

The Exploit Arms Race

The Progression of Attack and Defense in Software

Christien Rioux, Veracode

Exploits are valuable today.

**One day we'll ask ourselves
why they were ever free.**

The exploits that are valuable are the reliable,
productized, ready-to-hack kind.

Attackers only get one shot... if their exploit
doesn't work, they will likely get caught.

'Proof-Of-Concept' means 'Doesn't Work'

Producing free exploits just to participate in a hacker fashion show is an unlikely trend to continue.

Who should be buying exploits?

Corporations? Consultants? Government?

There's an exploit marketplace.

The people buying exploits are:

- **The hacker underground itself**
- **Consultancies**
- **Tools vendors**
- **Countries and intelligence agencies**

The exploits are reliable because the attackers are winning the arms race against the defenders.

So, What's the Endgame?

To understand the exploit arms race is to understand the 'nuclear peace' of hacking.

What happens when there is no more 'low hanging fruit'?

Is this situation possible? Is it likely?

Exploit Techniques Include:

Application-Level Techniques:

- Buffer Overflows
- Integer Overflows
- SQL Injections
- Cross-Site Scripting
- Cross-Site Request Forging

Network Techniques such as:

- Connection Hijacking
- Port Scanning And Sniffing
- Downgrade Attacks

And the list goes on...

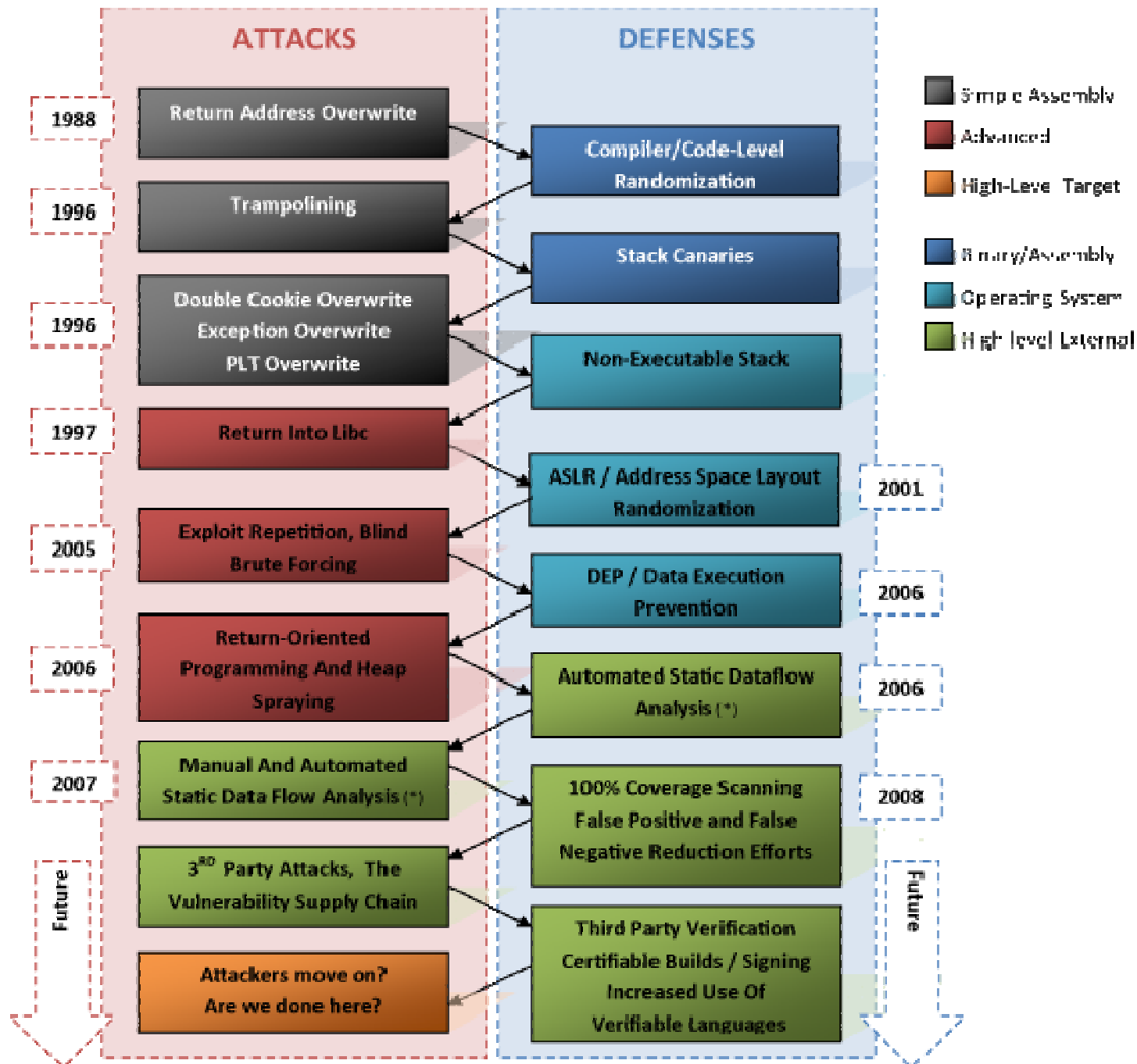
**No better time than now.
Let's get started.**

Lets look at buffer overflows first.

They're been the subject of more than 20 years
of research, from both the attacker and
defender's perspective.

Buffer Overflows: The Internet Worm

- The Internet worm affected about 6000 of the 60,000 connected machines in 1988.
- The vulnerability that allowed this to happen was a buffer overflow.
- 'Solving' the problem of buffer overflows is to this day considered a computationally impossible task.



Automation is the key

The attackers need just *one* good exploit.

Defenders need to fix *as many* vulnerabilities as possible.

To clean up the low-hanging fruit, one needs to automate, since there's so much of it.

What else? Integer Overflows

- Affects trusted file formats often:
 - PDFs
 - Word/PowerPoint documents
 - ZIP/Compressed files
 - Graphics and Movies
- Oh, and that little thing called 42.zip...
- We don't have a good solution for integer overflows.

What else? Cross-Site Scripting

- Some environments like Cold Fusion make XSS nearly unstoppable.
- Others like PHP require a lot of developer skill
- JSP is a good bet
- ASP is good too
- But again, automation or a completely different model is required to ensure that bad code can't create design-level flaws like XSS.

What else? Path Manipulation

- Operating system level chroot? You're on Windows, right? So... ../ ..\\. \\.,... uh
- Unicode throws a wrench into getting this right.
- Think about all those IIS 5.0 vulnerabilities...

So, how do we fix EVERYTHING??

You have a few choices:

- Write better code.
- Pick a better language.
- Automate finding your flaws.
- Get owned.

Do you really need that old code?

What is the cost of maintenance with security risks factored in?

A long time ago, @stake taught people about 'Return on Security Investment'.

People should have listened.

Therein lies the future.

When will the attacked be
willing to sacrifice their old code?

Their corporate baggage?

Mission critical control systems written in C?

One day, will we look at JAVA and C#
as security dinosaurs?