I progetti di ricerca a Verona:
Progettare le difese vs studiare gli attacchi

Dipartimento di Informatica
Università di Verona
MALWARE IN MODERN CYBERCRIME

MALICIOUS ACTOR → $ → CYBERCRIME MANAGER → $ → MALWARE EXPLOIT SERVERS

$ $ $ $ → COMPROMISED SITE → $ $ $ $
VULNERABILITIES

SYSTEM VULNERABILITIES

HUMAN ISSUE

Click Here

void sampleFunction() {
    char bufferA[50];
    char bufferB[16];
    printf("where do you live?");
    gets(bufferA);
    strcpy(bufferB,bufferA);
    return;
}

main() {
    printf("Hello World!");
    sampleFunction();
    printf("All done!");
}

BufferB
(Local variable)

BufferA
(Local variable)

FRAME POINTER

RETURN POINTER

Function call arguments

BACKDOOR

Attacker data

Attacker code

OVERFLOW

OVERWRITTEN

FRAME POINTER

RETURN POINTER

Function call arguments
SECURITY SCENARIOS

We strongly believe that a powerful defensive tool can be designed only by integrating (i) research on the target model analysis, and (ii) research on the threat model analysis, exploiting target vulnerabilities. Point (i) is central task of the project DEFiAnCE, while (ii) is developed inside the FIRB2013 project FACE - Formal Avenue for Chasing Malware (2014-2017). We are sure that the coexistence of these two projects in the same department would provide the ideal environment for generating new integration ideas resulting in an added value for both projects.

As far as the digital forensics theme is concerned, our University hosts a research group (led by Prof. Picotti) working on criminal law of computing and Dr. Flor is an active member of this group.

The research project will use the structures and the services of the Department of Computer Science of Verona, which is composed of 51 units in the academic staff, and of 10 units of technical and administrative staff. It also includes the lab SPY (Static Program Analysis by Abstract Interpretation). This Lab currently consists of 1 server and 5 PCs.

WHY ME?

The main scientific achievement since the beginning of my PhD is rooted in the formalization of the new framework of abstract noninterference (ANI) where we analyze the interference of properties instead of data, in this way we can tune the degree of interference we are interested in. In the years, I applied this promising approach to several contexts of program analysis and security: language-based security, slicing, obfuscation, malware detection and analysis. I believe that my strong formal expertise, in particular on ANI analyses, combined with the more applicative members of the research group both in Verona and abroad will provide a breeding ground for fruitfully reaching, or at least approaching, the expected results.

A.3 - BUDGET

| Budget O | Borne by other legal entities | 0 |
| Budget H | Borne by the host institution | 50.940 |
| MIUR M | Borne by MIUR | 294.937 |

A.1 - Staff of the host institution (professors, researchers, technicians, permanently employed; no to the fellows or graduate students, and so on, already under contract) | 50.940

A.2.1 - New contract for PI | 0

A.2.2 - New contracts for researchers | 144.608

A.2.3 - Fellows or similar already under contract | Not applicable

B - Overheads (60% A.1+A.2.1+A.2.2) | 117.329

C - Equipment | 9.000

D - Consulting services and similar | 6.000

E - Other operating costs | 18.000

F - costs borne by other legal entities | 0

G - Incentive for host institution | 0

Partial costs | 50.940

50.940 + 294.937 + 294.937 = 345.877

TOTAL COSTS: 0 €

TOTAL FUNDING: 294.937 €
FACE: Formal Avenue for Chasing malware

**METAMORPHISM**
*(static analysis)*

**MALWARE-ENVIRONMENT INTERPLAY**
*(dynamic analysis)*

---

**DORMANT BEHAVIOUR**

**CODE SHAPE MUTATION**

---

[SYMANTEC 2013]

2011 variants per malware rate 5:1
2012 variants per malware rate 38:1
**METAMORPHISM**

Self-modifying malware contains the metamorphic engine

**EXTRACT METAMORPHIC ENGINE BY**
**STATICALLY ANALYSING THE METAMORPHIC CODE**

**HARD**

**Phase semantics** [2010 VR] precisely models code changes, but leads to an undecidable detection scheme

Lose precision to gain decidability

\[ \forall P : \text{flow}(P) = \text{flow}(M) \Rightarrow P \text{ is variant of } M \]
METAPHOR SIGNATURE

Merge of the flow graphs (of system calls) after 100 mutations

Validation of the model on mobile malware
MALWARE-ENVIRONMENT INTERPLAY

Study **program interaction**: how the execution of a program interferes with the execution of another program.

Abstract Non-Interference (ANI) theory [2004 VR] on data

- Lift ANI theory on programs
- **HARD**

Analyse features that reveal its presence

- **ANTI-EMULATION**

Inputs that trigger the malicious behaviour

- **DORMANT BEHAVIOUR**

Validation of the model on mobile malware
• Analisi attiva del malware attraverso un gioco
• Valutazione empirica della metodologia
IA and MALWARE ANALYSIS

- Strumenti sviluppati:
  - ambiente per analisi empirica
  - algoritmi di analisi
- Problemi aperti:
  - Generazione automatica del modello
  - Raffinamento delle tecniche di analisi

Ambiente sperimentale

Metodologia analisi
DefiAnCE
A proactive Defence against Cyber Crime

• Goal 1: Data interference

• Goal 2: Program interference

• Goal 3: Data/program interference (Computer forensics)
AbScript

Static analyzer for dynamic languages (eg. PHP) based on abstract interpretation

Need to model and analyze the evolution of the code

Static analyzer which handles dynamic code mutations
TECHNICAL RISKS

ANALYSIS
PRECISE vs COMPLEXITY vs COVERAGE

PRECISE vs COMPLEXITY vs COVERAGE

PRECISE vs COMPLEXITY vs COVERAGE

SPY

STATIC EXHAUSTIVE

DYNAMIC PARTIAL

RECOVERY

HIGH COMPLEXITY
HARD

LOW COMPLEXITY

PRECEISE

LOOSE

Alan Turing

Kurt Godel
DESIGN and MEASURE CODE PROTECTION

Abstraction

For large/real programs control/data flow is too complex for being understandable by humans:

Reverse Engineering needs abstraction!
Reverse Engineering needs automated tools!

Abstract Interpretation is a general theory for approximating the semantics of dynamic systems (Cousot & Cousot 1977)
DESIGN and MEASURE CODE PROTECTION

Each tool is an Abstract Interpretation

We can quantify the security achieved by looking at proof complexity
DESIGN and MEASURE CODE PROTECTION

Transform code to make all tools blind

Removing noise means refining abstractions/complicating proofs

(Giacobazzi et al. 2000/2012)
DESIGN and MEASURE CODE PROTECTION

- Force the attacker to use automated tools (programs of large size and highly interconnected)
- Design code transformations making tools blind
- Determine lower bounds for proof complexity in obfuscated code
- Measure the degree of noise/slowdown induced in obfuscation

By constraining the adversary within a theorem prover we can quantify the security achieved from obfuscation.
PROJECTS

• FIRB-2013 (Coordinatore: Mila Dalla Preda)

• SIR 2014 (Coordinatore: Isabella Mastroeni)
  DefiAnCE: proactive DEFence against Cyber crimE (in fase di valutazione da parte del Ministero)

• Joint Project (Coordinatore: Isabella Mastroeni)
  AbScript: Abstract Interpretation based Analysis of Scripting Languages (settembre 2014 - agosto 2016)

• Joint Project (Coordinatore: Roberto Giacobazzi)
  Interpretation-based design and measurement of code-protecting transformations (settembre 2014 - agosto 2016)
Isabella Mastroeni
Abstract Interpretation, Malware analysis, non-interference, static analyzer PHP

Fausto Spoto
Abstract Interpretation, static analyzer PHP

Isabella Mastroeni
Abstract Interpretation, Malware analysis, non-interference, static analyzer PHP, game theory and AI

Alessandro Farinelli
game theory and AI

Mila Dalla Preda
Abstract Interpretation, Malware analysis, non-interference, static analyzer PHP