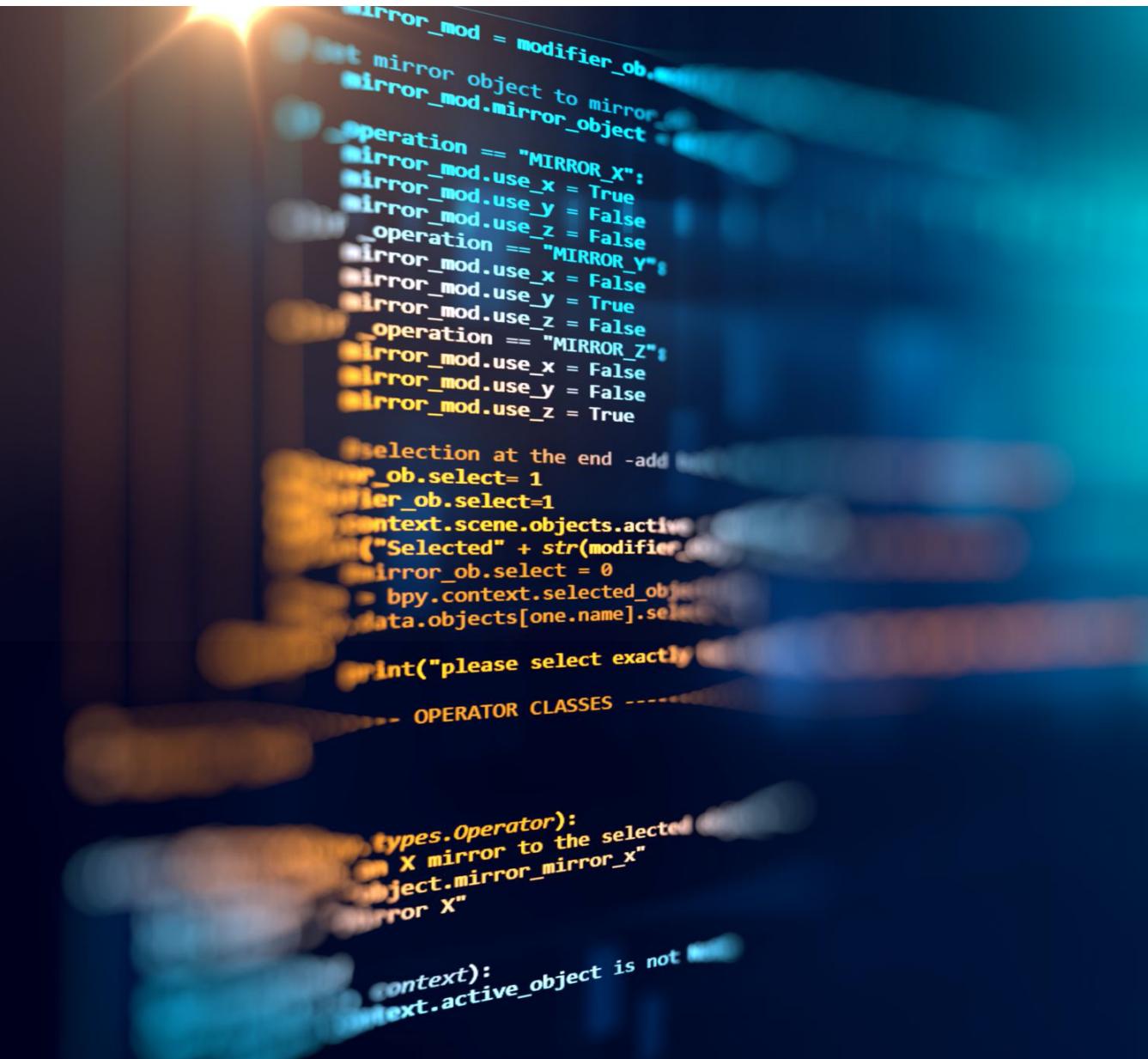


LANGUAGE BASED TECHNOLOGY FOR SECURITY

CHIARA BODEI
GIAN LUIGI FERRARI



The image shows a hand pointing towards a computer screen. The screen displays a block of Python code. The code appears to be a script for a 3D modeling application, specifically Blender, based on the context of the operators mentioned. The code includes logic for selecting objects, setting mirror modifiers, and handling operator classes.

```
mirror_mod = modifier_obj
# mirror object to mirror
mirror_mod.mirror_object = ob
operation = "MIRROR_X":
    mirror_mod.use_x = True
    mirror_mod.use_y = False
    mirror_mod.use_z = False
operation == "MIRROR_Y":
    mirror_mod.use_x = False
    mirror_mod.use_y = True
    mirror_mod.use_z = False
operation == "MIRROR_Z":
    mirror_mod.use_x = False
    mirror_mod.use_y = False
    mirror_mod.use_z = True

# selection at the end -add
ob.select= 1
ler_ob.select=1
context.scene.objects.active = eval("Selected" + str(modifier))
mirror_ob.select = 0
bpy.context.selected_objects = []
data.objects[one.name].select = 1
int("please select exactly one object")
----- OPERATOR CLASSES -----
types.Operator:
    # X mirror to the selected object.mirror_mirror_x"
    "mirror X"
    context):
        context.active_object is not None
```

PREREQUISITES

- If you've ever programmed in a variety of programming languages (C, Java, C++, OCaml, JavaScript etc.) then that will be a helpful skill.
- Also, if you know how programming languages are implemented, that will be quite useful too.

OUTLINE

- Course logistics
- Course objectives & Grading
- About us
- What is “Language-based Security”?

Course Information

- The teaching materials (lectures, readings and exercises) will be available on TEAMS
 - 714AA 20/21 - LANGUAGE-BASED TECHNOLOGY FOR SECURITY [WCY-LM]
- Prof.ssa Chiara Bodei – chiara.bodei@unipi.it
- Prof. Gian Luigi Ferrari – gian-luigi.ferrari@unipi.it
- Office hours: After classes

Course Objectives

- Issues
 - How does security typically fail in software?
 - Why does software often fail?
 - and what are the underlying root causes?
- What are ways to make software more secure?
 - principles, methods, tools & technologies
 - Secure programming languages (design and implementation)
 - practical experience with some of these



INTERMEZZO

TOP MAINSTREAM LANGUAGES (TIOBE INDEX)

Feb 2021	Feb 2020	Programming Language	Ratings	Change
1	2	C	16.34%	-0.43%
2	1	Java	11.29%	-6.07%
3	3	Python	10.86%	+1.52%
4	4	C++	6.88%	+0.71%
5	5	C#	4.44%	-1.48%
6	6	Visual Basic	4.33%	-1.53%
7	7	JavaScript	2.27%	+0.21%
8	8	PHP	1.75%	-0.27%
9	9	SQL	1.72%	+0.20%
10	12	Assembly language	1.65%	+0.54%

**Same data
2013-2017-
(<http://nvd.nist.gov>)**

#5899 vulnerabilitis due to buffer errors

#5851 vulnerabilities due to injection errors

#3106 vulnerabillities due to information leak

#53% of software vulnerabilities in NVD were buffer errors, injections and information leaks

All of these are issues
within the realm of
Programming
Language
Design

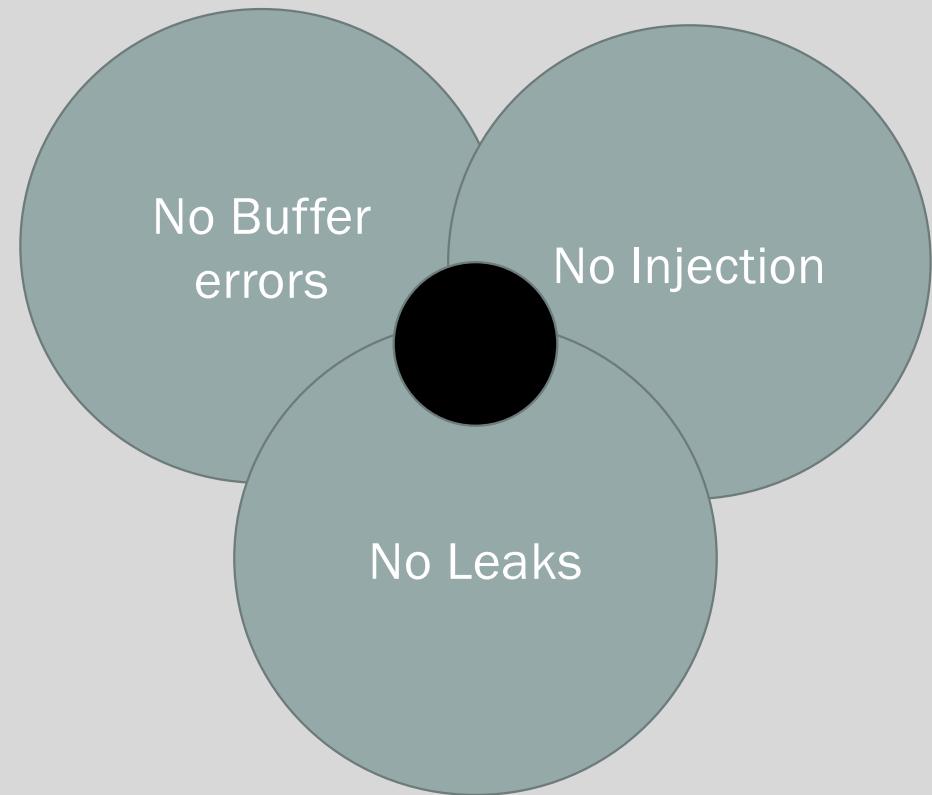
#5899 vulnerabilities due to buffer errors

#5851 vulnerabilities due to injection
errors

#3106 vulnerabilities due to information
leak

#53% of software vulnerabilities in NVD
were buffer errors, injections and
information leaks

**A Secure Programming Language
is one that provides
abstraction to avoid
the three main
categories of
vulnerabilities**



Avoid Buffer Errors Dynamically



John McCarthy 1958

- Managed memory
 - Garbage Collection introduced in LISP in 1958
- Now in
 - OO Languages
 - Java, C#, JavaScript, ...
 - Functional Languages
 - ML, Haskell, OCAML, F#, ...
 - Dynamic Languages
 - Ruby, Perl, ...

Avoid SQL Injection Statically



MICROSOFT 2007

- LINQ .NET's Language Integrated Query Framework
- LINQ to SQL manages relational data as objects without losing the ability to query
 - Statically typed

Permette di essere utilizzato all'interno del run time di .NET in modo interoperabile per fare delle query alle basi di dati relazionali pur continuando a rimanere in un modello di programmazione ad oggetti.

Avoid Information Leaks Statically



Andrew Myers, 2002

- Extends Java with information flow and access control enforced at compilation time and run time
 - Integrity
 - Confidentiality
 - Prevent information leaks

Controlla sia a livello statico usando i meccanismi tipici di analisi statica di Java che a tempo di esecuzione che non ci siano flussi di esecuzione scorretti.

TODAY MAIN STREAM PROGRAMMING LANGUAGES ARE NOT SECURE LANGUAGES

```
mirror_mod = modifier_obj.modifiers.new("mirror", "MIRROR")
# mirror object to mirror
# mirror_mod.mirror_object = ob
# mirror_mod.mirror_axis = 'Y'
# mirror_mod.use_x = True
# mirror_mod.use_y = False
# mirror_mod.use_z = False
# operation == "MIRROR_X":
# mirror_mod.use_x = True
# mirror_mod.use_y = False
# mirror_mod.use_z = False
# operation == "MIRROR_Y":
# mirror_mod.use_x = False
# mirror_mod.use_y = True
# mirror_mod.use_z = False
# operation == "MIRROR_Z":
# mirror_mod.use_x = False
# mirror_mod.use_y = False
# mirror_mod.use_z = True

# Selection at the end - add
# ob.select= 1
# mirror_ob.select=1
# context.scene.objects.active = context.scene.objects.active
# ("Selected" + str(modifier))
# mirror_ob.select = 0
# bpy.context.selected_objects[0].select = 1
# data.objects[one.name].select = 1
# print("please select exactly one object")

# --- OPERATOR CLASSES ---

# types.Operator:
#     X mirror to the selected
#     object.mirror_mirror_x
#     mirror_x

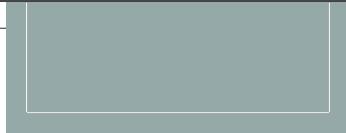
# context:
#     context.active_object is not None
```

SECURE PROGRAMMING
LANGUAGES
PROVIDE
ABSTRACTIONS
AND TOOLKITS TO
SUPPORT
SECURE PROGRAMMING

Vorremmo che i linguaggi di programmazione venissero progettati fin dall'inizio sicuri e che la sicurezza non sia solo legata all'aspetto di path quando il software è prodotto, sia dal punto di vista delle tecniche di ingegneria del software sia dal punto di vista degli strumenti che vengono usati per programmare e dunque vorremmo che la sicurezza non fosse sacrificata sull'altare dell'efficienza.

Secure Programming Language Design: issues to consider

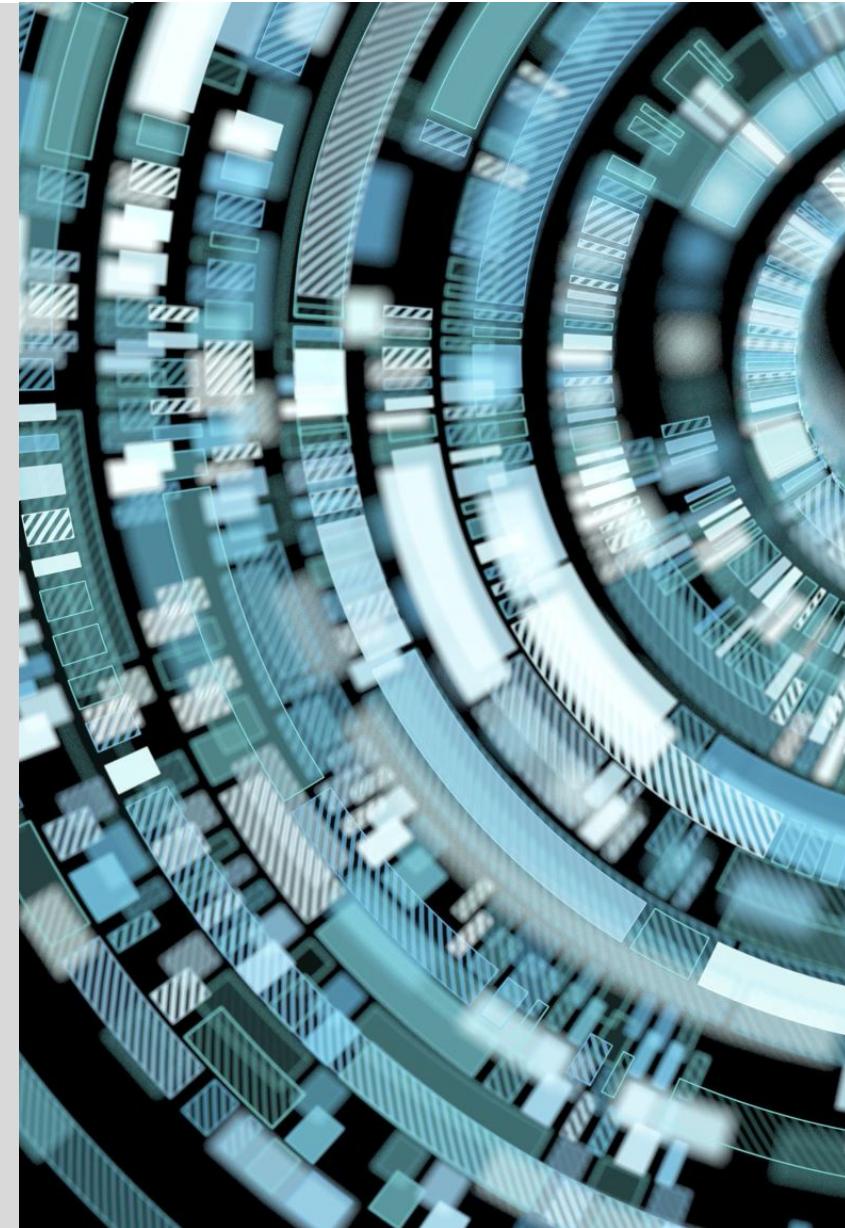
- Programming abstraction for security
- Interoperability
- Virtual machines & Run time supports
- Compilation that preserves security abstraction
- Static techniques (types, control and data flow analysis)
- Run-time type checking
- :



**END
INTERMEZZO**



IT'S TIME TO
INTRODUCE SECURITY
ABSTRACTIONS IN
LANGUAGE DESIGN
AND
IMPLEMENTATION

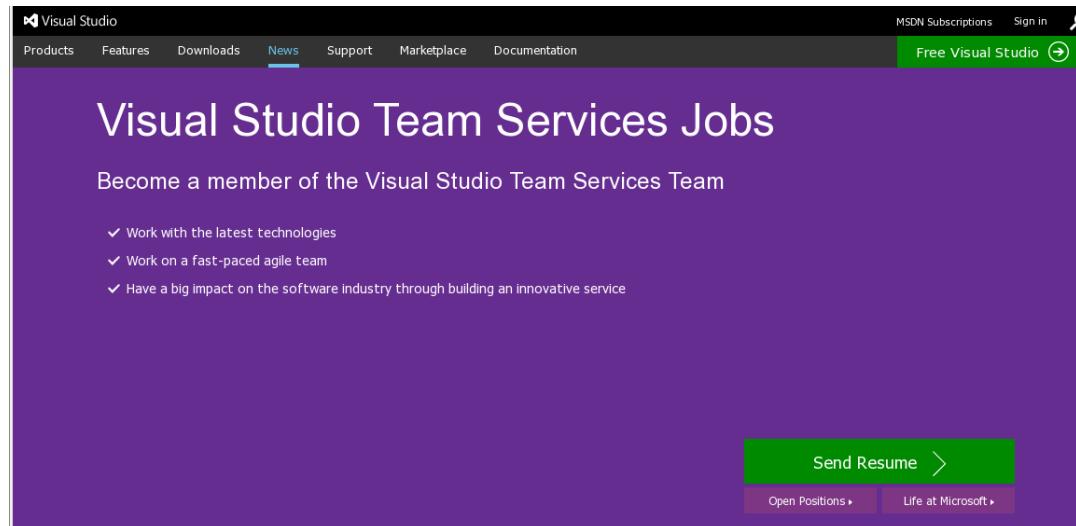


Who cares?

- In most cases, there is a clear mapping between top-level ICT professionals and jobs:
- Take Databases -> work at Oracle
- Take E-Commerce and services -> work at Amazon
- Take Networks -> work at Cisco
- Take Graphics -> work at Nvidia
- *Which companies develop compilers or interpreters?*

Microsoft

› Visual Studio, Excel, etc.



The screenshot shows a purple-themed web page for 'Visual Studio Team Services Jobs'. At the top, there's a navigation bar with links for Products, Features, Downloads, News (which is underlined), Support, Marketplace, Documentation, MSDN Subscriptions, Sign in, and a 'Free Visual Studio' button. Below the header, the main title 'Visual Studio Team Services Jobs' is displayed in white. A sub-headline 'Become a member of the Visual Studio Team Services Team' follows. A bulleted list of benefits includes: '✓ Work with the latest technologies', '✓ Work on a fast-paced agile team', and '✓ Have a big impact on the software industry through building an innovative service'. At the bottom right of the purple section is a green 'Send Resume >' button. Below this, two smaller buttons are visible: 'Open Positions' and 'Life at Microsoft'. A call-to-action 'Join us!' is located at the very bottom left of the page.

Visual Studio Team Services Jobs

Become a member of the Visual Studio Team Services Team

- ✓ Work with the latest technologies
- ✓ Work on a fast-paced agile team
- ✓ Have a big impact on the software industry through building an innovative service

Send Resume >

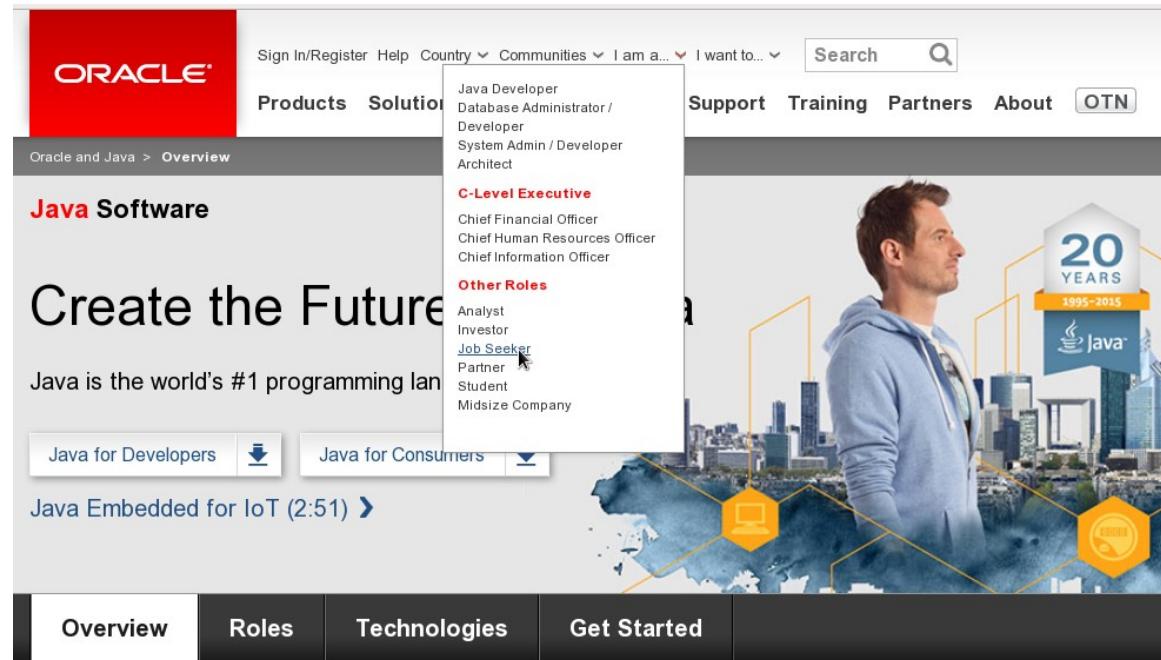
Open Positions | Life at Microsoft

Join us!

Become a key member of the Visual Studio Team Services (VSTS) service team led by [Brian Harry](#) and build the next generation of development services in the cloud! VSTS provides software development teams with version control, build automation, agile work management, social experiences and more to nearly 3,000,000 users.

Oracle

› Java Compiler, Java Virtual Machine



Intel

The screenshot shows the Intel Developer Zone website. The top navigation bar includes links for 'Join Today >', 'Log in', 'powered by Google', and a search icon. The main heading is 'Intel® C++ Compilers'. Below the header, there's a section titled 'Leadership Application Performance' with a bulleted list: 'Boost C++ application performance', 'Future-proof code by making code that scales', and 'Plugs right into your development environment'. A paragraph explains that the Intel C++ Compiler is available in four products based on application needs. Two product cards are shown: 'PARALLEL STUDIO XE' and 'SYSTEM STUDIO'. To the right are two circular icons representing 'Intel® C++ Compilers in Intel® INDE (support only)' and 'Intel® Bi-Endian Compiler'.

Leadership Application Performance

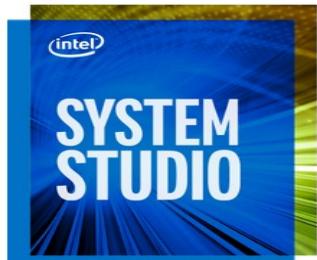
- Boost C++ application performance
- Future-proof code by making code that scales
- Plugs right into your development environment

If you are here, you are looking for ways to make your application run faster. Boost performance by augmenting your development process with the Intel® C++ Compiler. The Intel C++ Compiler plugs right into popular development environments like Visual Studio®, Eclipse®, XCode®, and Android Studio®. The Intel C++ Compiler is compatible with popular compilers including Visual C++® (Windows®) and GCC (Linux®, OS X® and Android®).

The Intel C++ Compiler is available in four products based on your application development needs:



Intel® C++ Compiler in Intel® Parallel Studio XE



Intel® C++ Compiler in Intel® System Studio



Intel® C++ Compilers in Intel® INDE (support only)



Intel® Bi-Endian Compiler

Google

Go, Dart, etc.

The screenshot shows the Dart programming language website. At the top, there's a navigation bar with links for 'GET STARTED', 'FUNDAMENTALS', 'WEB', 'SERVER', 'MORE', and a search bar. A dropdown menu under 'MORE' includes options like 'Code Samples', 'Synonyms with Other Languages', 'Dart by Example', 'Articles', 'Language Specification', 'Who Uses Dart', 'FAQ', and 'Logos and Colors'. Below the navigation, there's a large blue banner with the text 'Scalable, productive' and a subtext about Dart being an open-source, scalable programming language. To the right of the banner, there's a teal-colored section with the word 'opment'. In the center, there's a 'The Go Programming Language' section with a 'TRY GO' button. On the left, there's a 'Try Go' code editor window containing Go code for printing 'Hello, 世界'. At the bottom, there are buttons for 'Run', 'Share', and 'Tour', and a dropdown menu set to 'Hello, World!'. To the right of the Go section, there's a large image of the Go mascot, a gopher, and a 'Download Go' button with text about binary distributions for Linux, Mac OS X, Windows, and more.

Apple

- Objective-C. LLVM.

LLVM è un framework per progettare compilatori di macchine astratte, dunque è uno strumento con all'interno una quantità di meccanismi per analizzare il codice prodotto



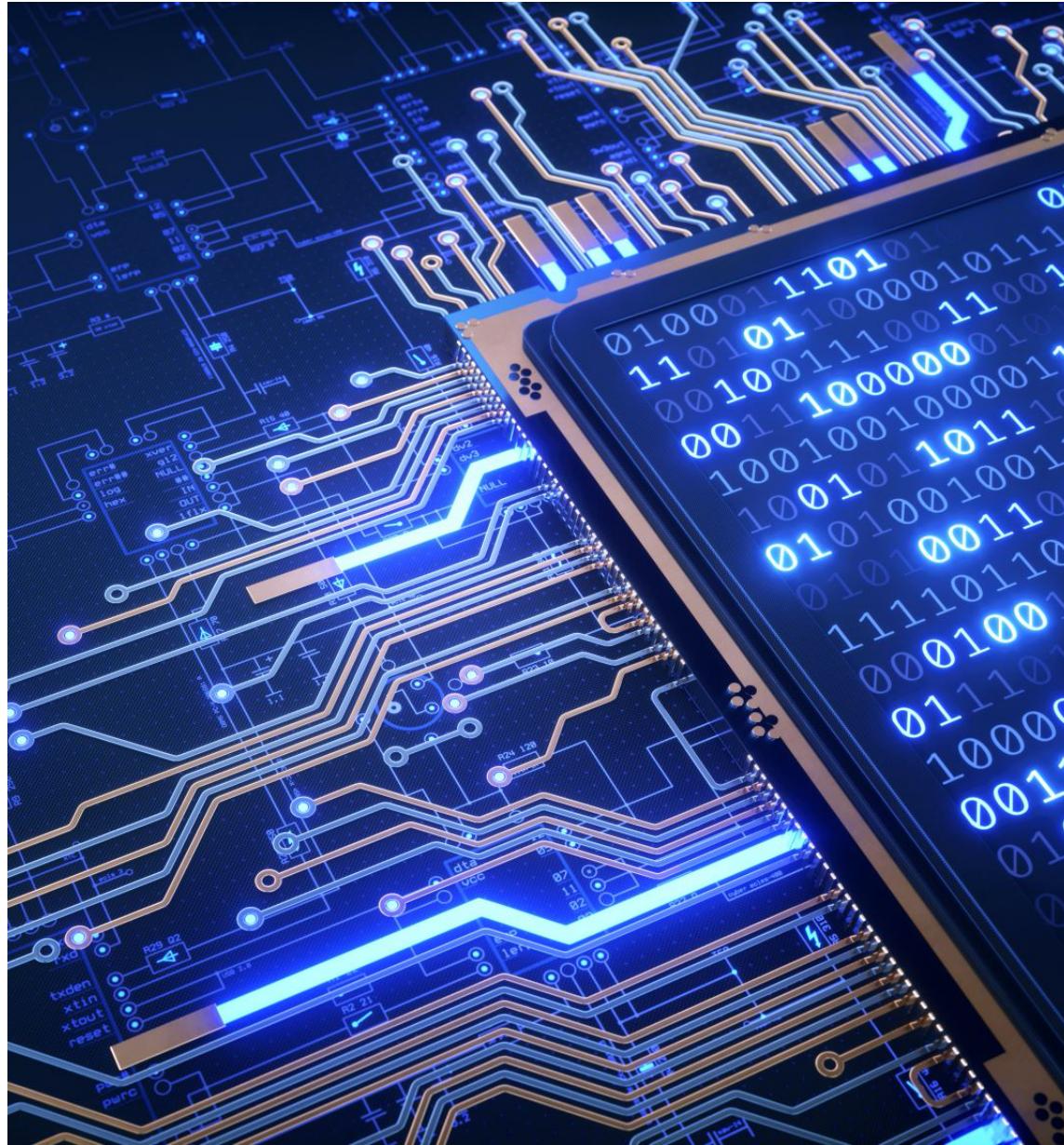
LLVM Compiler Overview

The LLVM compiler is the next-generation compiler, introduced in Xcode 3.2 for Snow Leopard, based on the open source [LLVM.org](#) project. The LLVM.org project employs a unique approach of building compiler technologies as a set of libraries. Capable of working together or independently, these libraries enable rapid innovation and the ability to attack problems never before solved by compilers. Multiple technology groups within Apple are active contributors within the LLVM.org community, and they use LLVM technology to make Apple platforms faster and more secure.

In Xcode, the LLVM compiler uses the Clang front end (a C-based languages project on LLVM.org) to parse source code and turn it into an interim form. Then the LLVM code generation layer (back end) turns that interim format into final machine code. Xcode also includes the LLVM GCC compiler, which uses the GCC compiler front end for maximum compatibility, and the LLVM back end, which takes advantage of LLVM's advanced code generator. This shows the flexibility of a library-based approach to compiler development. There are many other features, such as link-time optimization, more detailed diagnostic information, and even static analysis, that are made available to Xcode due to the adoption of LLVM.

About Objective-C

Objective-C is the primary programming language you use when writing software for OS X and iOS. It's a superset of the C programming language and provides object-oriented capabilities and a dynamic runtime. Objective-C inherits the syntax, primitive types, and flow control statements of C and adds syntax for defining classes and methods. It also adds language-level support for object graph management and object literals while providing dynamic typing and binding, deferring many responsibilities until runtime.



In a nutshell

- Skills on the design and implementation of secure programming languages is one notch more than a standard ICT professional skills

How programming languages are implemented?

Two major strategies (roughly speaking!!!!)

- interpreters (take source code and run it)
- Compilers (translate source code, run result)

Interpreters run programs “as is”

- Little or no static analysis and static optimization

Compilers do extensive static analysis

Most implementations use mixed technique (interpreter+compiler)

Why study the design and implementation of secure programming languages

- Prepare for many good jobs
- Improve understanding the security issues of program behavior:
 - Know how things work “under the hood”
- Increase ability to learn new languages
- Learn to build a large and secure system
- See many basic concepts of computer science jointly at work
- Computers are the only tools that increase cognitive power, so learn to control them

le vulnerabilità verranno viste con la lente di un progettista di nuovi linguaggi di programmazione oppure progetta i compilatori per i nuovi linguaggi, dunque dal punta di vista del linguaggio --> primitive del linguaggio + strumenti di implementazione

Then ... what will we do in this class

- Learn about different kinds of **software vulnerabilities** and **how** they can be handled by designing and implementing programming languages
- This will permit to gain **exposure to new ideas** in programming language design and implementation
- ... **Lab** to get your hands dirty
 - You will design and implement your own programming language
 - **Do not be afraid: a simple interpreter!!**

Course Objectives (cont.)

- Cutting-edge research
 - Learn how to extract (the important) info from security-related research articles
 - Learn about modern efforts toward a science of computer security
- Many problems/questions are open-ended. We will be exploring the known issues together.

Grading

- Class participation (20%)
 - discuss topics, articles & ask questions
- Project work (40%)
 - Project proposal and implementation during last 4 weeks of course
- Written exam (40%)
 - main concepts
 - Feasibility, main limitations, pros/cons
 - A few harder in-depth questions to test whether you caught subtle but essential details
-

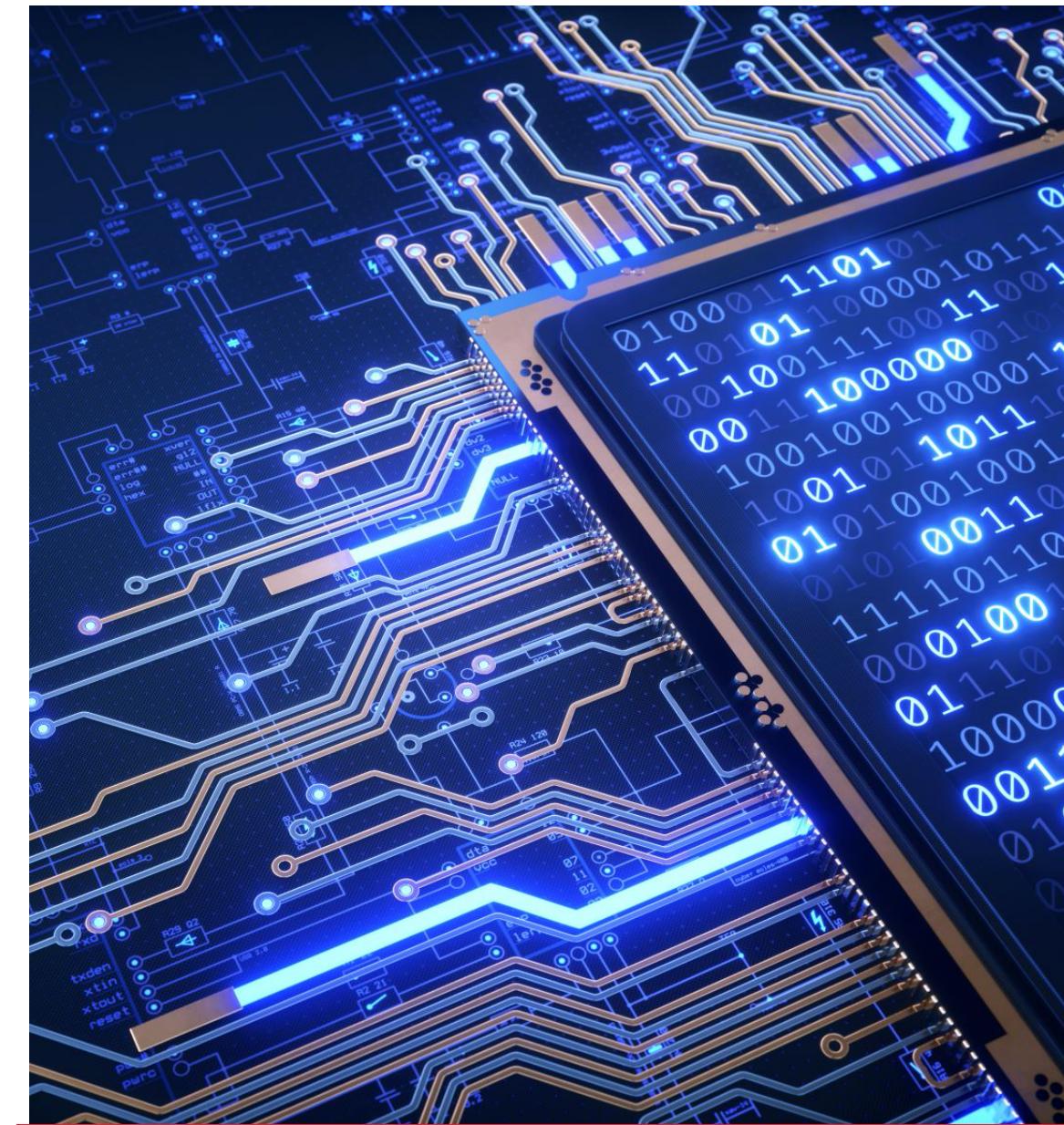
What is “Language Based Security”

- Leveraging theory of programming language design and compiler construction to enforce software security
- Two issues
 - new languages/tools for creating secure software
 - securing legacy code (e.g., written in C)
- Three stages of enforcement
 - static (find & fix vulnerabilities before runtime)
 - dynamic (detect and block attacks at runtime)
 - audit (recover and assign blame after an attack)

Challenges

- Is it possible to develop secure software that is guaranteed to be vulnerability-free?
- Scenario: You are hired to write the control software for self driving cars
 - it must NEVER fail (several lifes at stake)
 - it must cope with adversarial conditions (operate in a dynamic environment, have to work in all weather conditions anywhere on the planet)
 - it must be efficient (too slow = lifes at stake)
- Traditional approaches
 - test a lot (“It didn’t crash today...”)
 - write a proof (consisting of about 10K pages of math)





Challenges

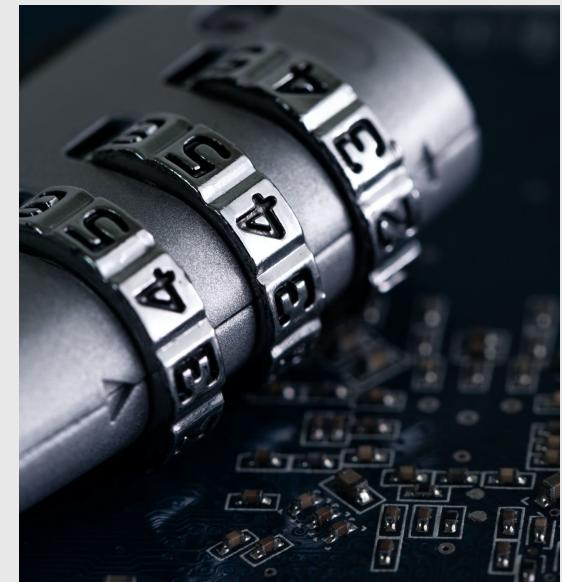
- Dipartimento delle informazioni per la sicurezza (DIS) wants secure software on their office workstations.
 - need web browsers, document readers, etc.
 - need internet connectivity
 - stores and/or reads top secret documents
- not feasible to rebuild the entire universe of software from the ground up
- software is proprietary (and usually closed-source)
- How to stop secrets from leaking?

Avere dei linguaggi per il coordinamento del software che permettono di comporre assieme software sviluppato sicuro con software legacy garantendo le proprietà di sicurezza ovviamente potrebbe aiutare.

Scienza della sicurezza significa avere dei meccanismi con cui sono in grado di stabilire scientificamente cosa vuol dire avere una politica di sicurezza, quali sono le sue proprietà e come posso avere dei meccanismi di enforcement

A Science of security

- Can we develop a **science of security** like we have for math or physics?
 - Are there **certified** “proofs” of security?
 - What does it even mean for a system to be “secure”?
- Can we determine that one software system is “more secure” than other?
- Are there some security policies that are provably unenforceable?
- Can we prove that certain enforcement mechanisms can enforce certain classes of policies and not others?



A dark, semi-transparent background image showing a large, dense pile of numerous small wooden blocks, each shaped like a human figure. The blocks are painted in various colors including light wood, dark brown, black, white, orange, red, teal, and blue. They are scattered across the frame, creating a sense of a diverse community or group.

MOTIVATIONS

Why Software Security

- Software plays a major role in providing security and is the major source of security problems.
- Software is the weakest link in the security chain, with the possible exception of “the human factor”
- Software security does not get much attention
 - in programming courses,
- or indeed, in much of the security literature!

Many discussions of security begin with Alice and Bob



**How can Alice communicate securely with Bob,
when Eve can modify or eavesdrop on the communication?**



THIS IS AN INTERESTING
ISSUE BUT **NOT** THE
BIGGEST PROBLEMS

Se usiamo un canale sicuro, magari i dati arrivano correttamente da una parte all'altra però quello che ci dovremmo domandare è: abbiamo dei meccanismi per essere sicuri che il computer di Alice non è attaccato e quindi non c'è qualcun altro che ne sta prendendo il controllo? Risolvere il problema della comunicazione è sicuramente importante ma non affronta il problema di avere degli attacchi dovuti a degli input maliziosi che possono viaggiare su canali di comunicazione sicura.

Alice's computer is communicating with *another computer*



How can we prevent Alice's computer from being hacked,
when it communicates with some other computer?

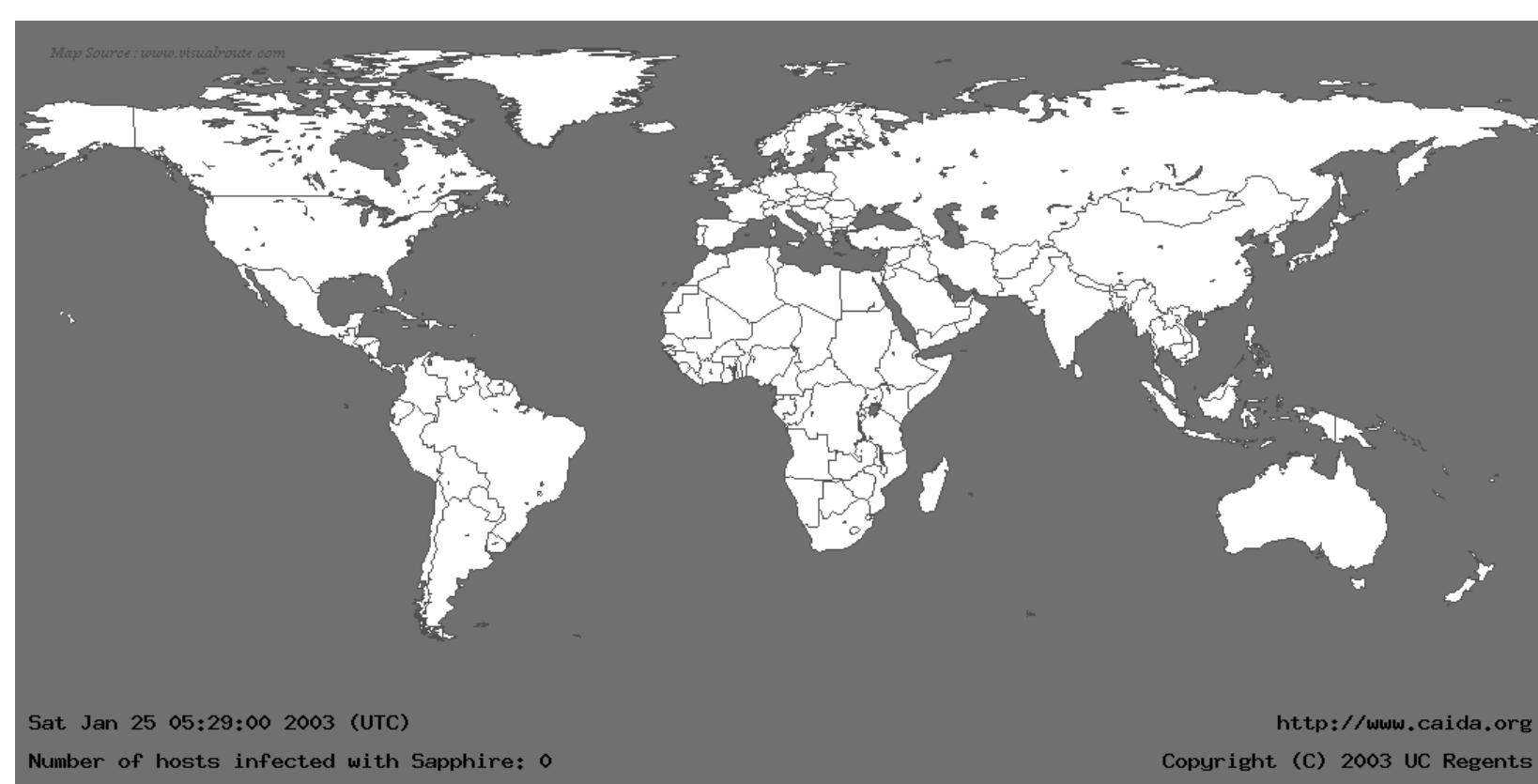
Or detect this? And then react ?

Solving the 1st problem - securing the communication - does not help!

The Slammer Worm (2003)

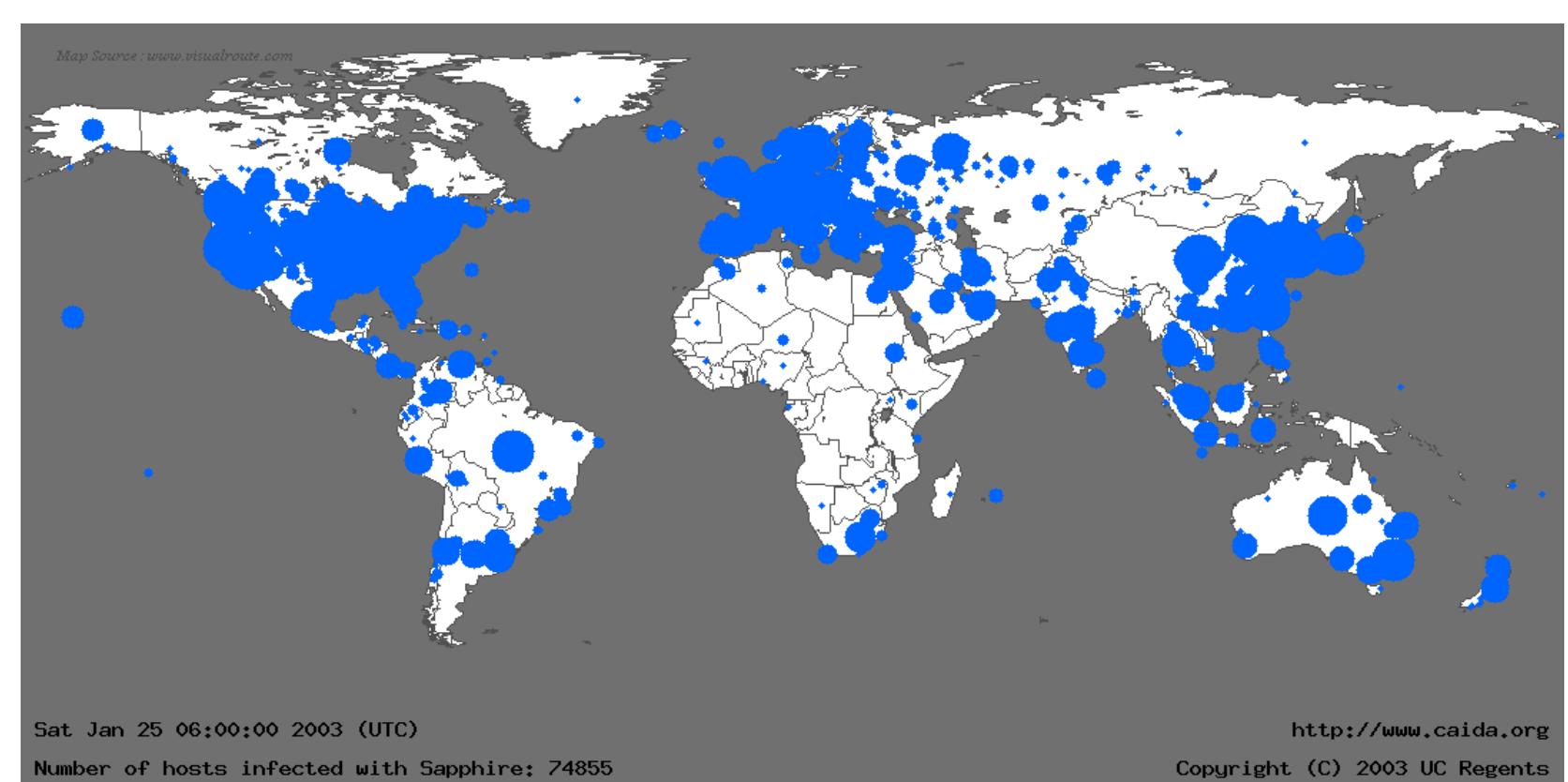
- Slammer is a 2003 computer worm that caused a denial of service on some Internet hosts and dramatically slowed general Internet traffic.
- It spreads rapidly, infecting most of its 75,000 victims within ten minutes.

FIGURA PRIMA CHE IL WORM EFFETTUASSE IL SUO ATTACCO



From *The Spread of the Sapphire/Slammer Worm*, by David Moore et al.

DOPO AVER EFFETTUATO L'ATTACCO



From *The Spread of the Sapphire/Slammer Worm*, by David Moore et al.



Security Problems

- To get an impression of the problem, have a look at
 - US-CERT bulletins
 - <http://www.us-cert.gov/ncas>
- CVE (Common Vulnerability Enumeration)
 - <https://cve.mitre.org/cve/>
- NIST's vulnerability database
 - <https://nvd.nist.gov/vuln/search>

Attackers

- Traditionally, hackers are amateurs motivated by fun
 - publishing attacks for the prestige
- Increasingly, hackers are professional
 - attackers go underground
 - zero-day exploits are worth money
 - attackers include organized crime with lots of money and (hired) expertise
- Ransomware is an important game changer, as it allows attackers to monetise nearly anything.

Software (in)security

- There are no silver bullets!
- Crypto or special security features do not magically solve all problems
 - if you think your problem can be solved by cryptography, you do not understand cryptography and you do not understand your problem”
[Bruce Schneier]
- Security is an emergent property of entire system
 - (Non-functional) security aspects should be integral part of the design, right from the start

Software Security is often a secondary concern

- Security is always a secondary concern
 - Primary goal of software is to provide functionality& services;
 - Managing associated risks is a derived/secondary concern
- There is often a trade-off/conflict between
 - Security
 - Functionality
- where security typically loses out

Functionality vs Security

- Functionality is about what software should do,
- Security is (also) about what it should not do
- **Unless you think like an attacker, you will be unaware of any potential threats**

Lost Battles?

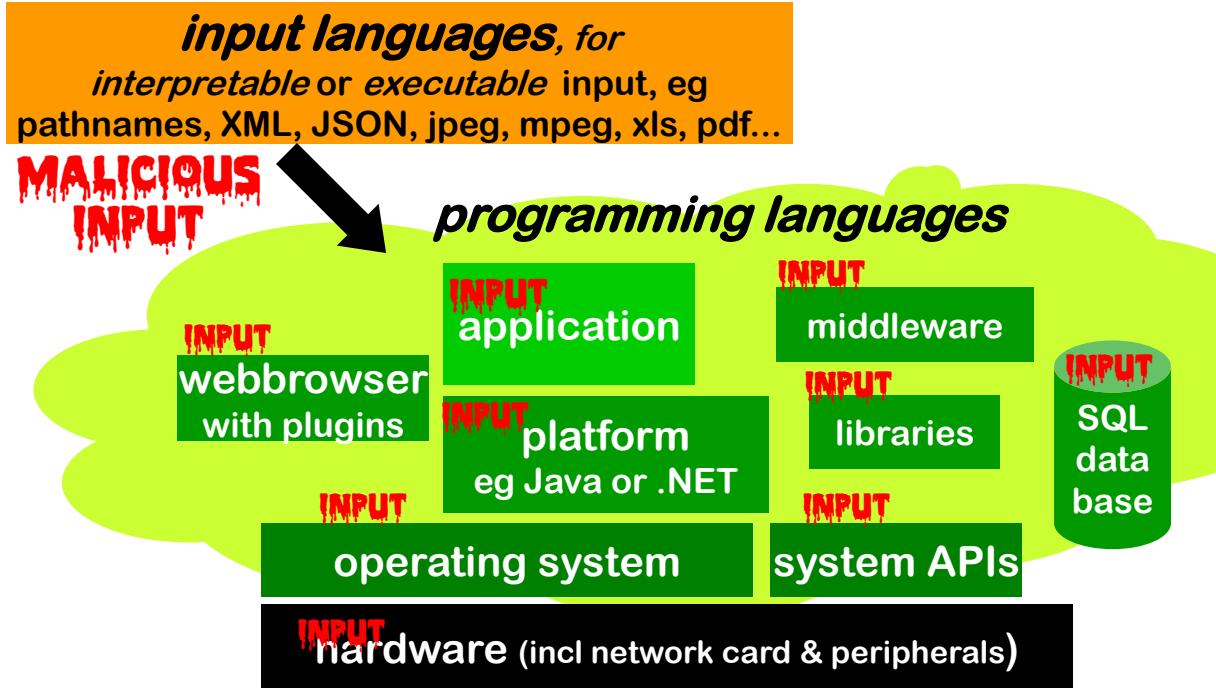
- operating systems (OSs)
 - with huge OS, with huge attack surface
- Traditional programming languages PENSATI PER ESSERE EFFICIENTI MA NON SICURI
 - with easy to use, efficient, but very insecure and error-prone mechanisms
- web browsers
 - with JavaScript, plug-ins for Flash & Java, access to microphone, web cam, location, ...
- email clients
 - which automatically cope with all sorts of formats & attachments

**"After writing PHP forum software for three years now,
I've come to the conclusion that it is basically impossible
for normal programmers to write secure PHP code.**

**It takes far too much effort. PHP's raison d'etre is that it
is simple to pick up and make it do something useful.
There needs to be a major push ... to make it safe for the
likely level of programmers - newbies.**

**Newbies have zero chance of writing secure software
unless their language is safe. ... "**

[Source <http://www.greebo.cnet/?p=320>]



THE SOFTWARE WEAKNESS

The weakness in depth

- Software runs on **a huge, complicated infrastructure**
 - HW, OS, platforms, web browser, lots of libraries & APIs, ...
- Software is built using **complicated languages**
 - programming languages and input languages (SQL, HTML, XML, mp4, ...)
- using **various tools**
 - compilers, IDEs, pre-processors, dynamic code downloads
- **All of these may have security holes, or may make the introduction of security holes very easy & likely**

Software Security: Recap

- A variety of problems are due to
 - lack of **awareness** of threats, but also of **what should be protected**
 - lack of **knowledge** of potential security problems, but also of **solutions**
- people choosing functionality over security
- compounded by complexity of software written in a variety of programming languages, using unsecure compilers, using large APIs, and running on huge infrastructure

Flaws vs Vulnerability

- **security weaknesses / flaws:**
 - things that are wrong or could be better done
- **security vulnerabilities**
 - Flaws that can actually be exploited by an attacker
 - Flaws are accessible: attacker has to be able to get at it
 - Flaws are exploitable: attacker has to be able to do some damage with it
- Example: turning off Wifi and BlueTooth network connection, many security vulnerabilities become flaws

Design vs Implementation Flaws

Design flaws

- vulnerability in the design

Implementation flaws aka code-level defects

- vulnerability in the software introduced during coding (either by the programmer or by the compiler tool chain)

Overall consensus: **implementation flaws and design flaws roughly equally common**

- Vulnerabilities also arise on other levels
 - configuration flaws when installing software on a machine
 - unforeseen consequence of the intended functionality (eg spam)

Implementation flaws

- For flaws introduced during coding, we can distinguish
- Flaws that can be understood **looking at the program itself**
 - simple typos, confusing two program variables, off-by-one
 - error in array access, errors in the program logic,...
- Flaws due to the interaction with the underlying platform **or other systems (compilers) and services**
 - buffer overflows in C(++) code
 - SQL injection, XSS, CSRF,.... in web-applications
 - Deserialisation attacks in many programming languages

Example: compilation

```
1 package Bank;  
2  
3 public class Account{  
4     private int balance = 0;  
5  
6     public void deposit( int amount ) {  
7         this.balance += amount;  
8     }  
9 }
```

Listing 1. Example of Java source code.

Immaginando di compilare il codice precedente in C, si ottiene che la classe viene implementata in termini di una struct, questa struct ha il campo balance a 0 e ha la possibilità di avere un puntatore al codice di una funzione che fa il deposito. La privacy del balance in questo caso non è più garantita, è possibile accedere al balance e si può assegnare un nuovo metodo allo struct al posto di quello definito dal programmatore.

Example compilation (cont.)

```
1 typedef struct account_t {
2     int balance = 0;
3     void ( *deposit ) ( struct Account*, int ) = deposit_f;
4 } Account;
5
6 void deposit_f( Account* a, int amount ) {
7     a->balance += amount;
8     return;
9 }
```

Listing 2. C code obtained from compiling the Java code of Listing 1.

Example Compilation (Cont)

When the Java code in Listing 1 interacts with other Java code, the latter cannot access the contents of `balance` since it is a private field. However, when the Java code is compiled into the C code in Listing 2 and then interacts with arbitrary C code, the latter can access the contents of `balance` by doing simple pointer arithmetic. Given a pointer to a C `Account` **struct**, an attacker can add the size (in words) of an `int` to it and read the contents of `balance`, effectively violating a confidentiality property that the source program had.

BISOGNA LAVORARE NON SOLO A LIVELLO DELLA PROGETTAZIONE DEL LINGUAGGIO DI PROGRAMMAZIONE MA ANCHE A LIVELLO DELLA STRUTTURA DEL RUN TIME E SUL PROCESSO DI COMPILAZIONE DUNQUE SULLE TECNICHE DI IMPLEMENTAZIONE DEL LINGUAGGIO DI PROGRAMMAZIONE.

Why?

This violation occurs because there is a discrepancy between what **abstractions** the **source language** offers and what **abstraction** the **target language** has.

The source languages provide powerful abstractions whose goal is allowing a programmer to write better code.

The source-level abstractions can be used to enforce security properties, but target languages that do not preserve such security properties are vulnerable to attacks.

Unfortunately, most target languages cannot preserve the abstractions of their source-level counterparts

Bad & Good News

- The bad news
 - people **keep making** the **same** mistakes
- The good news
 - people **keep making** the **same** mistakes
 - so we can do something about it!

Java code: spot the (security) flaws

```
int balance;

void decrease(int amount)
{ if (balance <= amount)
    { balance = balance - amount; }
else { printf("Insufficient funds\n"); }
}

void increase(int amount)
{ balance = balance + amount;
}
```

Possibili difetti:

- 1) in increase() amount può essere negativo
- 2) inversione della direzione del confronto nell'if di decrease
- 3) balance non è inizializzato
- 4) balance non è privato
- ...

Java code: spot the (security) flaws

```
int balance;

void decrease(int amount)
{ if (balance <= amount)
    { balance = balance - amount; }
else { printf("Insufficient funds\n"); }
}

void increase(int amount)
{ balance = balance + amount;
}
```

balance >= amount

Java code: spot the (security) flaws

```
int balance;  
  
void decrease(int amount)  
{ if (balance <= amount)  
    { balance = balance - amount; }  
else { printf("Insufficient funds\n"); }  
}  
  
void increase(int amount)  
{ balance = balance + amount;  
}
```

balance >= amount

What if amount is negative?

Java code: spot the (security) flaws

```
int balance;  
  
void decrease(int amount)  
{ if (balance <= amount)  
    { balance = balance - amount; }  
else { printf("Insufficient funds\n"); }  
}  
  
void increase(int amount)  
{ balance = balance + amount;  
}
```

balance >= amount

What if amount is negative?

What if balance + amount is too large for int?

What if amount is negative

lack of input validation of (untrusted) user input

could be a design flaw rather than an implementation flaw

```
balance >= amount
```

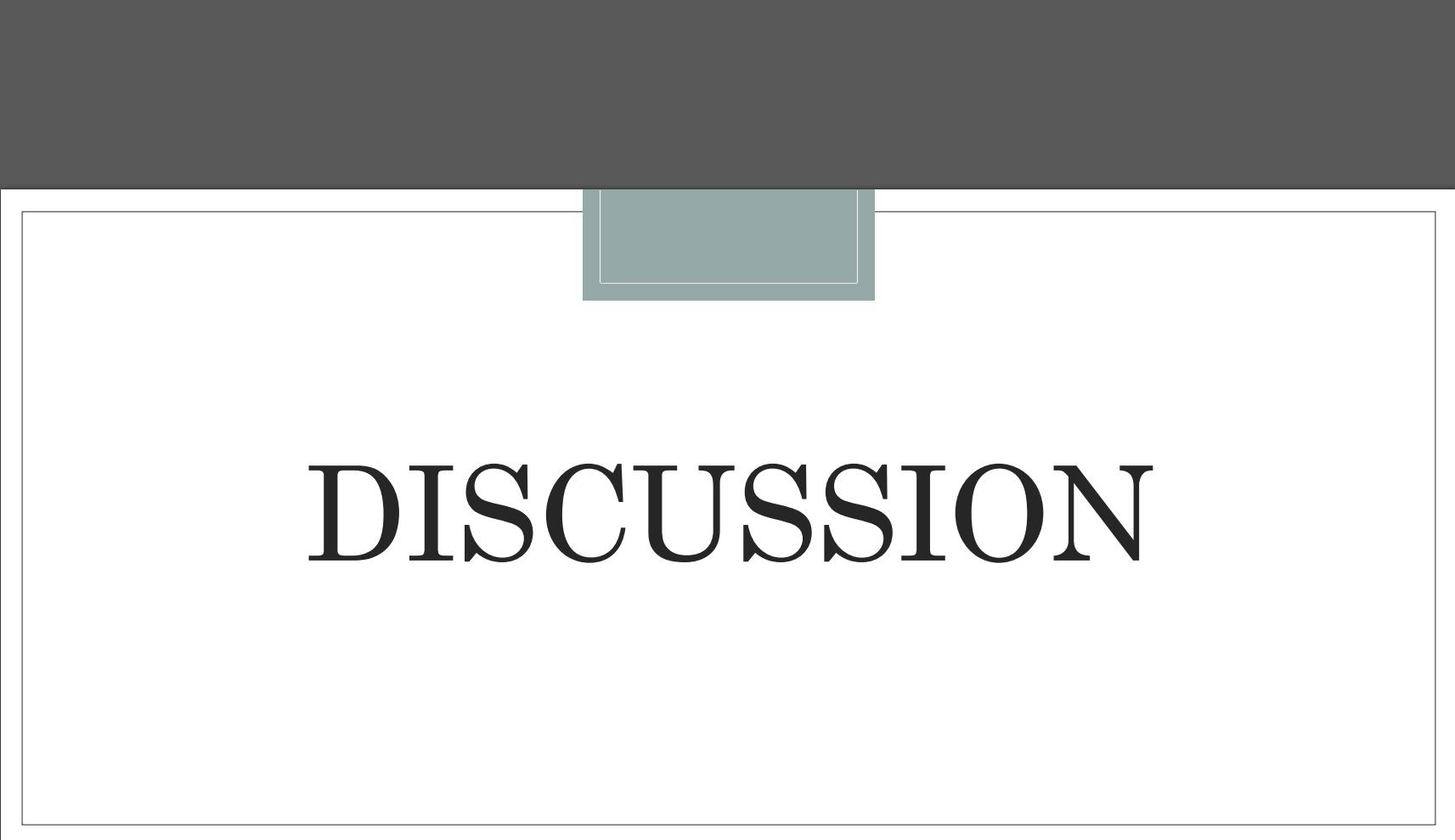
Logical (design) error

What if balance + amount is too large for a 64 bit int?

Lower level error

Interaction with the underlying virtual machine

Dipende da cosa c'è sotto per essere in grado di risolvere anche questo tipologie di problemi.



DISCUSSION