Security and Attack Surface of Modern Applications

Felix 'FX' Lindner

HITBSecConf2007

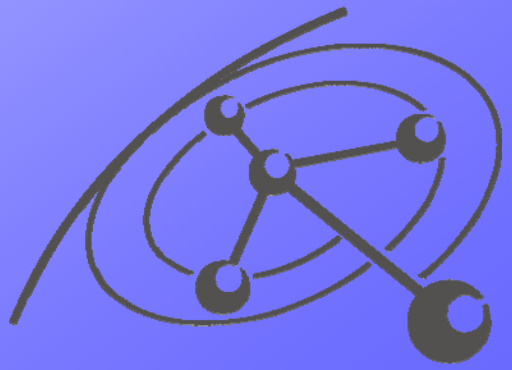
Why am I giving this talk?

- Hackers like hex, 0day and NA6™ talks.
  - Sorry, there won't be any.

We have more important things to worry about.

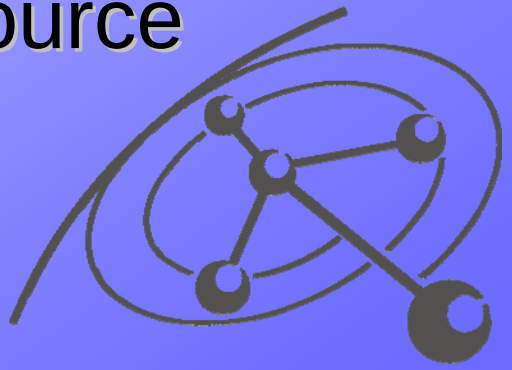
The world is changing!

- If *we* don't fix security *ourselves*, legislation will do it for us, and they will surely f*ck it up **big time**.
- We know things are broken, it's high time that we start fixing them
  - Ever taken money from an ATM?
  - Ever bin to a hospital?
  - Do you like that?



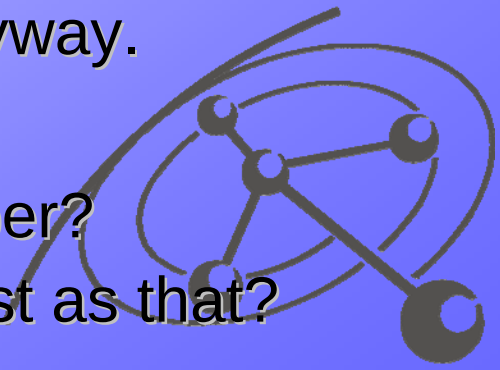
Security is a quality issue

- To reduce the number of vulnerabilities, the number of actual faults must be reduced
- Size of commercial software in average doubles every 18 months
- The defect density is stable since 20 years at 0,5 to 2,0 faults per 1000 lines of source
  - No change with all the new and shiny programming languages



So, let's just fix it!

- Top players in industry hired everyone they can get
  - Microsoft alone needed more (good) professional security review for Vista than the market could provide
  - But even then: much more code than people to read it
- Open Source approach for public review
  - Experienced people cost a lot of money, why should they work for free?
  - They are all busy working in the industry anyway.
- Humans do not scale well
  - Software doubles every 18 months, remember?
  - Can you double security professionals as fast as that?



Recurity Labs

Automated Software Testing

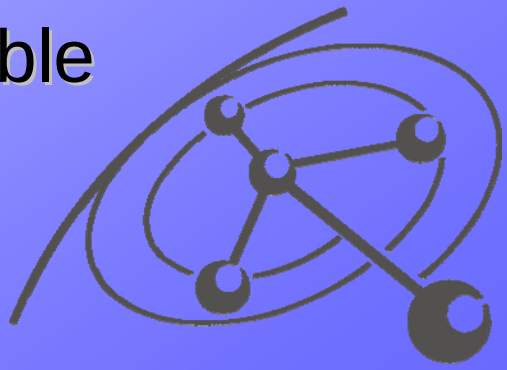
- Software testing seems to be an evacuated science field, measured by the publications.
  - Today, academia focuses on provably correct systems for niche applications… like… aircrafts.
- Today's testing finds max. 30% of the faults but eats up 50%-80% of the development budget
- Security fixes are software modifications. Most have a chance >15% to cause a new fault at least as severe as the fixed issue.

Testing issues

- Software testing research from 1970's
  - We don't even manage to invent *one* new testing method per new programming language
- Methods don't scale
  - Full test of a single addition of 2 variables takes 500 Million Years with 100000 tests per second!
  - More tests don't necessarily find more bugs
- Extreme lack of personal
  - Software testers are rare and expensive
  - (Decent) security specialists are extremely rare and expensive
  - Practitioners from both scenes are no academics and have no access to research funding whatsoever

Testing issues: flying blind

- OOP code is currently un-testable
  - Unit tests are almost never security relevant
  - Automated source code analysis is hard in procedural code – it's impossible in OOP
  - C++ Templates anyone?
- Parallel code execution
  - No testing method known
- Cryptographic mechanisms are un-testable
  - Exercise for the audience:
    Prove the correct implementation of a single cryptographic hash function in C



Silver line at the edge of the screen

- Secure development processes seem to have some visibly positive impact
  - Microsoft's SDL is the prime example
- But…
  - They are expensive
  - They will only be followed while the expected loss due to security issues is higher than the cost of the process

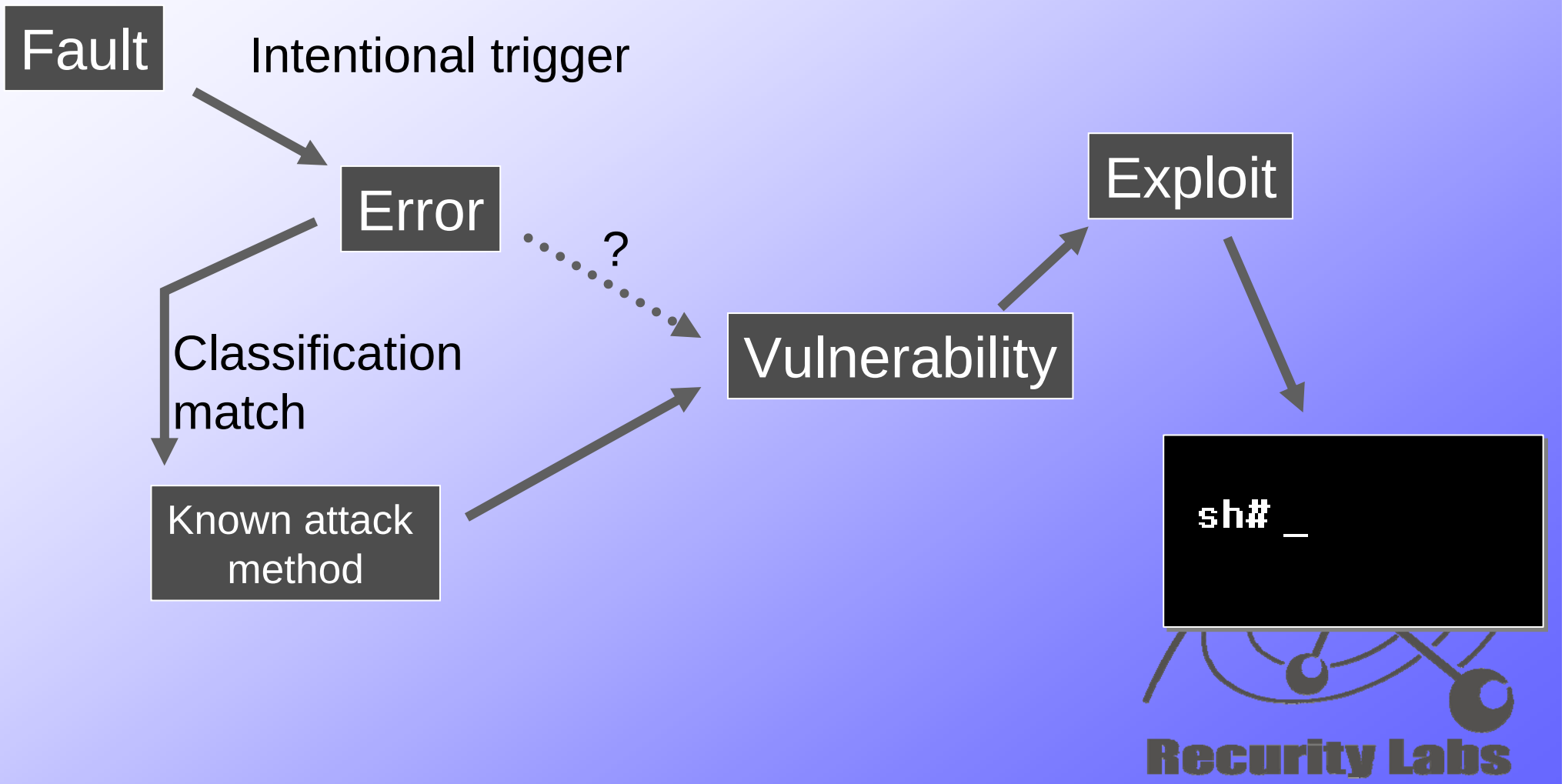
The relation between bugs, vulnerabilities and exploits

…or why hackers should finally start to care about accuracy

Terminology

- Fault:
  The root cause of the fuckup
- Error:
  Instance of the fault that actually happens (Murphy)
- Vulnerability:
  Fault type, for which hackers know how to (mis)use it to gain elevated privileges
- Exploit:
  Instance of a vulnerability application, mostly automatic

The Bug Connection



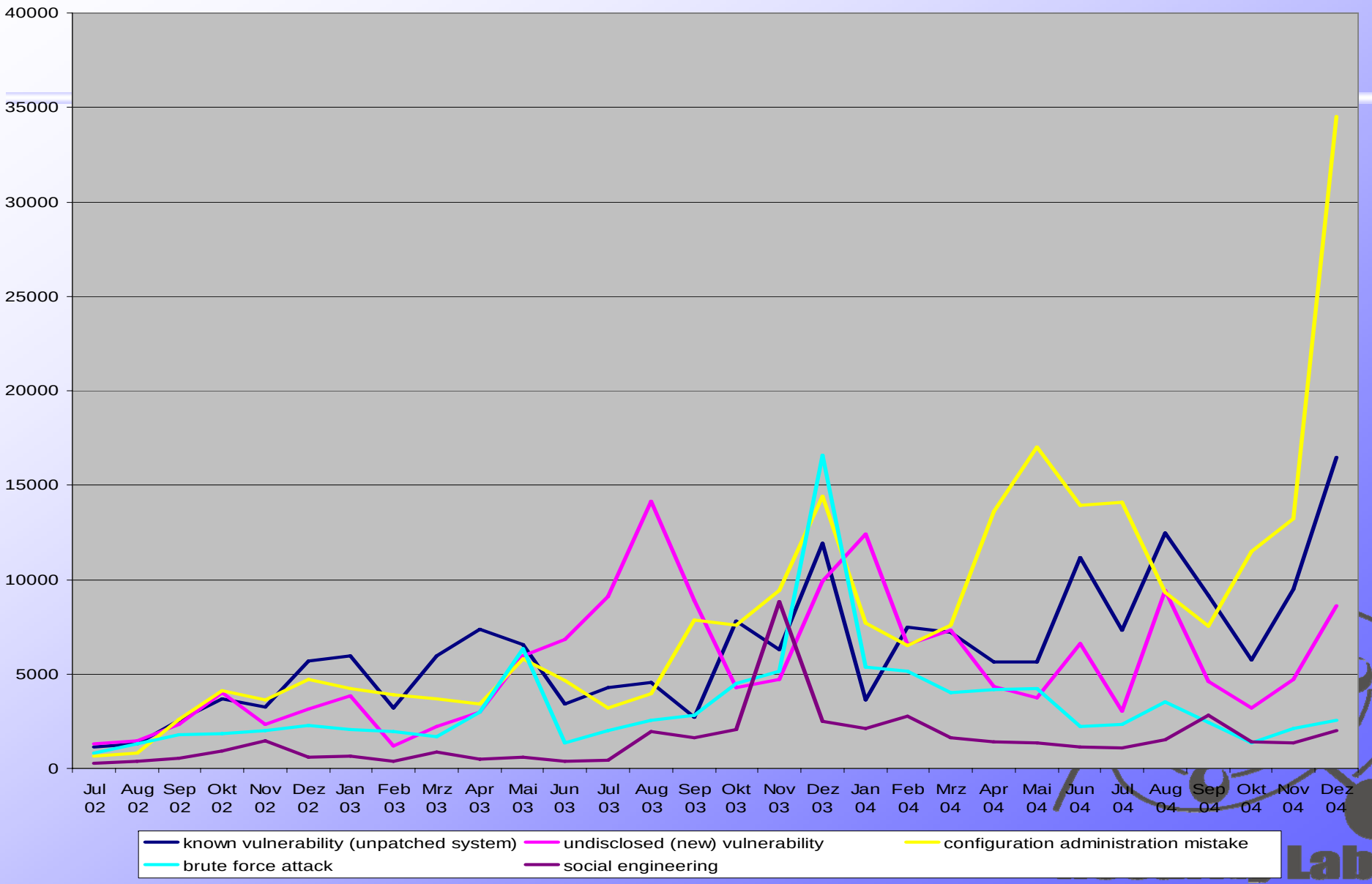
Skill and time

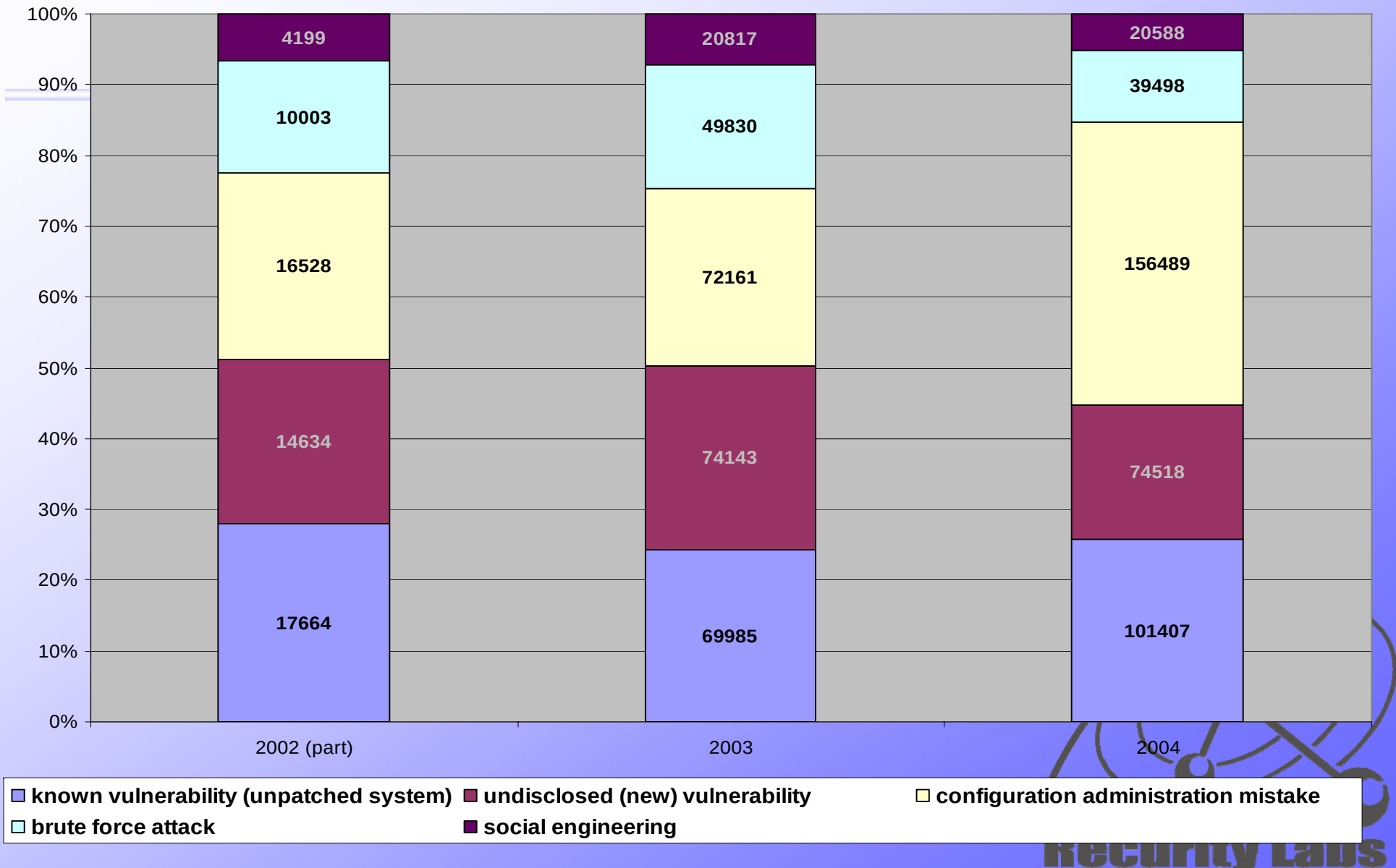
- Fault:
  No skill and time required, fuckups are the responsibility of software designers and developers. (PAL)
- Error:
  Intentionally causing errors requires some skill and time
- Vulnerability:
  Developing a new “bug class” requires creativity and significant skills and time
- Exploit:
  Writing an exploit requires little skills but quite some time

What type of squirrel are you?

p0wnage in the wild

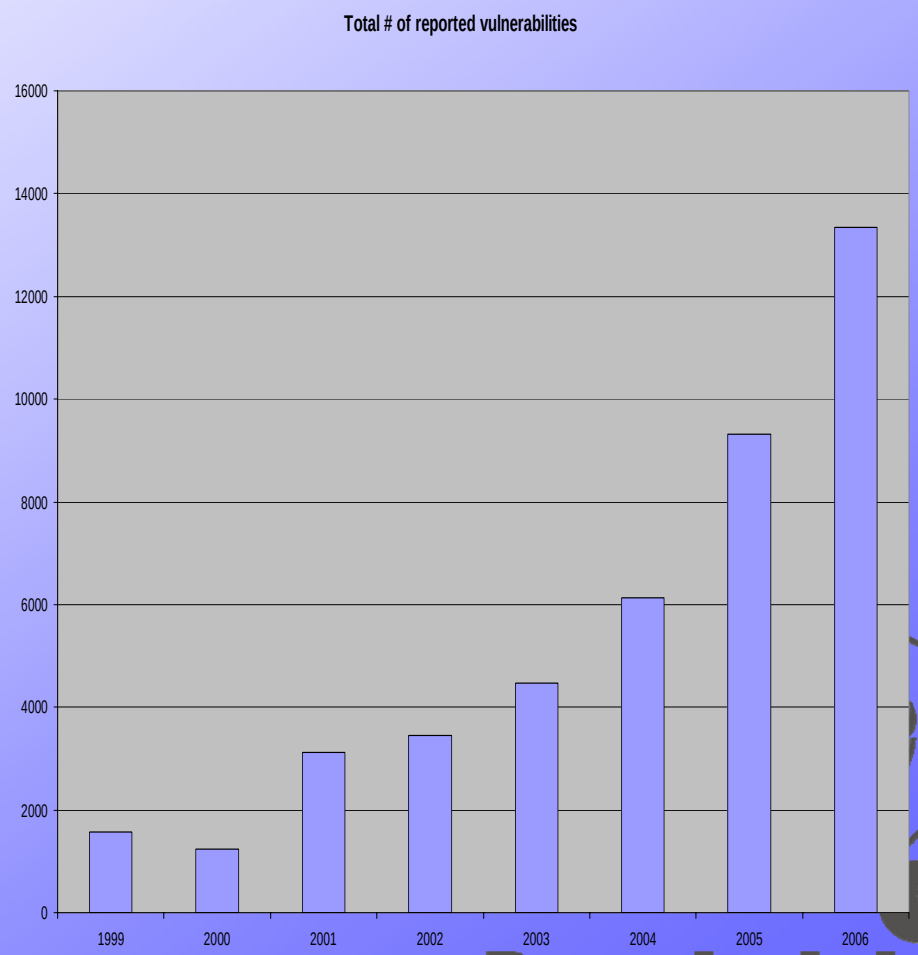
Zone-H Method Chart




Zone-H Method per Year
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
4199
10003
16528
14634
17664
20817
49830
72161
74143
69985
20588
39498
156489
74518
101407
2002 (part)
2003
2004
known vulnerability (unpatched system)
undisclosed (new) vulnerability
configuration administration mistake
brute force attack
social engineering

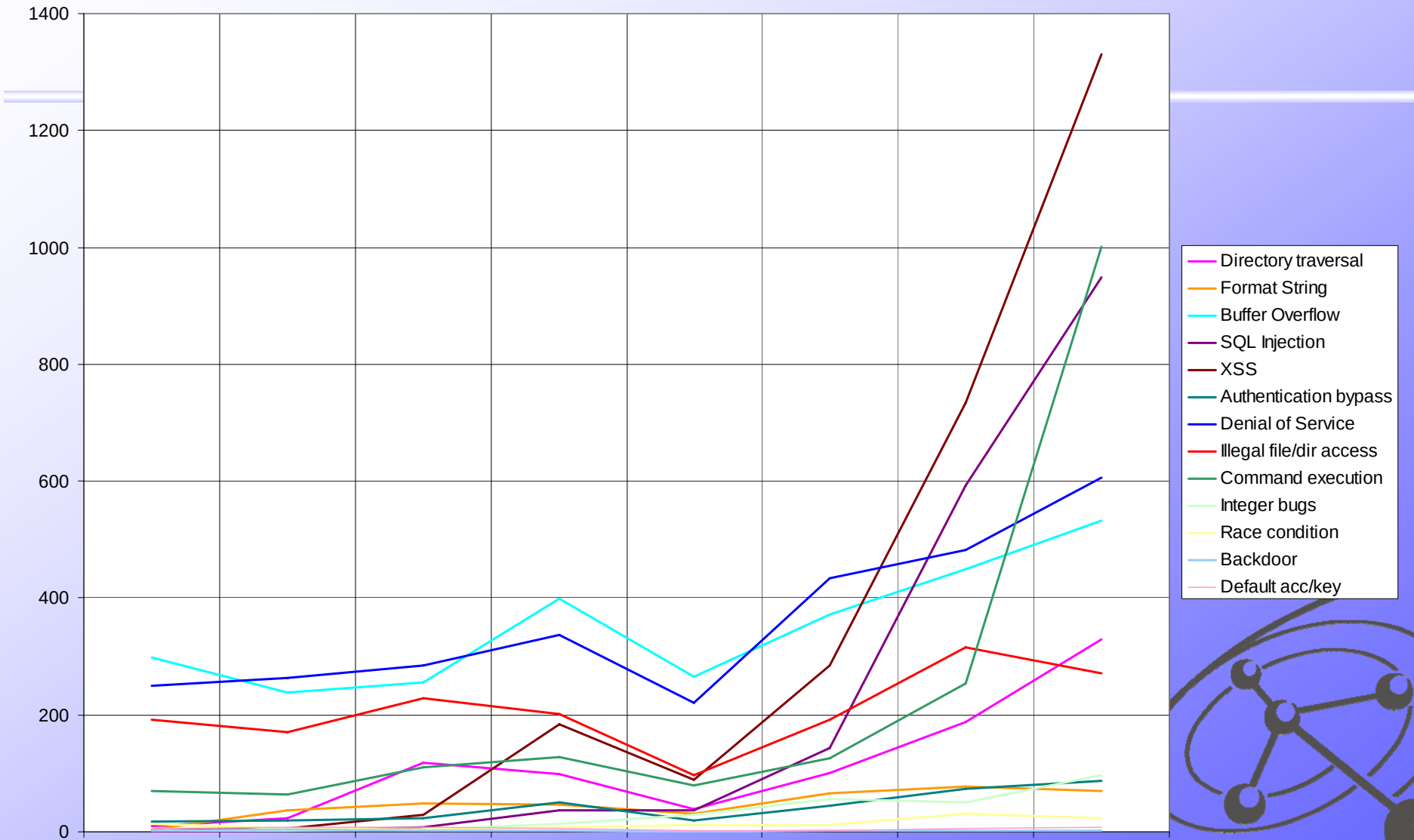
Total # of Vulnerabilities

- Massive increase in the number of known vulnerabilities
- Extremely complex configuration of enterprise solutions
- Further development of attack methods ☺



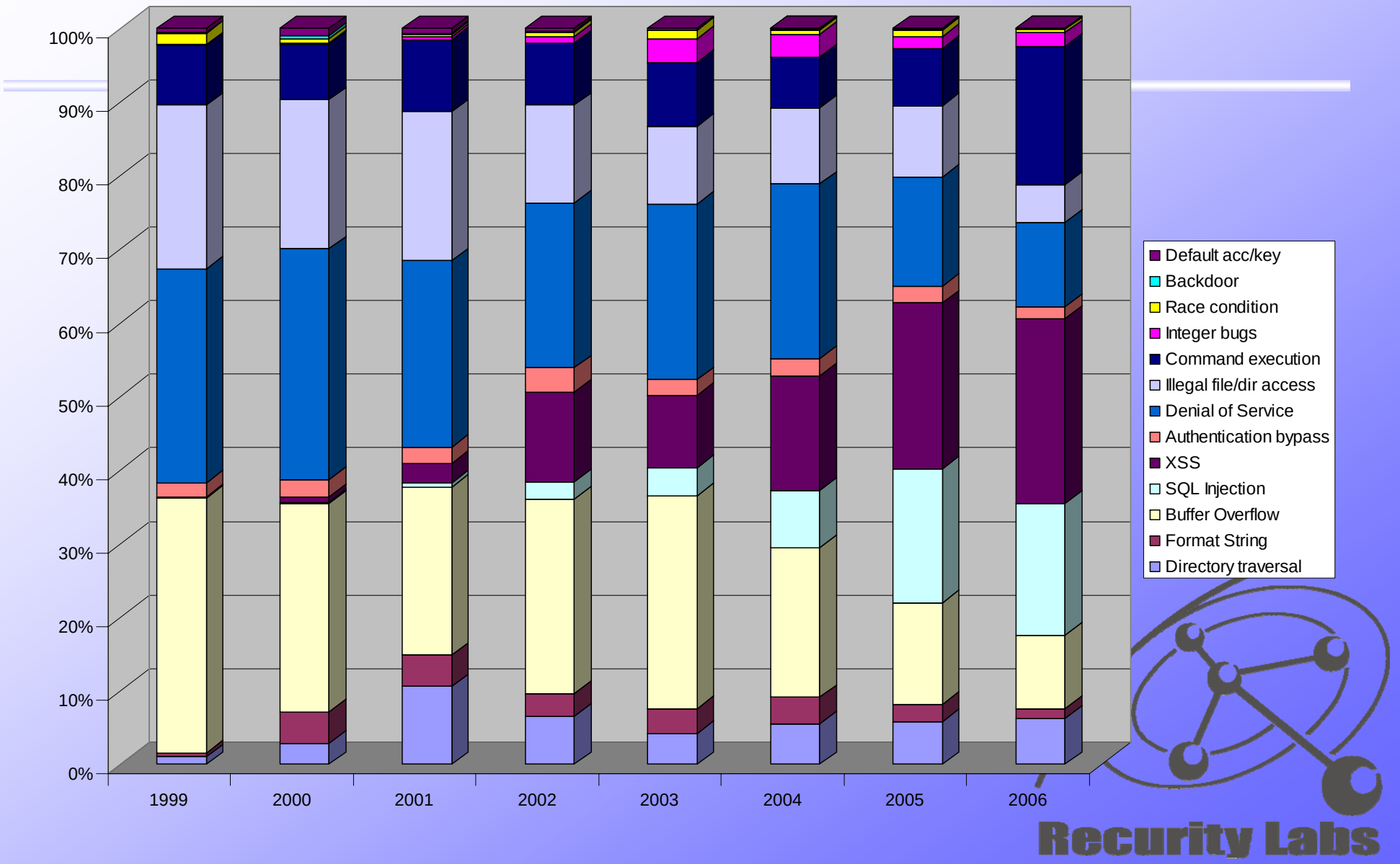
CVE DB

CVE Vulnerabilities



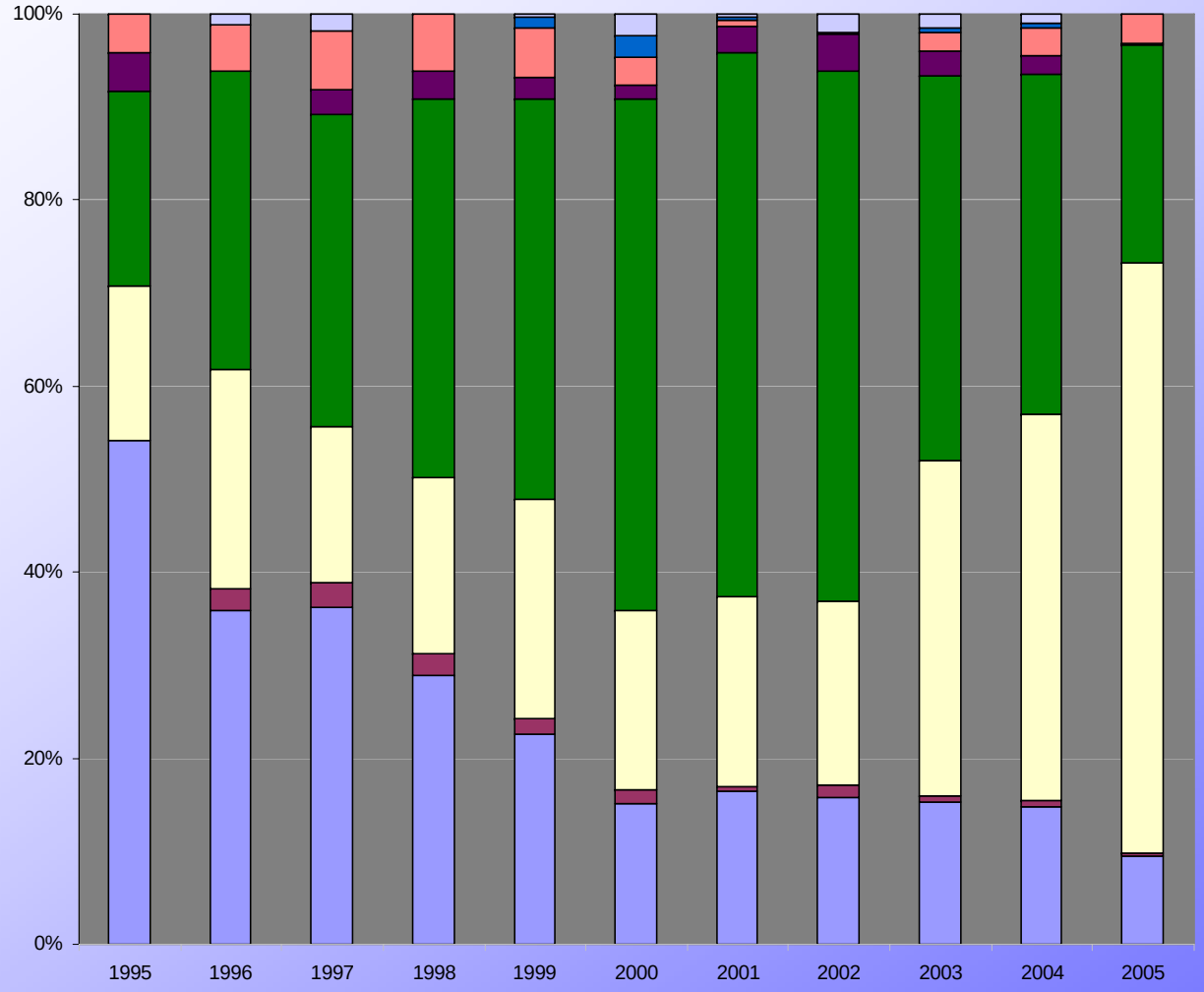
Recurity Labs

CVE Vulnerability distribution per year



Recurity Labs

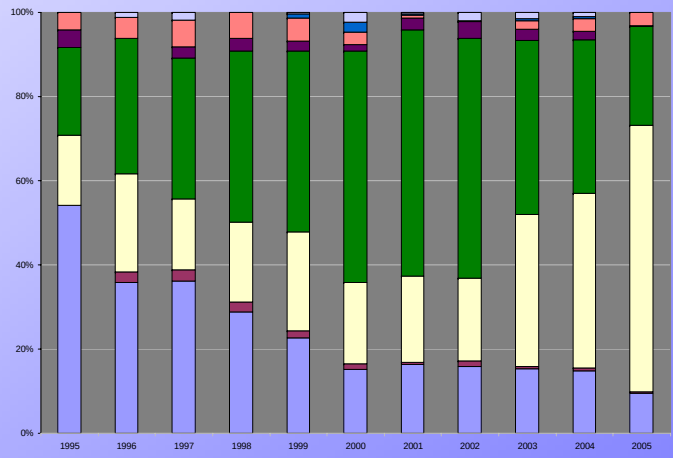
What part is vulnerable?



Data source:
icat.nist.gov
based on CVE

Recurity Labs

What do we see?

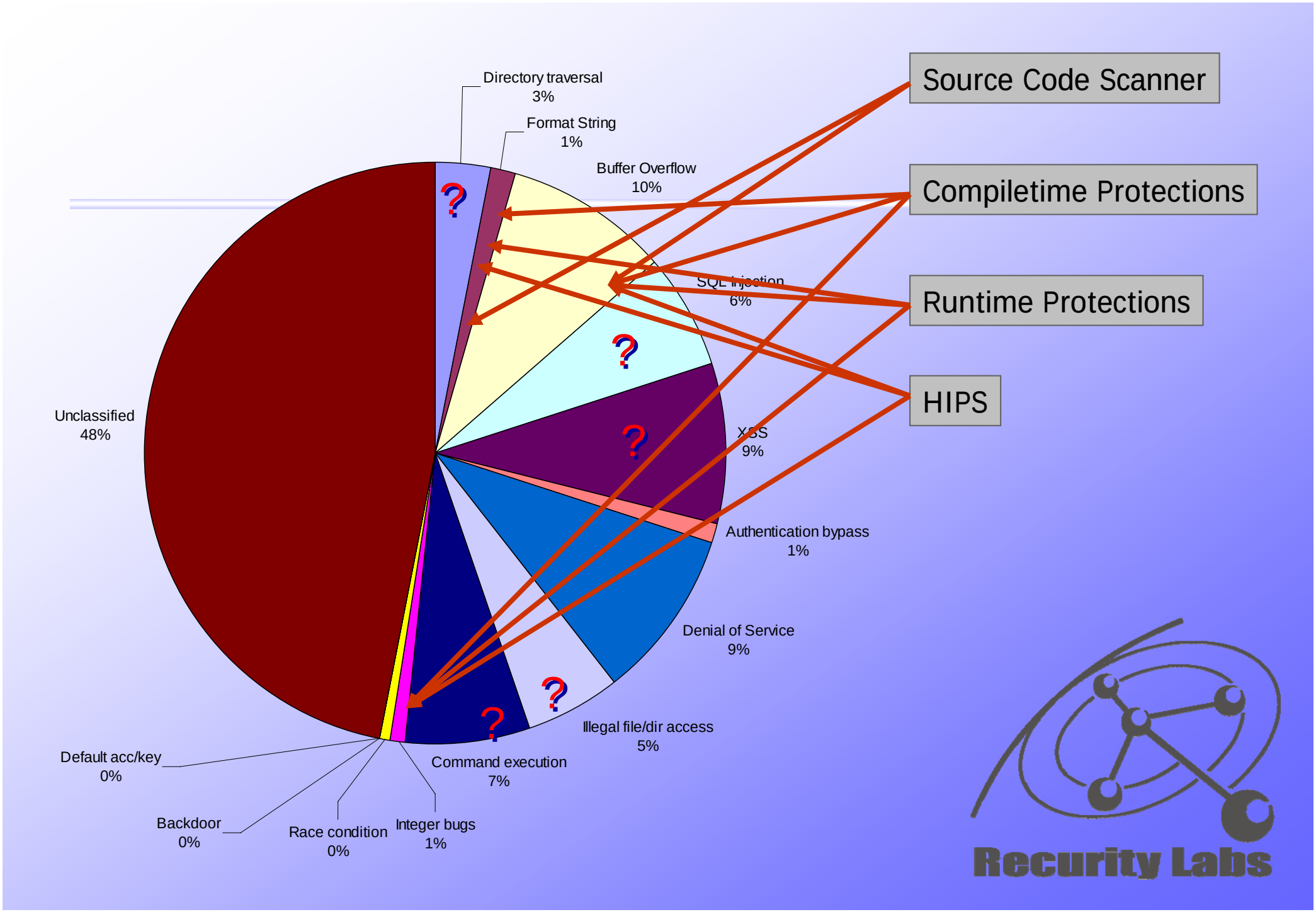


- In the 90s, every OS was easy to hack by itself.
- Around 2000, many servers were vulnerable pre-authentication
- The only solution was to hide everything behind a firewall and hope that the exposed elements would be patched before they were compromised
- Now the attackers focus on the client

Recurity Labs

What changed since 2000?

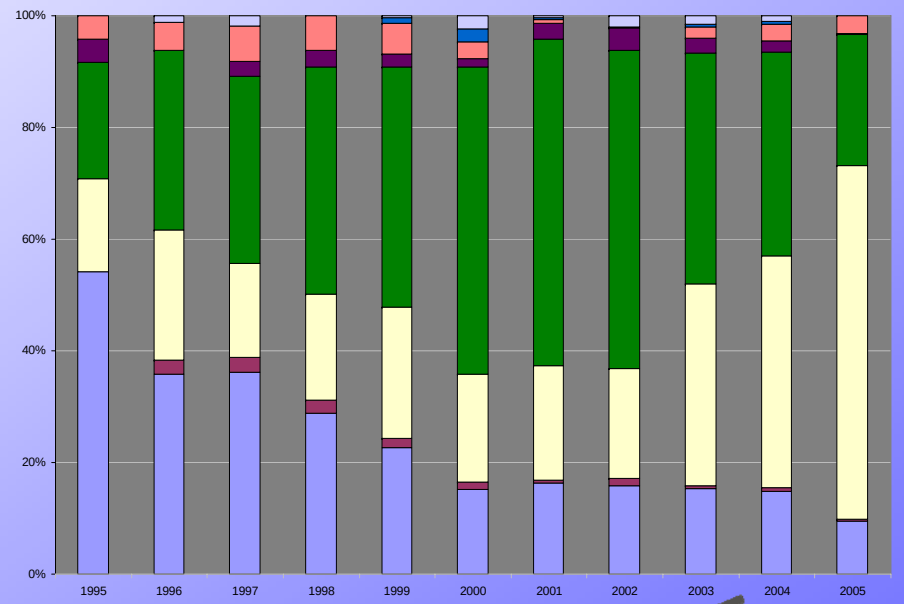
- Automatic source code inspection
  - RATS / ITS
  - Prefast / Prefix
- Security at compile time
  - Warnings when using “no-no functions” such as gets()
  - Introduction of stack canaries in all relevant C compilers
- Security at runtime
  - W ^ X
  - Introduction of heap canaries
  - Address space randomization


Directory traversal
3%
Format String
1%
Buffer Overflow
10%
SQL injection
6%
XSS
9%
Authentication bypass
1%
Denial of Service
9%
Illegal file/dir access
5%
Command execution
7%
Integer bugs
1%
Race condition
0%
Backdoor
0%
Default acc/key
0%
Unclassified
48%
Source Code Scanner
Compiletime Protections
Runtime Protections
HIPS
Recurity Labs

Security Paradigms Reviewed

What happened to: Perimeter Security?

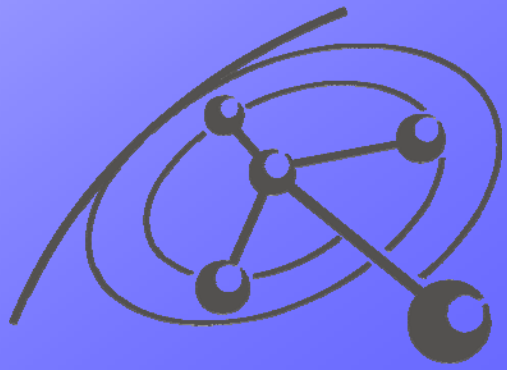
- It was introduced to hide the vulnerable servers.
- Great administrative tool to control what gets exposed
- Never really worked well
  - Now we tunnel everything over HTTP and call it Web Service, just to get past all those firewalls.
- But…



Recurity Labs

What happened to: Perimeter Security?

- Higher bandwidth required faster processing. The result:
  - Firewall code in Kernel space
  - Routers as firewalls
  - Everything written in C for speed
- Dynamic protocols need to be filtered. The result:
  - More complex filter logic than TCP/IP quadruple matching
  - Deep protocol inspection
- What cannot be tunneled through must be terminated at the firewall. The result:
  - VPN termination in firewall products
  - VPN Key material on the firewall
    - or –
    Firewall linked to RADIUS / LDAP / Active Directory



So, today's Firewall is:

- A Multi-Protocol parsing engine
- Written in C
- Running in Kernel space
- Allowed full corporate network access
- Holding cryptographic key material

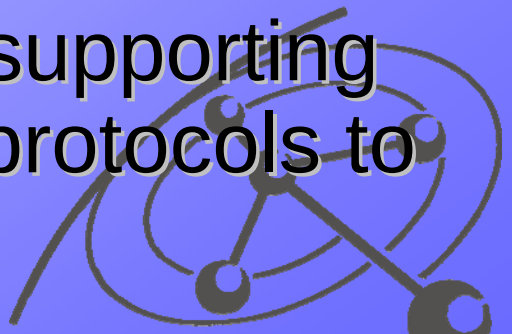
… and still considered a security device?

What happened to: The Detection Paradigm?

- The idea was to detect attacks
- It was marketed to detect intrusions
  - So how do you detect something that the vendor is not able to prevent in the first place?
- Detection paradigm can not work
  - Generation of attacks is always computationally cheaper than detection.
  - Human intelligence is extremely expensive and surprisingly rare.
- Think of IDS logs as Spam mail in you inbox
  - Now imagine you had no spam filter
    - or -
    Imagine you have your current spam filter's reliability

What happened to: The Detection Paradigm?

- Thought experiment:
  We define a secure server as one that does not expose a single vulnerability when parsing and handling the supported protocol.
- Name a single non-trivial, widely used server that never violated the definition above.
- Calculate the probability of an IDS supporting over 140* network and application protocols to comply with the definition.



* The number is taken from ISS Proventia product line, but it's just a thought experiment.
http://documents.iss.net/literature/proventia/ProventiaNetworkIPS_Datasheet.pdf

What happened to: Intrusion Prevention?

- Anti-Hacker technologies
  - Prevent exploitation by preventing known exploitation techniques
  - Faults become bug classes frequently
    - Integer bugs
    - Un-initialized data bugs
    - NULL pointer dereferences
  - Some of the architectural technologies actually made a difference

What happened to: The self-defending Network?

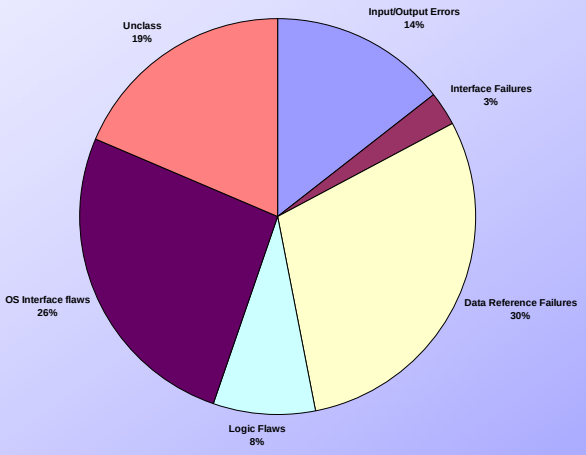
- The Cisco Security Monitoring, Analysis and Response System (CS-MARS) and the Cisco Adaptive Security Device Manager (ASDM) do not validate the Secure Sockets Layer (SSL)/Transport Layer Security (TLS) certificates or Secure Shell (SSH) public keys…
- Processing a specially crafted HTTP GET request may crash the Cisco Secure Access Control Server CSAdmin service. This vulnerability is also susceptible to a stack overflow condition.
- Cisco Security Agent Management Center (CSAMC) contains an administrator authentication bypass vulnerability when configured to use an external Lightweight Directory Access Protocol (LDAP) server for authentication.

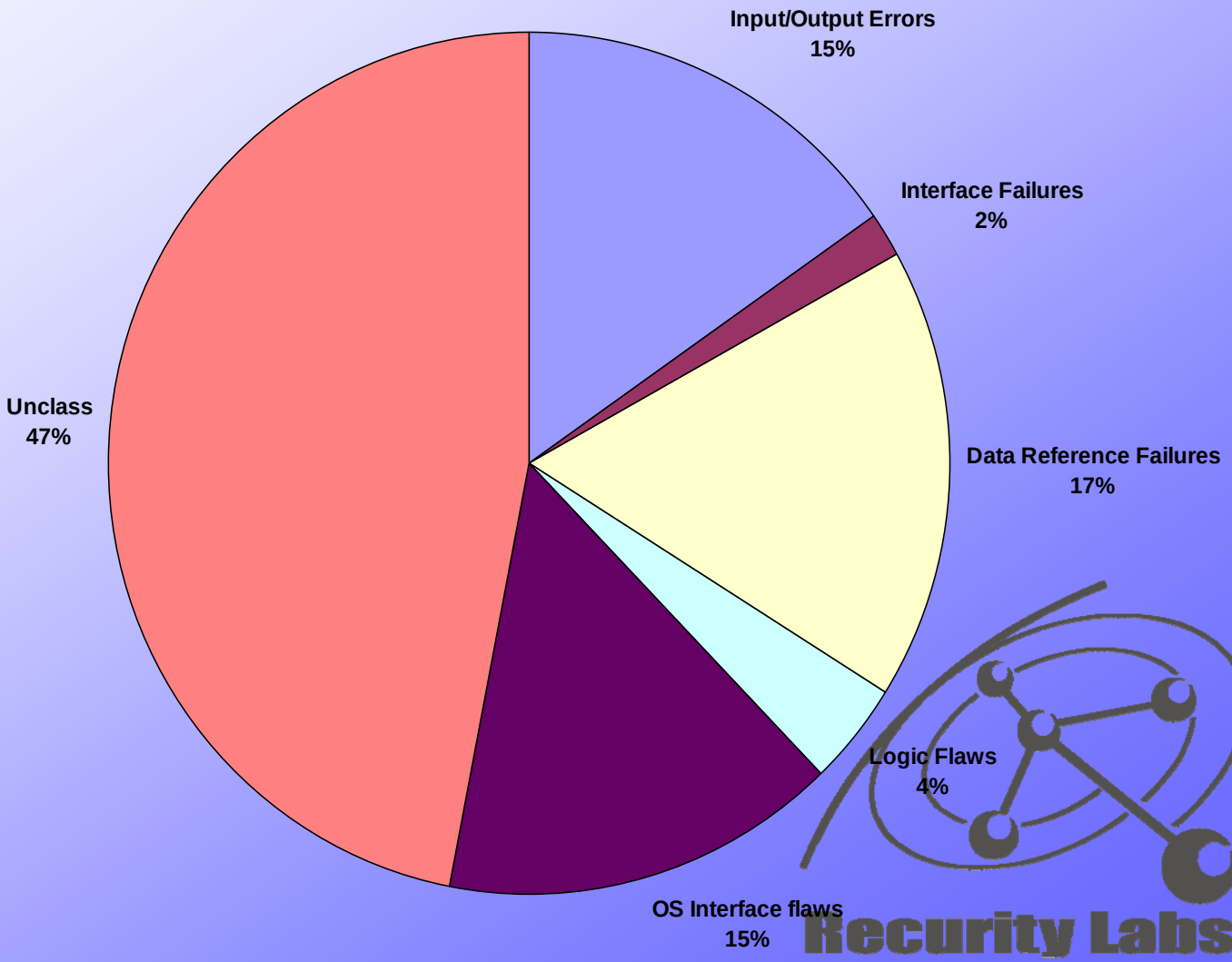
What happened to: The self-defending Network?

- A vulnerability in the Cisco Guard may enable an attacker to send a web browser client to a malicious website with the use of Cross Site Scripting (XSS) when the Guard is providing anti-spoofing services between the web browser client and a webserver.
- Cisco Security Monitoring, Analysis and Response System ships with an Oracle database. The database contains several default Oracle accounts which have well-known passwords.
- Cisco Intrusion Prevention System (IPS) software version 5.1 is vulnerable to a denial of service condition caused by a malformed packet.

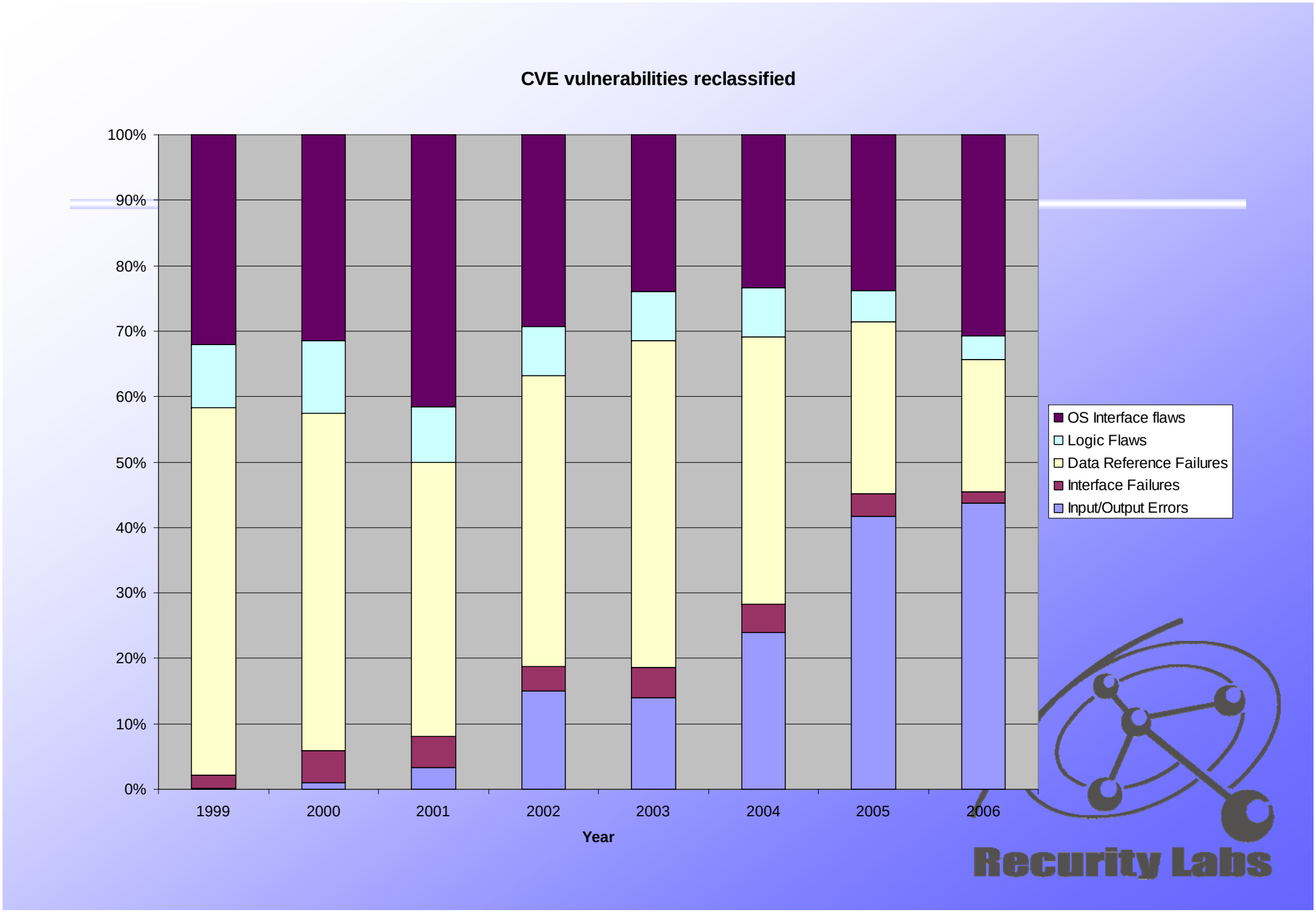
What is all this telling us?

Reclassification of CVE





Recurity Labs


CVE vulnerabilities reclassified
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1999
2000
2001
2002
2003
2004
2005
2006
Year
OS Interface flaws
Logic Flaws
Data Reference Failures
Interface Failures
Input/Output Errors
Recurity Labs

Parser Bugs – the Past

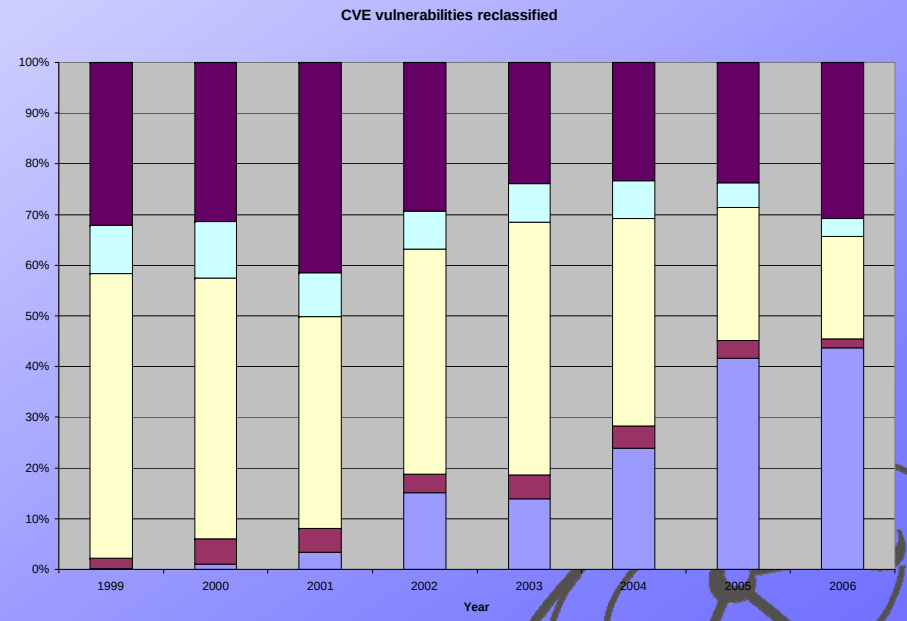
- By far the most common attack vector are vulnerabilities in code interpreting foreign data, also known as parsers
  - Protocol parsers
  - File format parsers
    - Images
    - Office documents
  - Programming language parsers (HTML and JavaScript)
- Almost all of the faults lead to data reference failures
  - Remember the trend on the previous slide?

Parser Bugs XML

- XML is meant to be always possible to parse if the structure is correct
- This assumes that:
  - You don’t write a search & copy parser in C
  - You don’t parse XML by RegEx
  - You validate the XML structure
  - You have a XSD along with your XML

Web 2.0

- I/O Faults are the rising class™
- Logic faults are stable over the years
- Web 2.0 is all about I/O
  - User created (provided) content
  - Web service APIs
- Enormous trust is placed in the browser as the only client a Web 2.0 user needs
  - With 128 toolbars installed
  - All developed in C/C++
  - All running in the address space of the browser



Recurity Labs

Web Frameworks

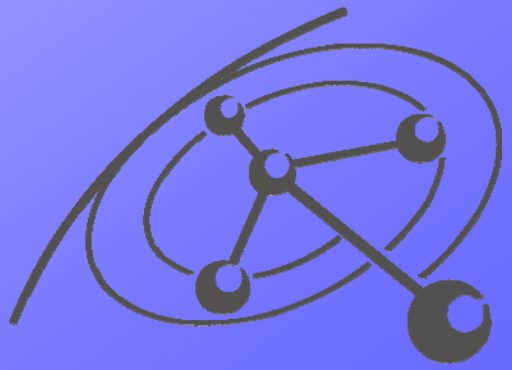
- Complexity kills
  - Web frameworks are way to complex
  - Microsoft Windows is **the** complexity showcase
- Use of large amounts of unknown functionality kills as well
  - Parsing classes
  - Decoding classes
  - Format transformations (Images, Charts)
  - Databases
  - Side effects of all kinds

Framework Example: Ruby on Rails

- Ruby on Rails allows rapid web development
  - MVC architecture
  - Database abstraction
- Default scaffolding prevents standard issues, but doesn't provide much either
  - A single line of view code can introduce XSS
  - Once in the database, the data is mostly trusted by the Rails code
- Active Scaffolding can do more
  - Comes fully SQL-Injectable in may cases

Web Infrastructures

- Mesh-ups just distribute your malicious input faster and more reliably than anything before
  - So far, we have mainly seen *individual* Web 2.0 applications used to distribute Web Malware
- Think: arbitrary malformed input under the Creative Commons License
  1. Input your “XSS”+”SQL”+”FormatString” in any Web 2.0 application you can find.
  2. Mesh them all up, make your data travel
  3. Ask Goggle where your attacks show up
- Back end systems are still old:
  - CORBA, RPC, DCOM, TIBCO



Changes in the way we audit

Java is secure

- No buffer overflows
- No (vulnerable) format strings
- No direct memory access
- Optionally validated code

Java has issues

- Java is predestined for interface and Input/Output faults
  - UNICODE
  - Platform depended file and directory names
  - Mass-failing of filters
- Java suffers from race conditions
  - Multi-Threaded
  - When integrated in multi-process environments, IPC and synchronization issues show

Java has issues standardized

- Integer overflows are standardized in Java
  - The JVM must not generate an exception when a numeric variable overflows or is truncated

int sum = items * price;

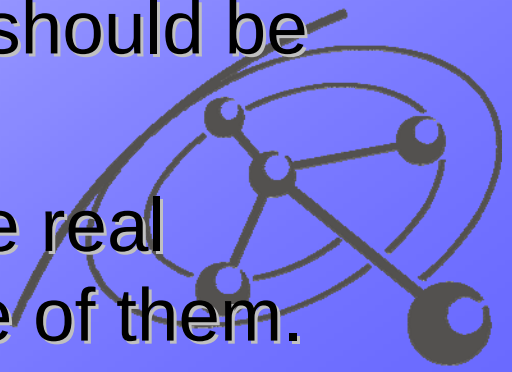
How about:

- price = $1000
- items = 4.294.968

So should we just read more code?

Security is a design issue

- We need to design systems the right way
  - The security tools are just not working
  - Defense in depth is one of our few hopes
  - Get used to the fact that things break
- Write less code, but better code
  - Not everything you *could* do yourself should be done by yourself
  - Respect that software is there to solve real problems for people, security isn't one of them.



Security is a design issue

- Reduce complexity wherever you can
  - You will have less to worry about
- Adding another security *feature* isn't reducing complexity at all
  - If you find yourself doing that, go back to the design board
  - If you find yourself asking the user for a security relevant decision, go back to the design board, square one
  - How about asking someone?

Thank you

…for listening to my rants.

Shouts to:

DanKamInSky,
Phenoelit,
Halvar, Ero,
shadown,
Toralv, Manu,
Gramels, Luiz

And you ☺

Felix ´FX´ Lindner
Head

fx@recurity-labs.com

Recurity Labs GmbH, Berlin, Germany
http://www.recurity-labs.com