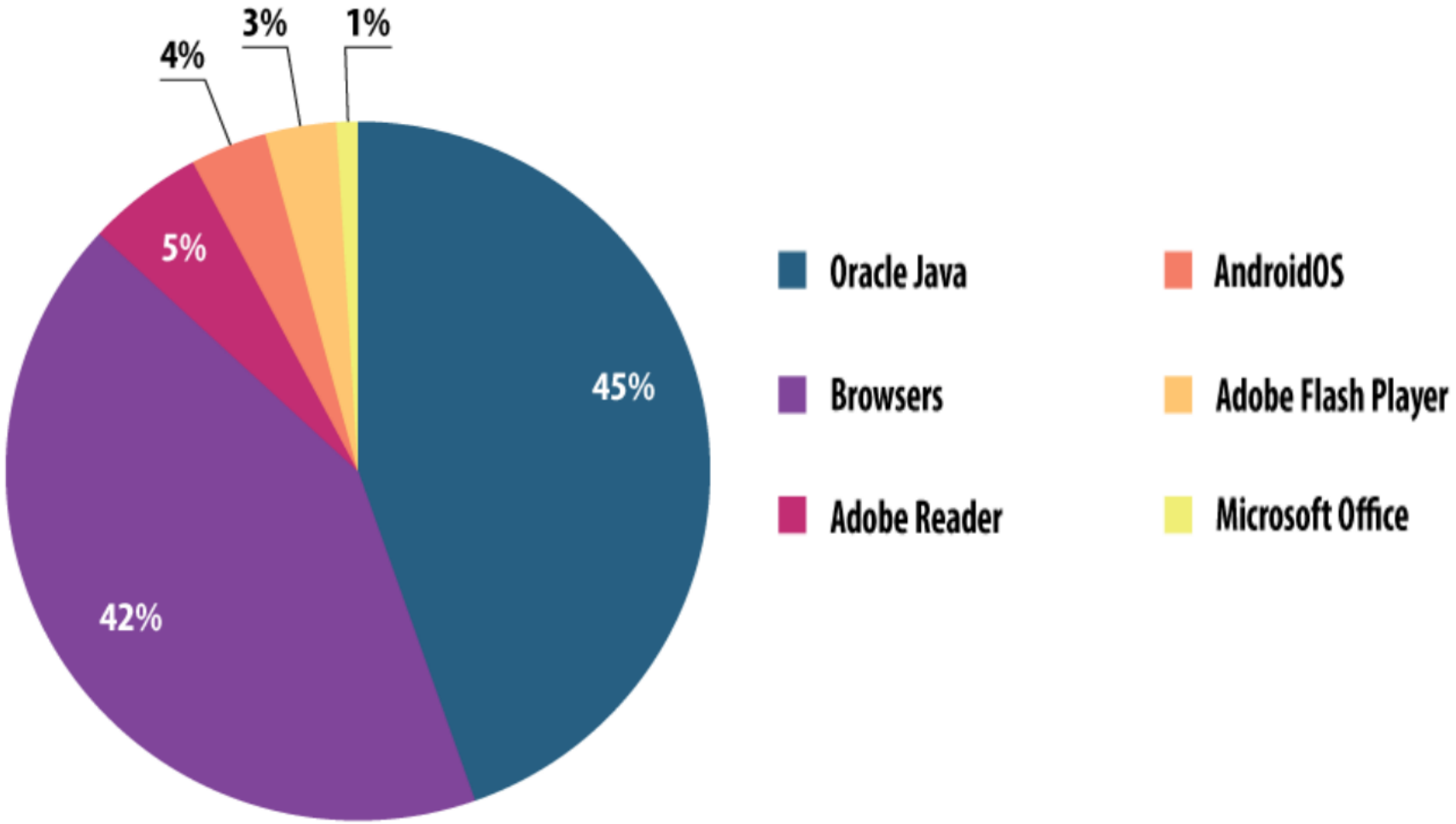
CVEs in the NVD



CVEs in the NVD

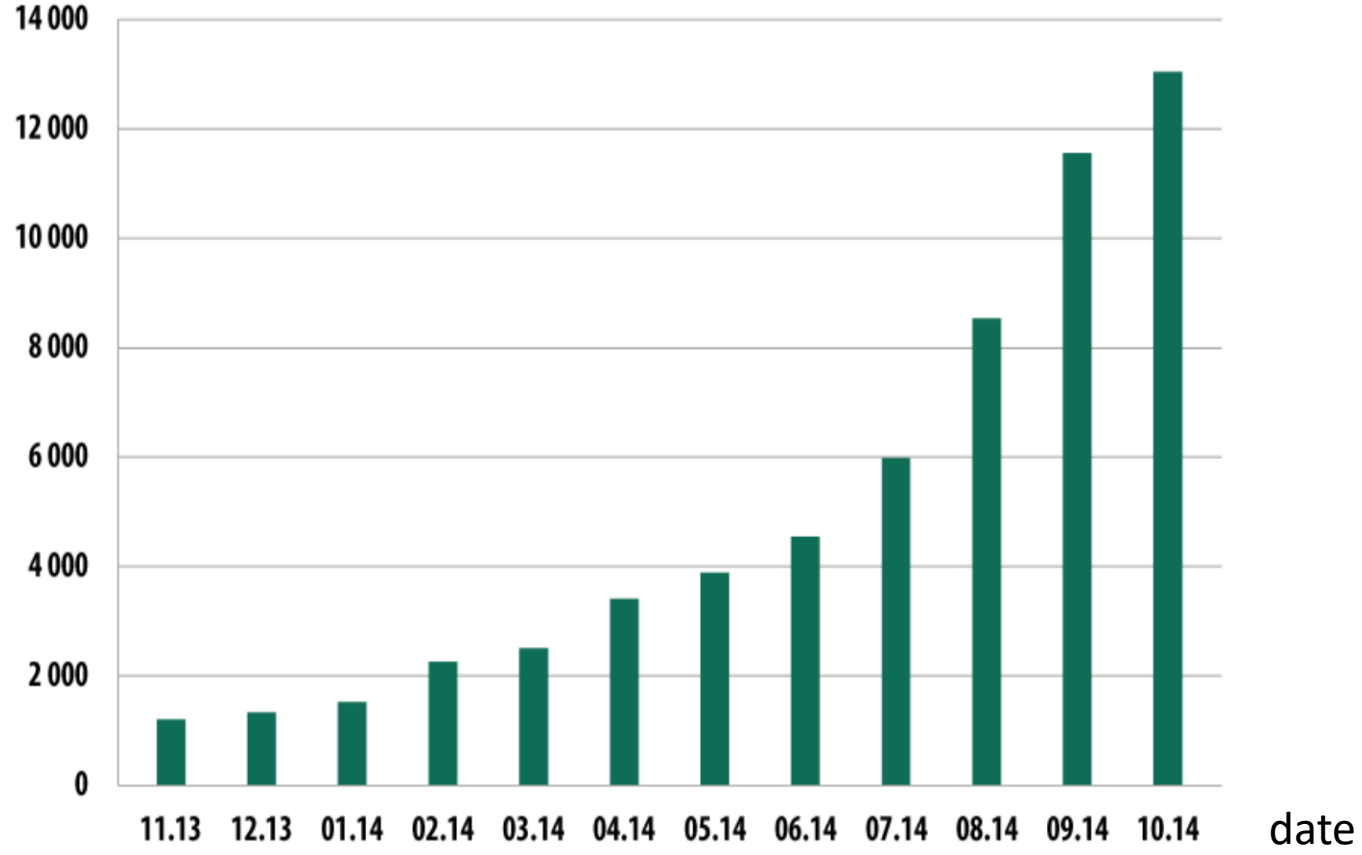


Vulnerable applications being exploited



Source: Kaspersky Security Bulletin 2014

Mobile malware (Nov. 2013 – Oct. 2014)



The rise of mobile banking Trojans (Kaspersky Security Bulletin 2014)

Introduction

Sample attacks

The computer security problem

Two factors:

- **Lots of buggy software** (and gullible users)
- **Money can be made from finding and exploiting vulnerabilities**

1. Marketplace for vulnerabilities
2. Marketplace for owned machines (PPI)
3. Methods to profit from owned client machines

Why own machines: (1) IP address and bandwidth stealing

Attacker's goal: look like a random Internet user

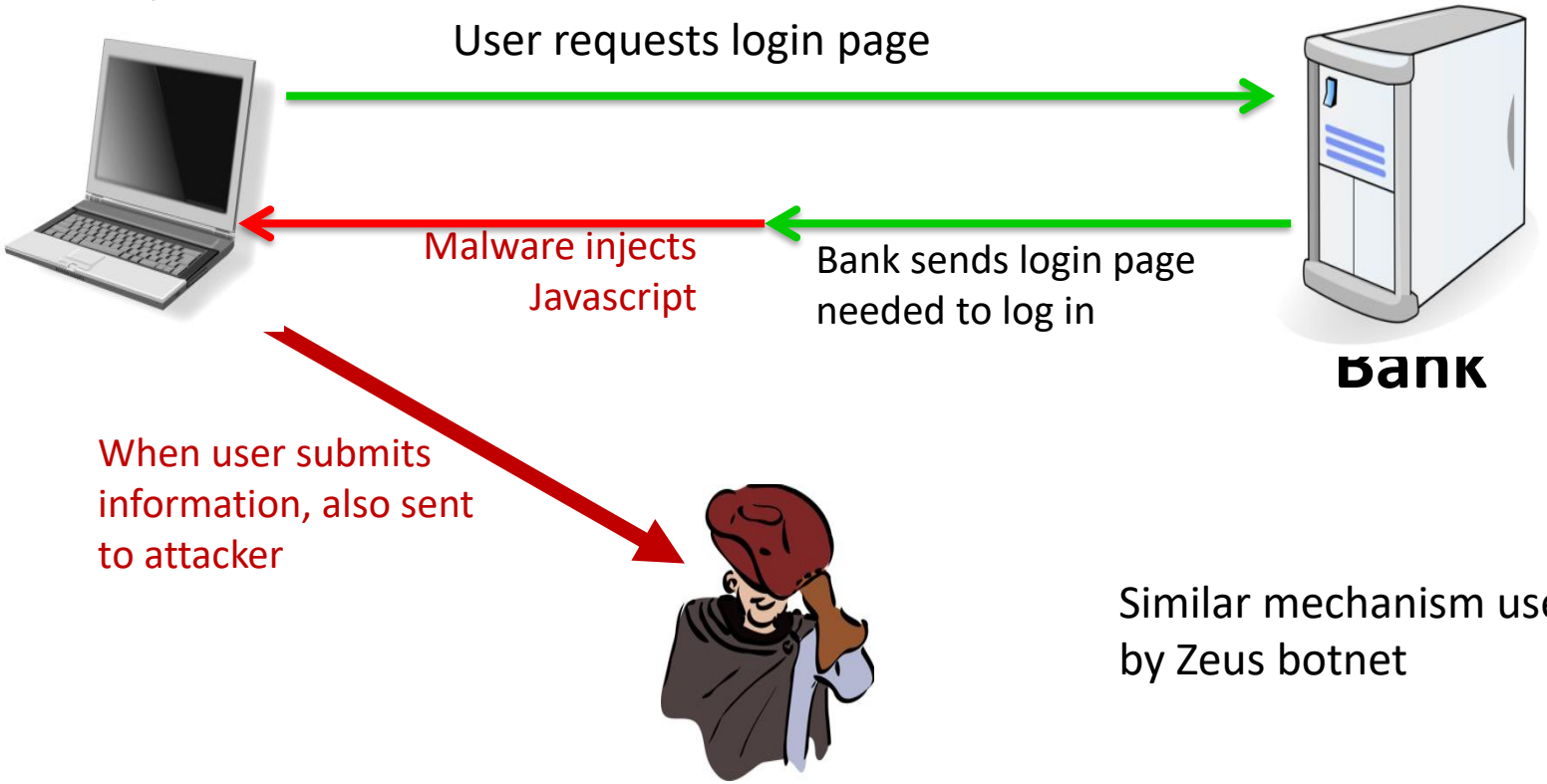
Use IP address of infected machine or phone for:

- **Spam** (e.g. the storm botnet)
  - 1:12M pharma spams leads to purchase
  - 1:260K greeting card spams leads to infection
- **Denial of Service:** Services: 1 hour ($20), 24 hours (100$)
- **Click fraud** (e.g. Clickbot.a)

Why own machines: (2) Steal user credentials and inject ads

keylog for banking passwords, web passwords, gaming pwds.

Example: SilentBanker (and many like it)



Similar mechanism used by Zeus botnet

Server-side attacks

- Financial data theft: often credit card numbers
  - Example: Target attack (2013), ≈ 140M CC numbers stolen
  - Many similar (smaller) attacks since 2000
- Political motivation:
  - Aurora, Tunisia Facebook (Feb. 2011), GitHub (Mar. 2015)
- Infect visiting users

Example: Mpack

- PHP-based tools installed on compromised web sites
  - Embedded as an iframe on infected page
  - Infects browsers that visit site
- Features
  - Management console provides stats on infection rates
  - Sold for several hundred dollars
  - Customer care can be purchased, one-year support contract
- Impact: 500,000 infected sites (compromised via SQL injection)
  - Several defenses: e.g. Google safe browsing

Insider attacks: example

Hidden trap door in Linux (nov 2003)

- Allows attacker to take over a computer
- Practically undetectable change (uncovered via CVS logs)

Inserted line in wait4()

```
if ((options == (__WCLONE|__WALL)) && (current->uid = 0))
                retval = -EINVAL;
```

Looks like a standard error check, but …

See: http://lwn.net/Articles/57135/

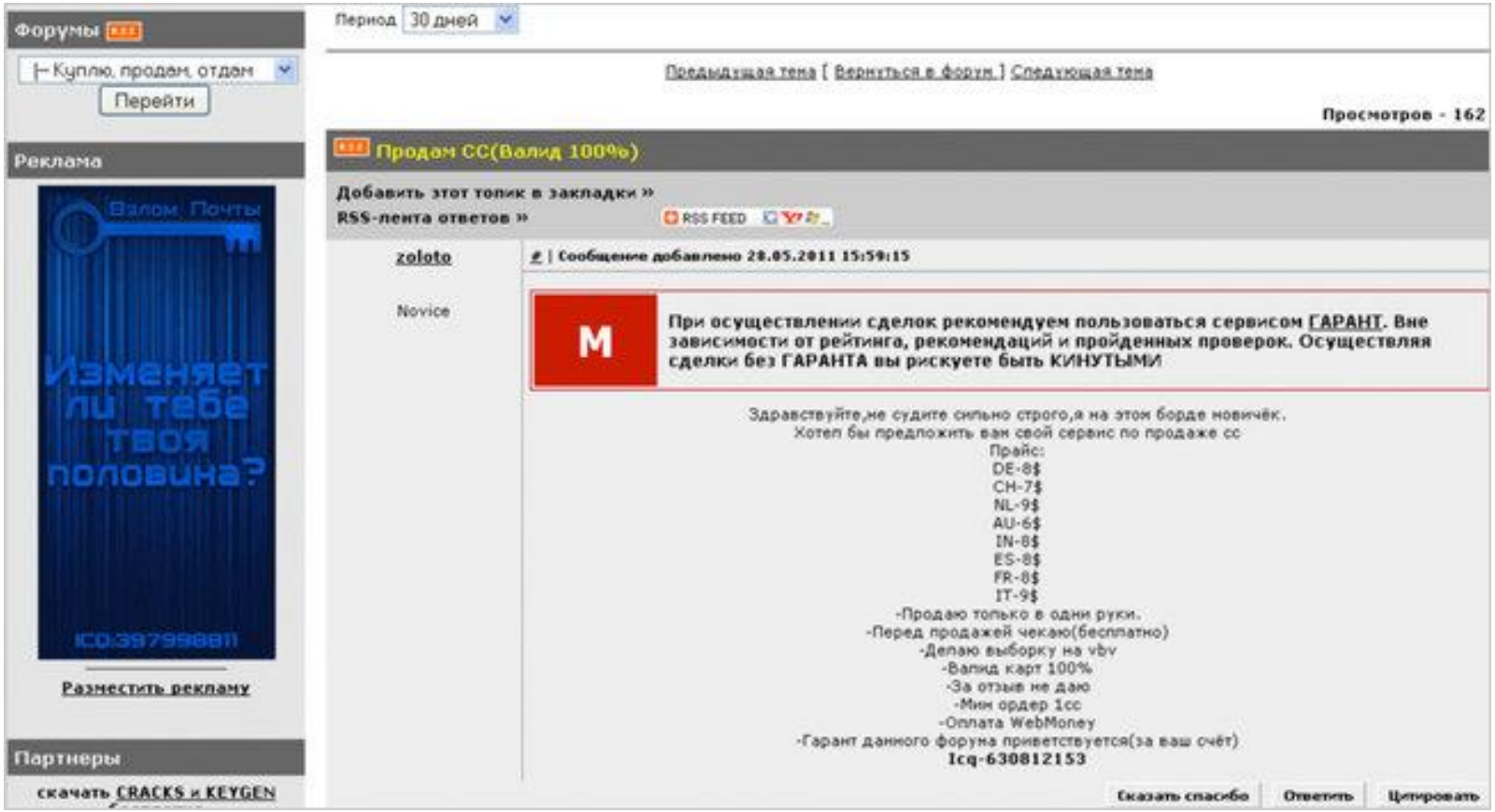
Many more examples

- Access to SIPRnet and CD-RW: 260,000 cables ⇒ Wikileaks
- SysAdmin for city of SF government. Changed passwords, locking out city from router access
  - https://www.cio.com.au/article/255165/sorting_facts_terry_childs_case/?pp=4&fp=&pf=1&fpid=
- Inside logic bomb took down 2000 UBS servers
  - https://www.theregister.co.uk/2006/12/13/ubs_logic_bomber_sentenced/

Introduction

The Marketplace for Vulnerabilities

Hacker zoloto offered credit cards for sale on the Web site HackZone .ru.



Marketplace for Vulnerabilities

Option 1: bug bounty programs (many)

- Google Vulnerability Reward Program: up to $100K
- Microsoft Bounty Program: up to $100K
- Mozilla Bug Bounty program: $500 - $3000
- Pwn2Own competition: $15K

Option 2:

- ZDI : $2K – $25K

Marketplace for Vulnerabilities

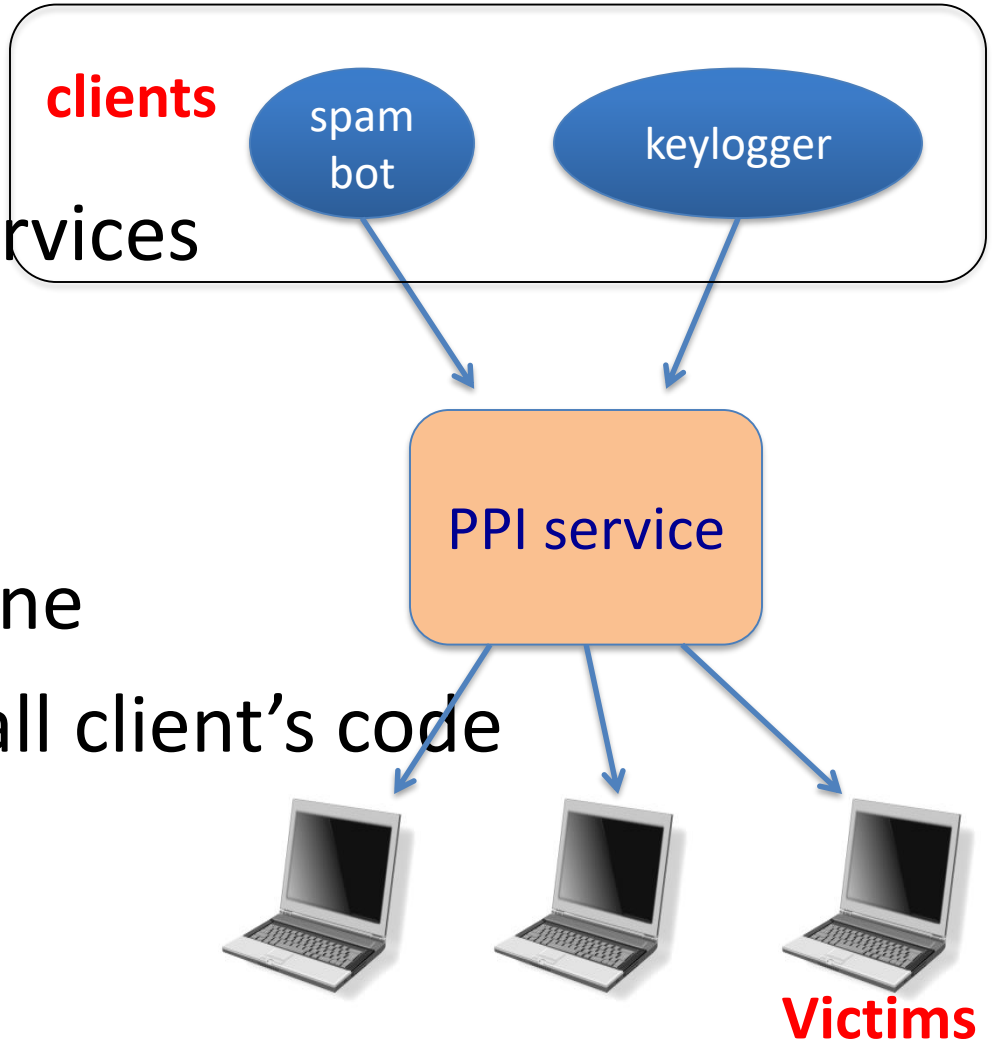
Option 3: Black Market

ADOBE READER	$5,000–$30,000
MAC OSX	$20,000–$50,000
ANDROID	$30,000–$60,000
FLASH OR JAVA BROWSER PLUG-INS	$40,000–$100,000
MICROSOFT WORD	$50,000–$100,000
WINDOWS	$60,000–$120,000
FIREFOX OR SAFARI	$60,000–$150,000
CHROME OR INTERNET EXPLORER	$80,000–$200,000
IOS	$100,000–$250,000

Source: Andy Greenberg (Forbes, 3/23/2012)

Marketplace for owned machines

Pay-per-install (PPI) services

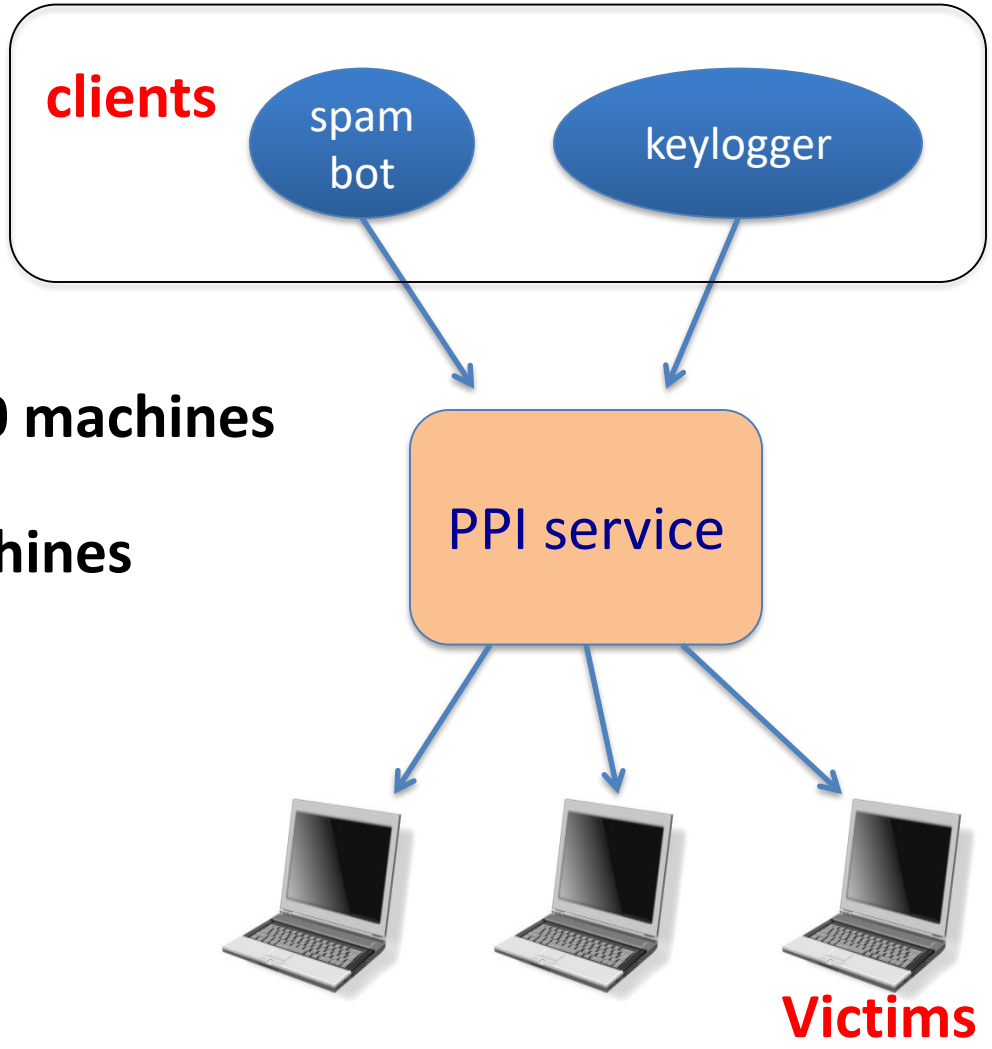


PPI operation:

1. Own victim's machine
2. Download and install client's code
3. Charge client

Source: Cabalerro et al. (www.icir.org/vern/papers/ppi-usesec11.pdf)

Marketplace for owned machines



Cost: **US - $100-180 / 1000 machines**

Asia - $7-8 / 1000 machines

Source: Cabalerro et al. (www.icir.org/vern/papers/ppi-usesec11.pdf)

This course

Goals:

- Be aware of exploit techniques
- Learn to defend and avoid common exploits
- Learn to architect secure systems

This Course

Part 1: Basics (architecting for security)

- Securing apps, OS, and legacy code
  Isolation, authentication, and access control

Part 2: Web security (defending against a web attacker)

- Securing websites, browser security model

Part 3: Network security (defending against a network attacker)

- Monitoring and architecting secure networks.

Part 4: Mobile security

Part 5: Hardware Security (SGX, Hardware Trojans)

Part 6: Distributed Systems Security (PBFT, Consensus)

Ken Thompson’s clever Trojan

Ken Thompson, co-author of UNIX, recounted a story of how he created a version of the C compiler that, when presented with the source code for the "login" program, would automatically compile in a backdoor to allow him entry to the system.

This is only half the story, though. In order to hide this trojan horse, Ken also added to this version of "cc" the ability to recognize if it was recompiling itself to make sure that the newly compiled C compiler contained both the "login" backdoor, and the code to insert both trojans into a newly compiled C compiler. In this way, the source code for the C compiler would never show that these trojans existed.

What is Security?

- Achieving something in the presence of adversaries
  - Internet is full of adversaries
  - There are insider adversaries for air-gapped systems
  - Thus design of systems need to worry about security

- A High Level Plan for Security Centric System Design
  - Policy: “Only X can access file F”
  - Common goals: Confidentiality, Integrity, Availability
  - Threat Models: “Can Y physically grab the file server?”
  - Mechanisms: The knobs that can be controlled to uphold your security policy, but also be flexible to uphold a different policy
  - Resulting Goal: “No way the adversary in the threat model to violate policy”

Why is security hard?

- Need to guarantee policy, assuming threat models
- Difficult to think of all possible ways that attacker might break in
- Realistic threat models are open-ended (Negative models)
- Easy to check a positive goal ("X has access to File F")
- Weakest link matters
- Iterative process: Design, Update Threat Model as necessary, assess vulnerability → Design, Update …..

What if perfect security is not achievable?

- Best effort
- Each system will have some breaking point – need to analyze and understand – e.g., penetration testing
- Need to manage security risk vs. benefit tradeoff
- Risk based security model
- Manual auditing often can help
- Make the cost of attack high – deterrence
  - Either by law
  - Technologically

Revisiting Clickfraud

- Google ad cost is PPC (pay per click)
- Google shares some of this revenue with website which generated (also PPC)
- If you are a website operator, what do you do?
- Get fake clicks using botnets
- Fabio Gasperini: Clickfraud prosecution in US

Perfect Security is not achievable

- Best effort
- Each system will have some breaking point – need to analyze and understand – e.g., penetration testing
- Need to manage security risk vs. benefit tradeoff
- Risk based security model
- Manual auditing often can help
- Make the cost of attack high – deterrence
  - Either by law
  - Technologically

Why policy matters in security

- Example: Sarah Palin's email account hacked
  - Yahoo accounts have username/password and security questions
  - User can login with username/password
  - If user forgets password – can reset by answering security question
  - Security questions are sometimes easier to guess
  - Some one guessed Palin's highschool, birthday etc
  - Policy amounts to: can log in with either password or security questions

Policy Matters: iCloud Leaks

- August 2014, 500+ private pictures of celebrities were posted on 4chan
- Initially believed to have been brute-force guessing exploiting the fact that iCloud didn't rate limit password checks
- Later turned out to be spear-fishing. Attacker sent emails saying account has been compromised and made it look like they're from Apple/Google

What to do?

- Think hard about implications of policy statements
- Some policy checking tools can help – but you need to specify ‘what is bad’
- Difficult in distributed systems: don’t know what everyone is doing

What might go wrong in threat models/assumptions?

- Human factors not accounted for: ex. Phishing attack
- Computational assumptions change over time:
  - MIT's kerberos system used 56-bit DES keys since mid 1980s
  - Now it costs about $100 to get it cracked
- All SSL certificate CAs are fully trusted
  - To connect to an SSL-enabled website, your browser verifies the cetificate
    - Certificate is a combination of server's host name, and cryptographic key, signed by a trusted CA
  - 100s of CAs are trusted by most browsers
  - In 2011, two CAs were compromised – issued fake certificates for many domains (google, yahoo, tor, …)
  - http://en.wikipedia.org/wiki/Comodo_Group
  - http://en.wikipedia.org/wiki/DigiNotar

Limitations in Assumptions

- Assuming your hardware is trustworthy
  - If NSA is your adversary – it is not necessarily true
    - https://www.schneier.com/blog/archives/2013/12/more_about_the.html
- Assuming good randomness in cryptography
  - Often source of randomness may not be good, and keys may be compromised
  - https://factorable.net/weakkeys12.extended.pdf
- Assuming OS to be secure
  - Bugs? Backdoors? Trojans?
- Machine is disconnected from the Network
  - Did not stop stuxnet worm

What to do to avoid limitations in threat models?

- More explicit and formalized threat models to understand possible weaknesses
- Simpler and more general threat models
- Better design may lessen reliance on certain assumptions
  - E.g., alternative trust models that does not rely on full trust in CAs
  - E.g., authentication mechanisms that aren't susceptible to phishing

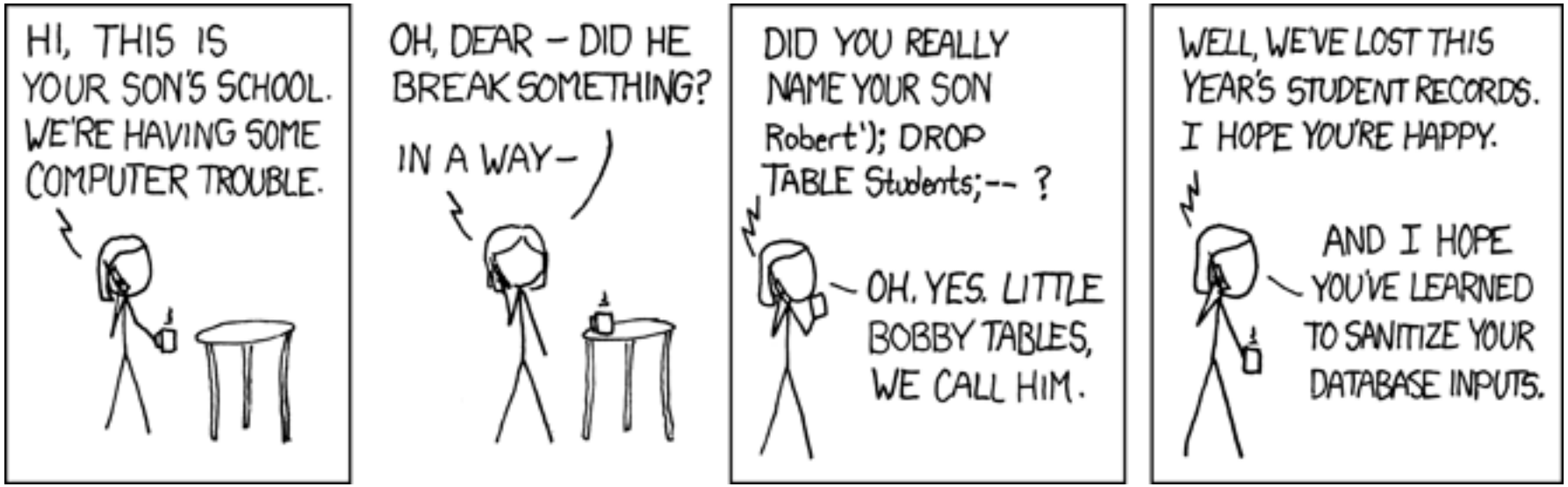
Problems with mechanisms

- Bugs in security mechanism (e.g. OS kernel) lead to vulnerabilities (e.g. CVE-2010-0003)
- Might get pwned by code you didn't know existed (e.g., Intel SMM and SMI)
- If application is enforcing security, application bugs can lead to vulnerabilities
  - Example: Missing access control checks in Citigroup's credit card website
    http://www.nytimes.com/2011/06/14/technology/14security.html?_r=0
  - Example: Android's Java SecureRandom weakness leads to bitcoin theft

Some implementation bugs

- Buffer overflow, Use-after-free, Double-free
- Decrementing stack pointer past the end of stack – into some other memory location
  - http://www.invisiblethingslab.com/resources/misc-2010/xorg-large-memory-attacks.pdf
- Not checking sanity of inputs
  - SQL injection (e.g., see XKCD on next slide)
  - Command injection (e.g., ShellShock)

Bobby Tables



https://xkcd.com/327/