La ricerca in sicurezza informatica

Il punto di vista di un ingegnere del software

Barbara Russo
Faculty of Engineering - Free University of Bolzano

Bolzano, 19 aprile 2023

Il cyber risk

- L’attacco informatico
- La tecnologia software e la supply chain
- Il fattore umano

R. Baldoni CyberSec2023

L'attacco

Remote Access Trojan (RAT)

- *Consente a un aggressore di prendere il controllo del computer della vittima da remoto*
  - Ottenere l'accesso non autorizzato ai sistemi
  - Rubare informazioni sensibili e
  - Svolgere attività dannose dal sistema della vittima

RAT - come accede

- Può essere installato sul computer della vittima accedendo tramite
  - **tattiche di social engineering**
    - **e-mail di phishing**
    - **a lungo termine**
    - **raccolta di dati**
  - **vulnerabilità del software**
- Una volta installato, il RAT può essere eseguito silenziosamente in background, consentendo all'aggressore di monitorare e controllare il sistema compromesso all'insaputa della vittima

Misure di protezione

- utilizzo di password forti e uniche
- evitare e-mail e download sospetti
- aggiornamento del software e dei sistemi operativi

Naturalmente non basta …

Attacco - Ricerca

- Prevenzione
- Rilevamento
- Risposta agli incidenti

R. Baldoni CyberSec2023

Attacco - Ricerca

- Prevenzione
- Rilevamento
- Risposta agli incidenti

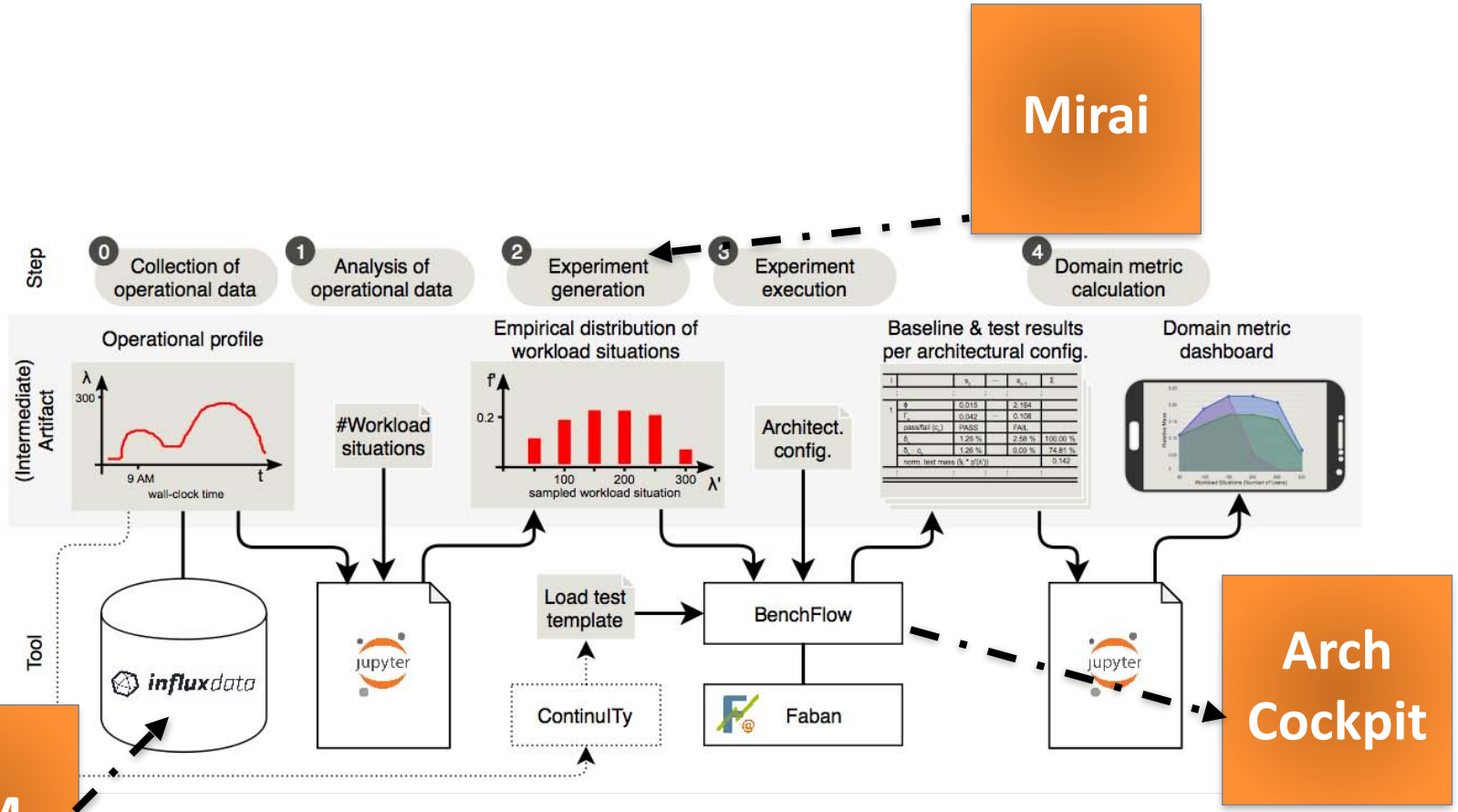
R. Baldoni CyberSec2023

Rilevamento tempestivo

- Idea:
  - Rilevare e prevenire gli effetti degli attacchi tramite lo studio delle anomalie nelle prestazioni e colli di bottiglia del sistema (per es. anomalie nei tempi di risposta)

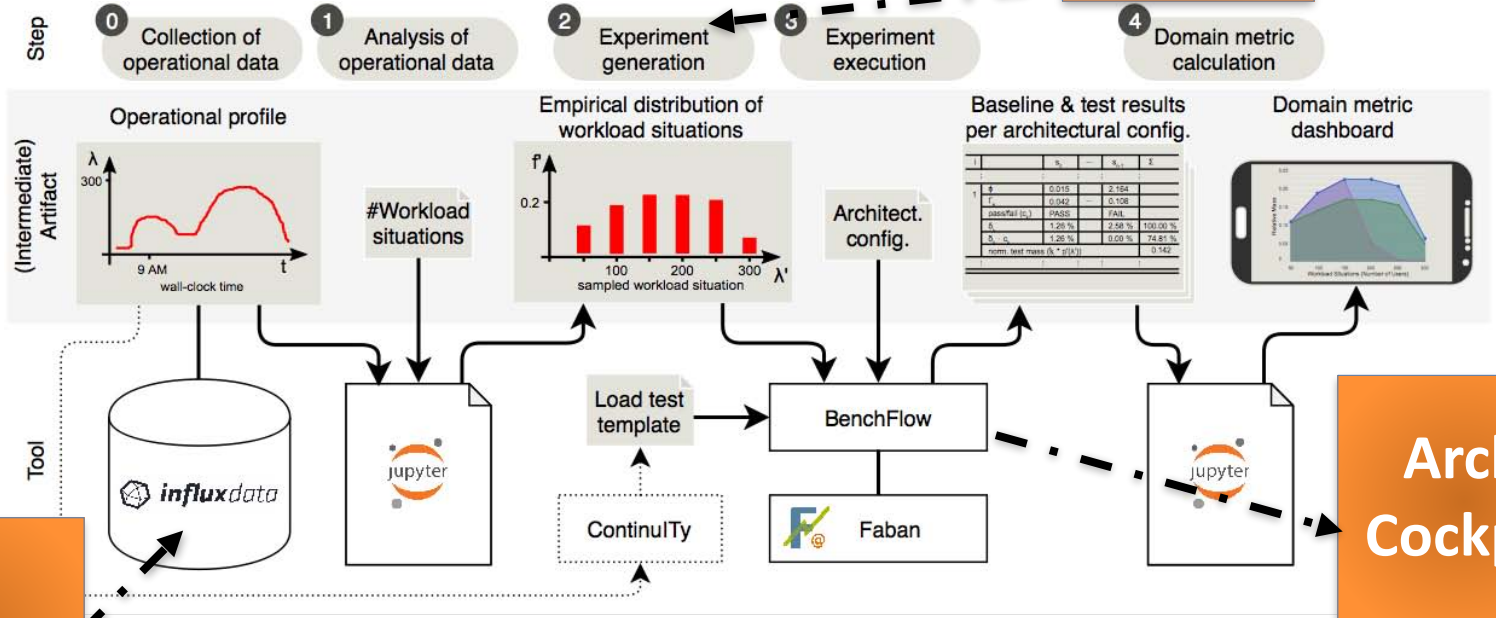
Alberto Avritzer, Vincenzo Ferme, Andrea Janes, Barbara Russo, André van Hoorn, Henning Schulz, Daniel S. Menasché, Vilc Queupe Rufino: *Scalability Assessment of Microservice Architecture Deployment Configurations: A Domain-based Approach Leveraging Operational Profiles and Load Tests*. J. Syst. Softw. 165: 110564 (2020)

Rilevare anomalie di prestazioni

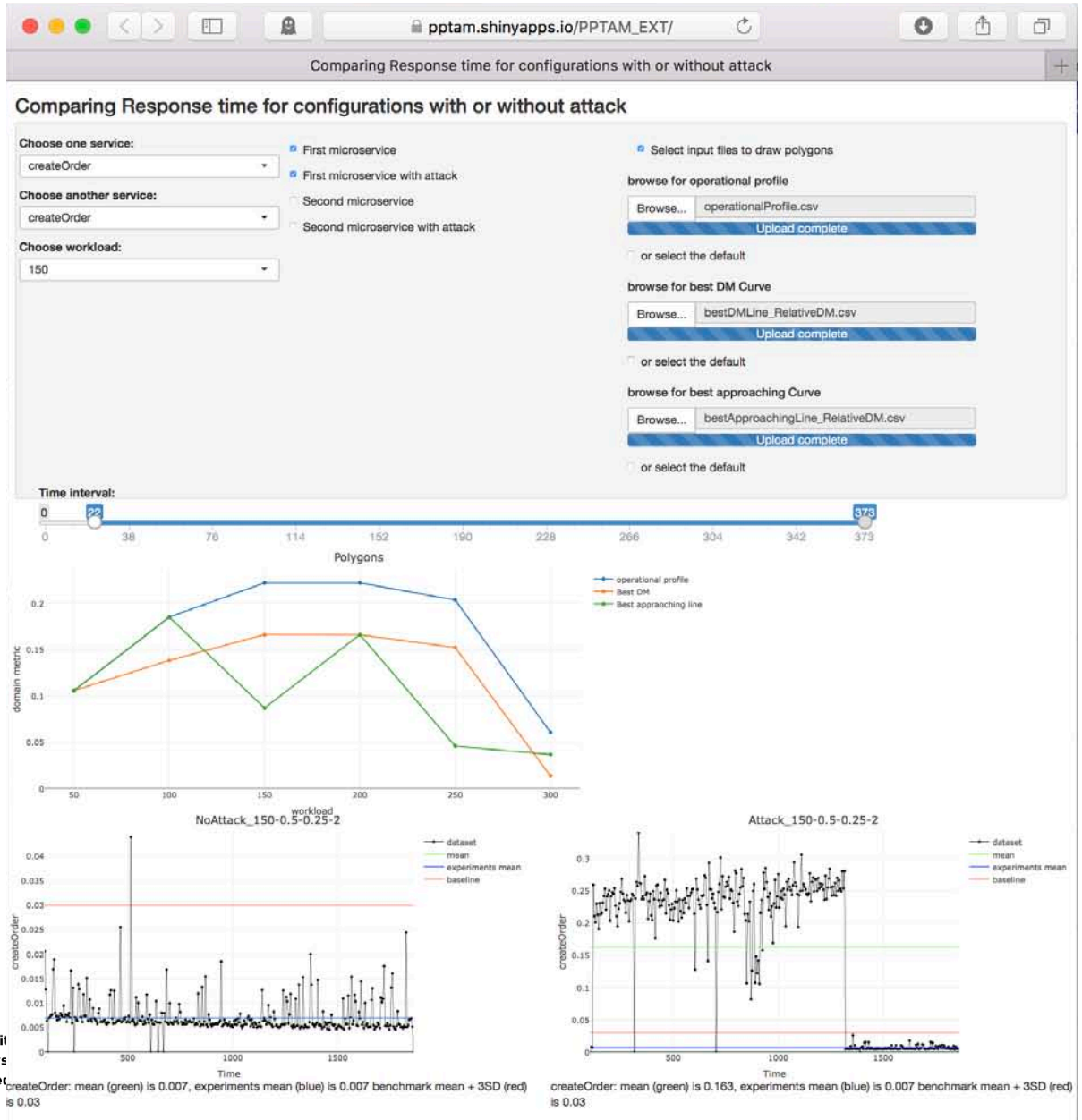


Rilevare anomalie di prestazioni

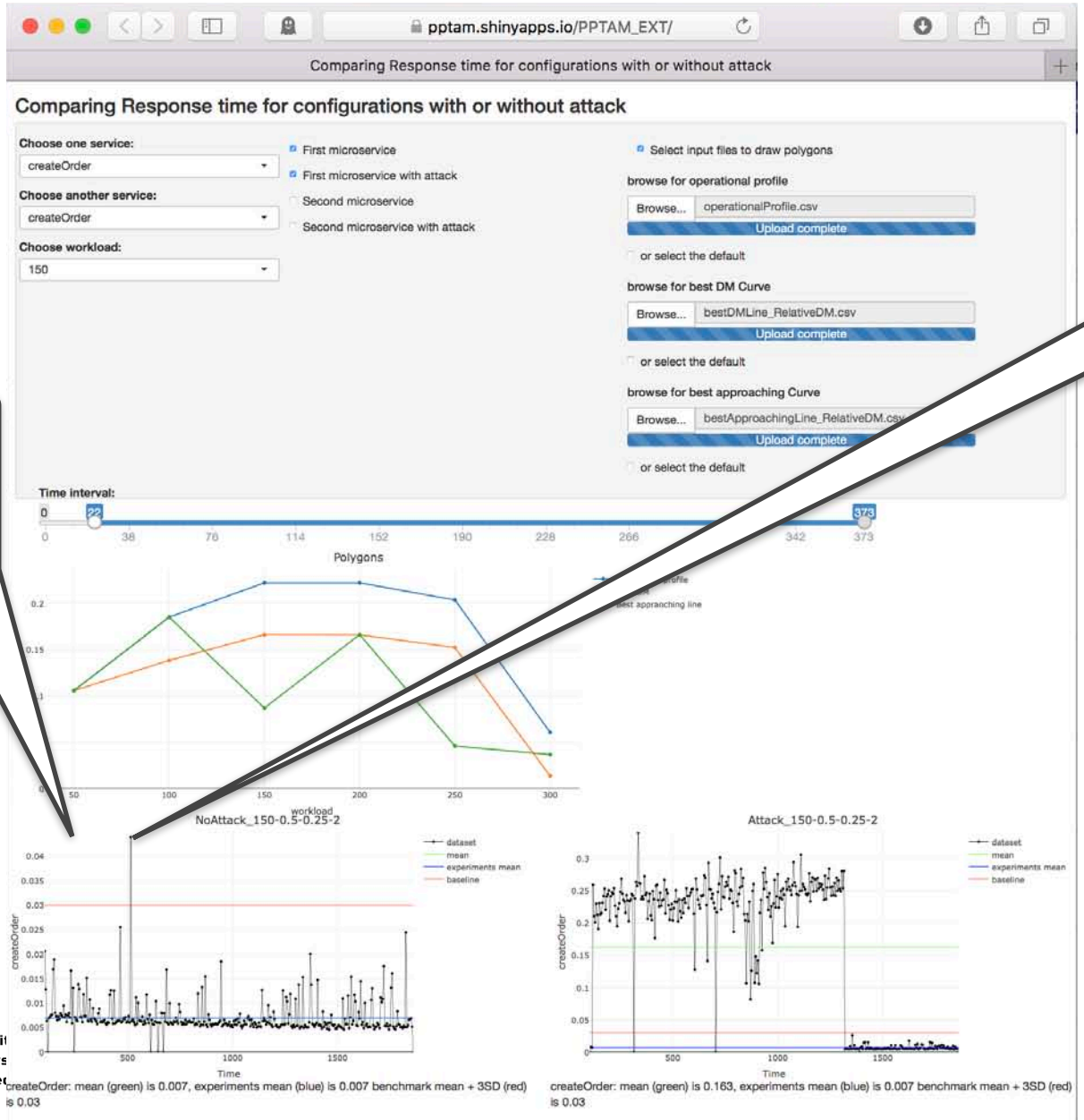
- Microservizi sistema telecomunicazioni Ericsson
- Microservizi sistema Trainticket



Monitoring cockpit



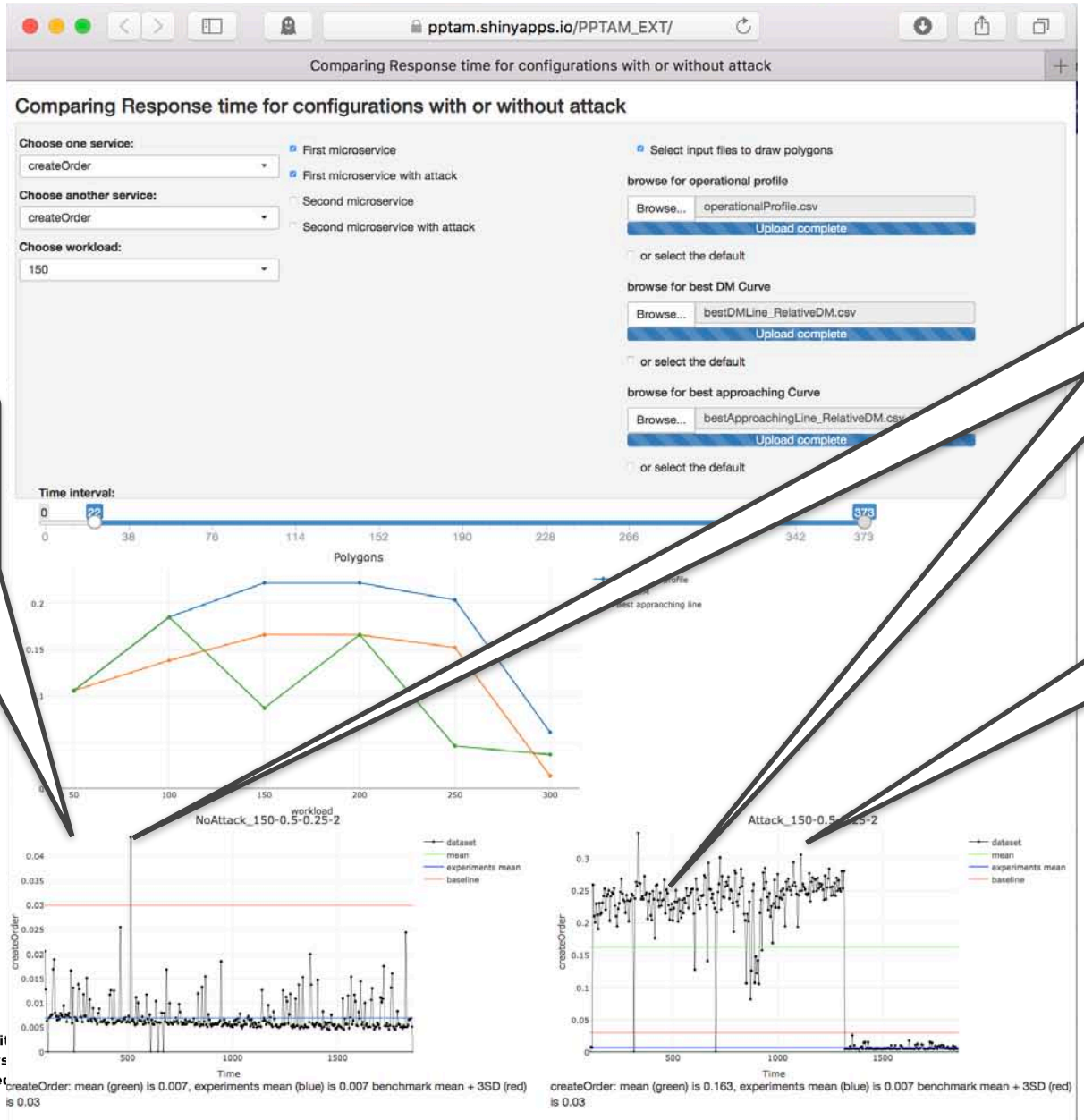
Monitoring cockpit



Nessun attacco

Fallimento sporadico del servizio

Monitoring cockpit



Nessun attacco

Fallimento del servizio

Sotto attacco

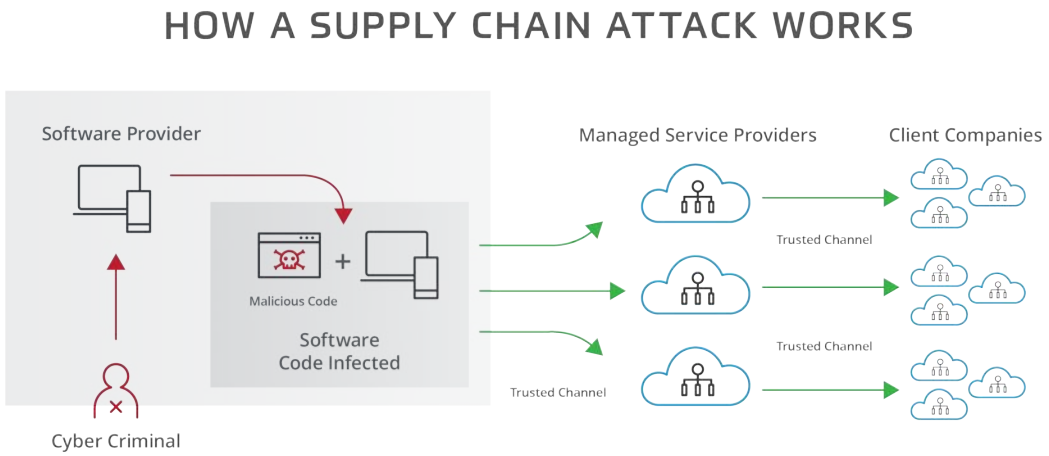
La tecnologia software

Tecnologia software - ricerca

- Affidabilità della tecnologie e della catena della distribuzione e tecnologie di terze parti (**technology supply chain**)
- **Processo di sviluppo** della tecnologia software
- **Competenze** in sicurezza informatica

Catena di distribuzione software

- Nel 2021 si è registrata una crescita del 650% su base annua degli attacchi alla sicurezza che hanno sfruttato la catena di distribuzione di software open source
- Molti aggressori prendono di mira i gestori di software di parti terze e i loro utenti per attaccare il software che li usa



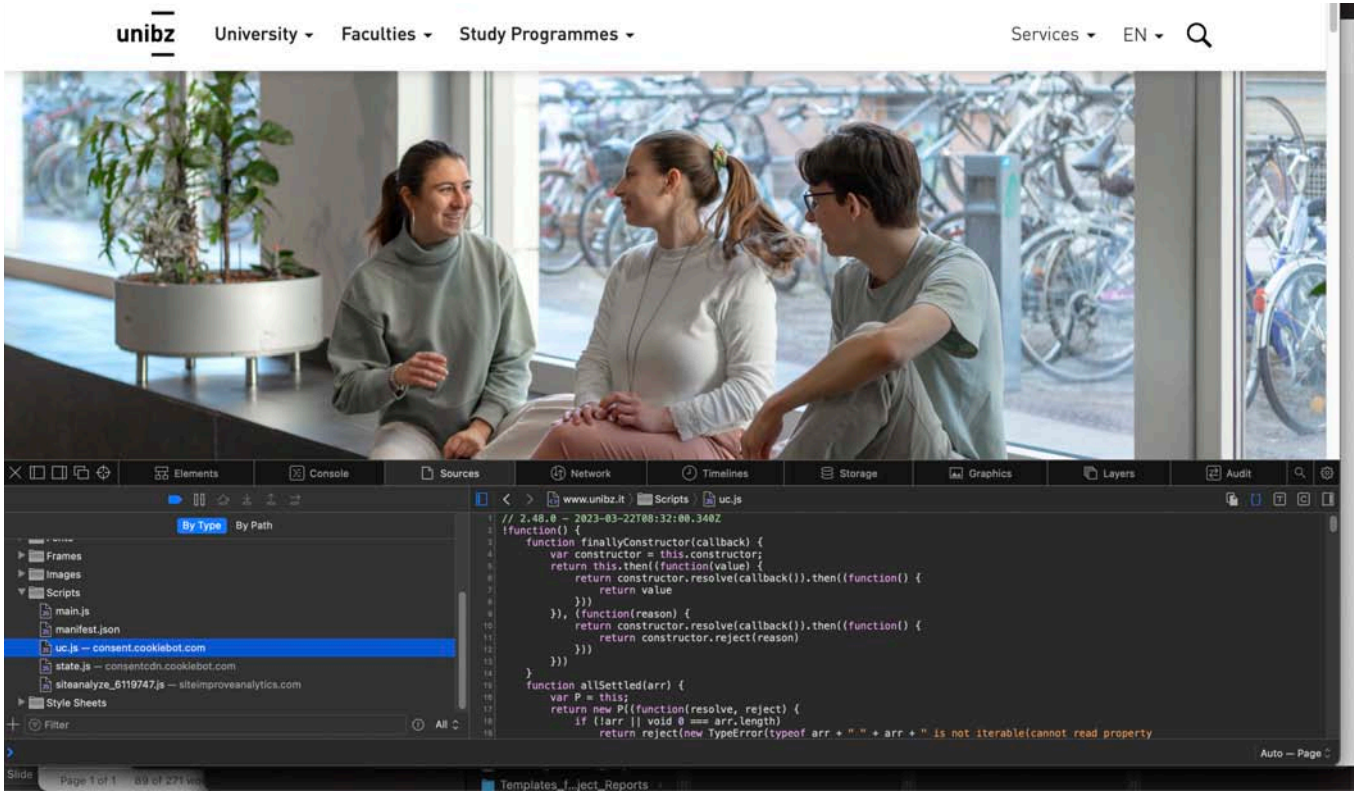
Ricerca - il caso *npm*

- *npm* è l'acronimo di *Node Package Manager*, un'applicazione e un repository popolare per lo sviluppo e la condivisione di codice *JavaScript (pacchetti)*
  - PayPal, LinkedIn e Groupon usano js

N. Zahan, T. Zimmermann, P. Godefroid, B. Murphy, C. Maddila and L. Williams, "What are Weak Links in the npm Supply Chain?," *2022 IEEE/ACM 44th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, Pittsburgh, PA, USA, 2022, pp. 331-340, doi: 10.1145/3510457.3513044.

Scripts nelle pagine web

```
<!DOCTYPE html>
<html>
  <head>
    <title>Esempio di codice HTML che carica uno script esterno</title>
    <script src="https://sito-esterno.com/script.js"></script>
  </head>
  <body>
    <h1>Benvenuto!</h1>
    <p>Nuovo sito</p>
  </body>
</html>
```

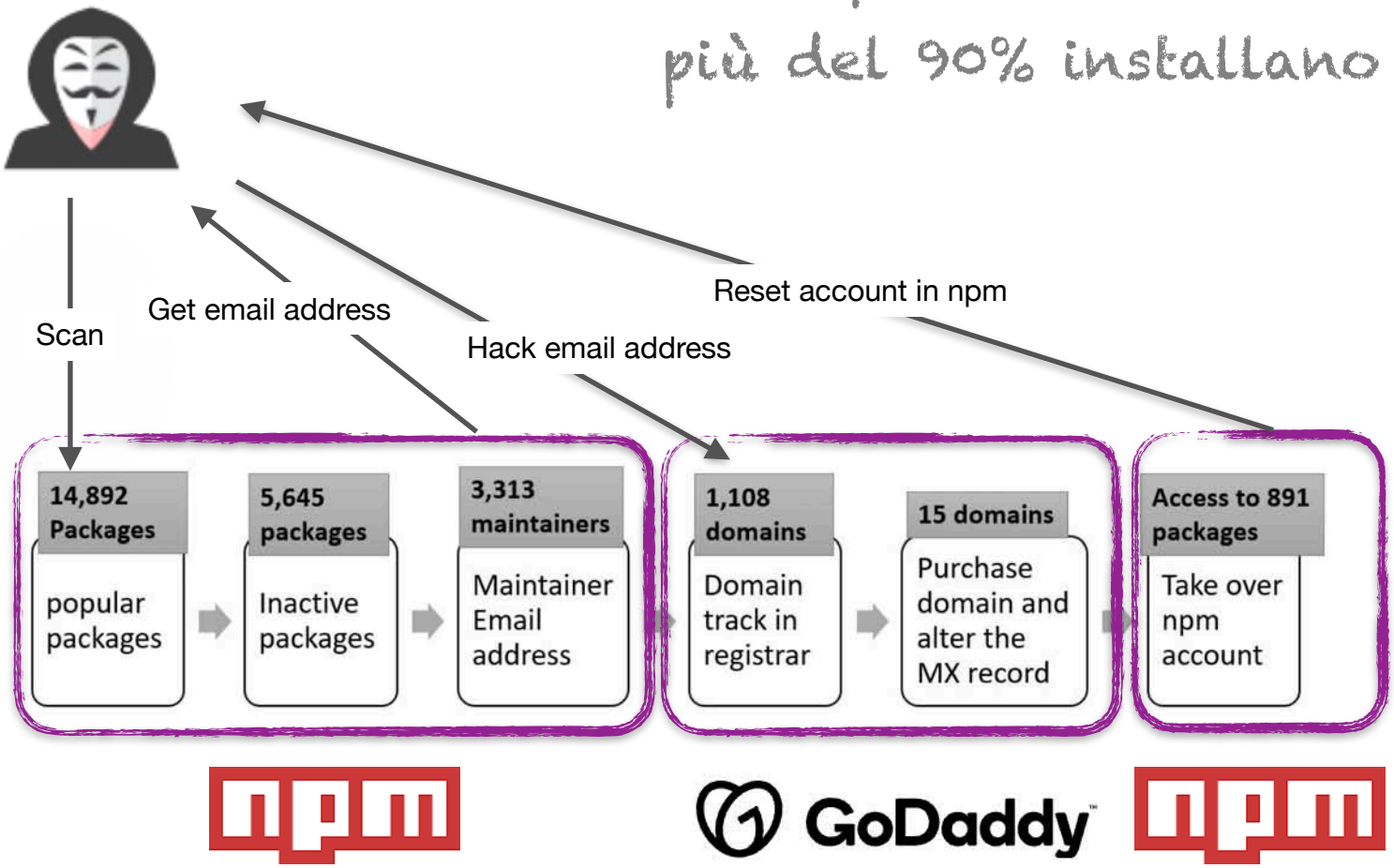


Il caso *npm* - vulnerabilità

- 2022. Analizzati 1,63 milioni di pacchetti JavaScript gestite in *npm*
- Individuati sei indicatori di debolezza dei pacchetti JavaScript in *npm*

Alcuni Risultati

3,635 pacchetti dannosi;
più del 90% installano script

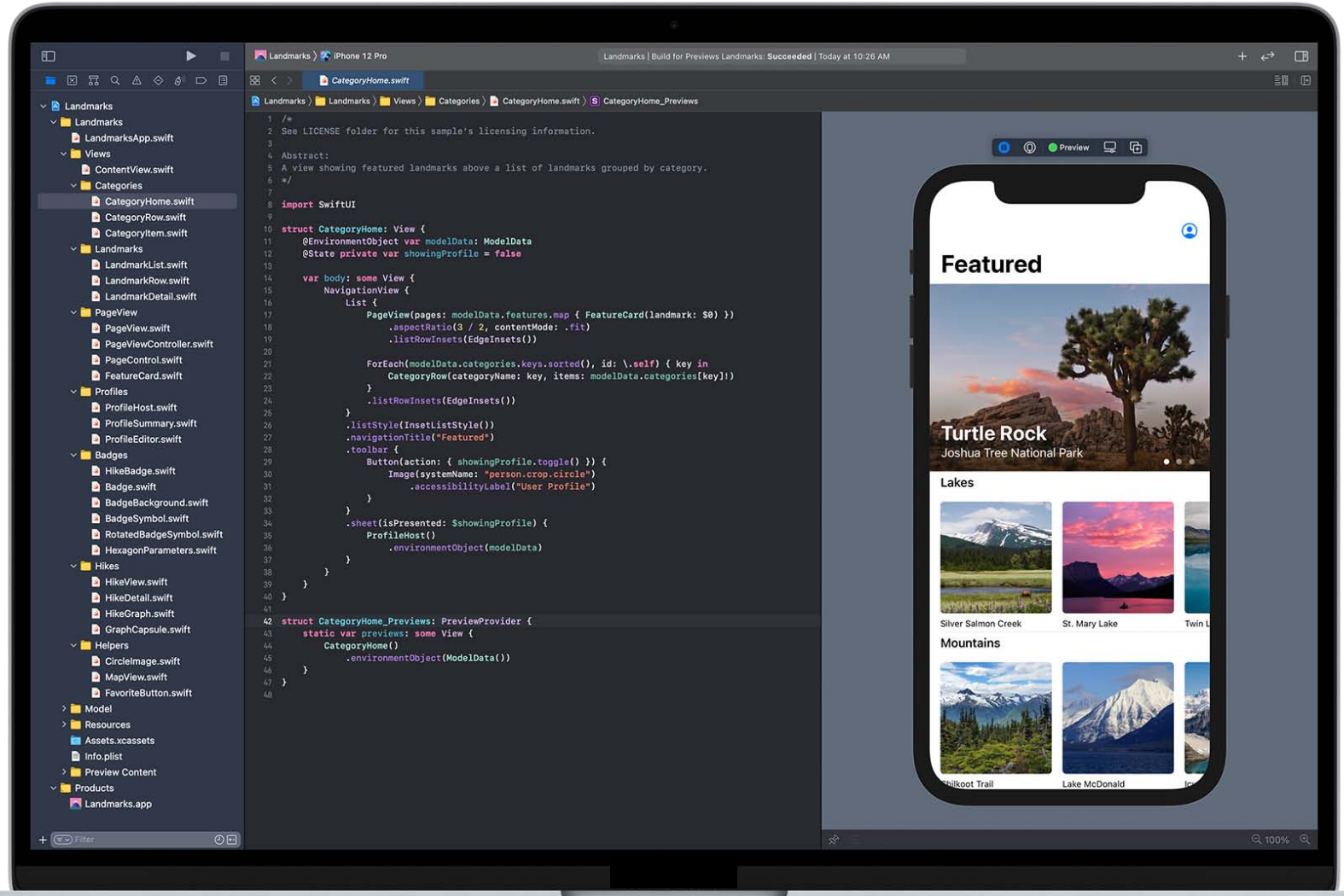


Lessons learned

- La sicurezza va garantita
  - a livello di sistema e non solo per singola tecnologia
  - sulla catena dei fornitori
  - durante lo sviluppo e la manutenzione del software

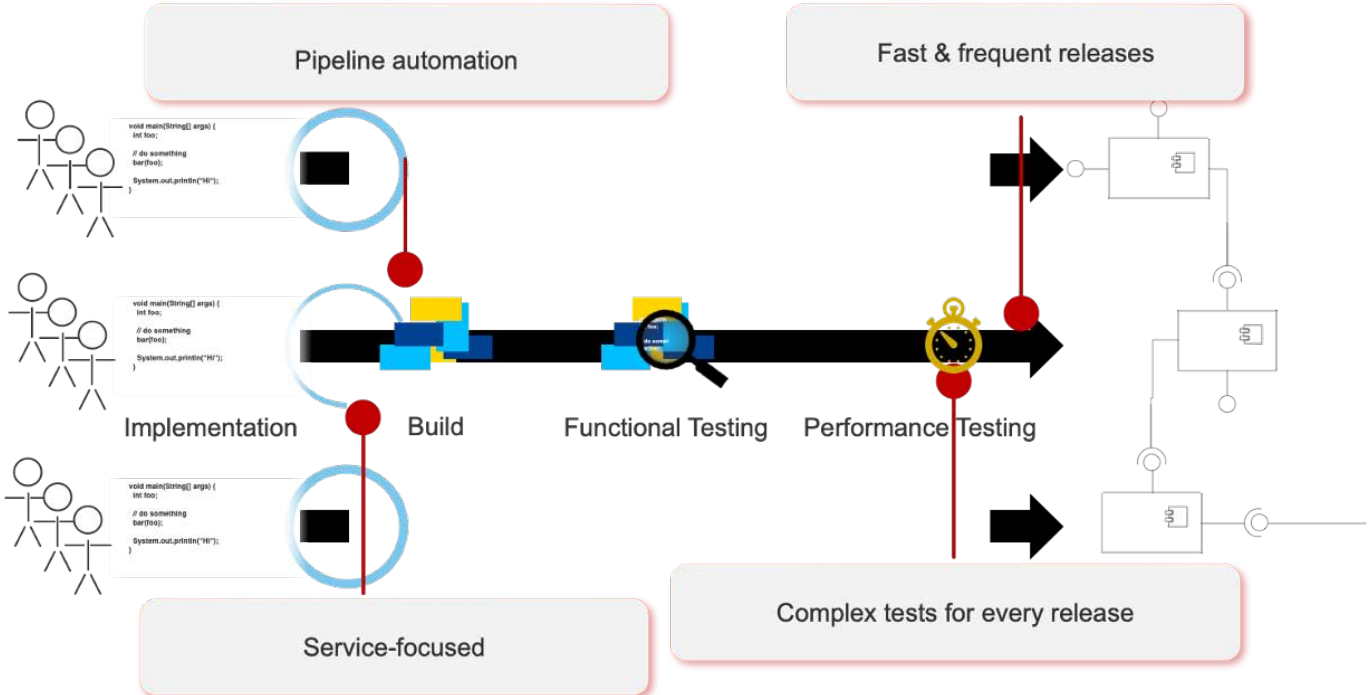
Processo di sviluppo del software

Processo di sviluppo del software



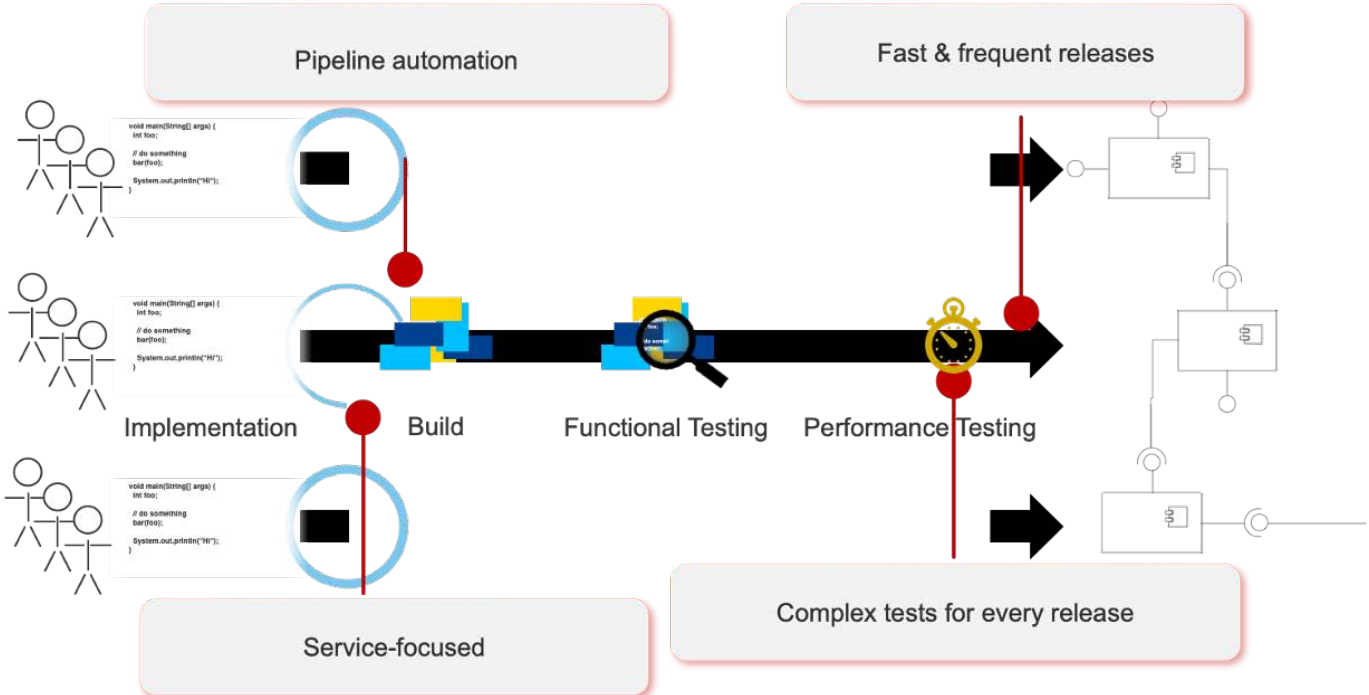
Processo di sviluppo moderno

- DevOps
  - Test automatizzato in pipelines
  - Rilasci frequenti
  - Installazione diretta nell'ambiente dell'utente



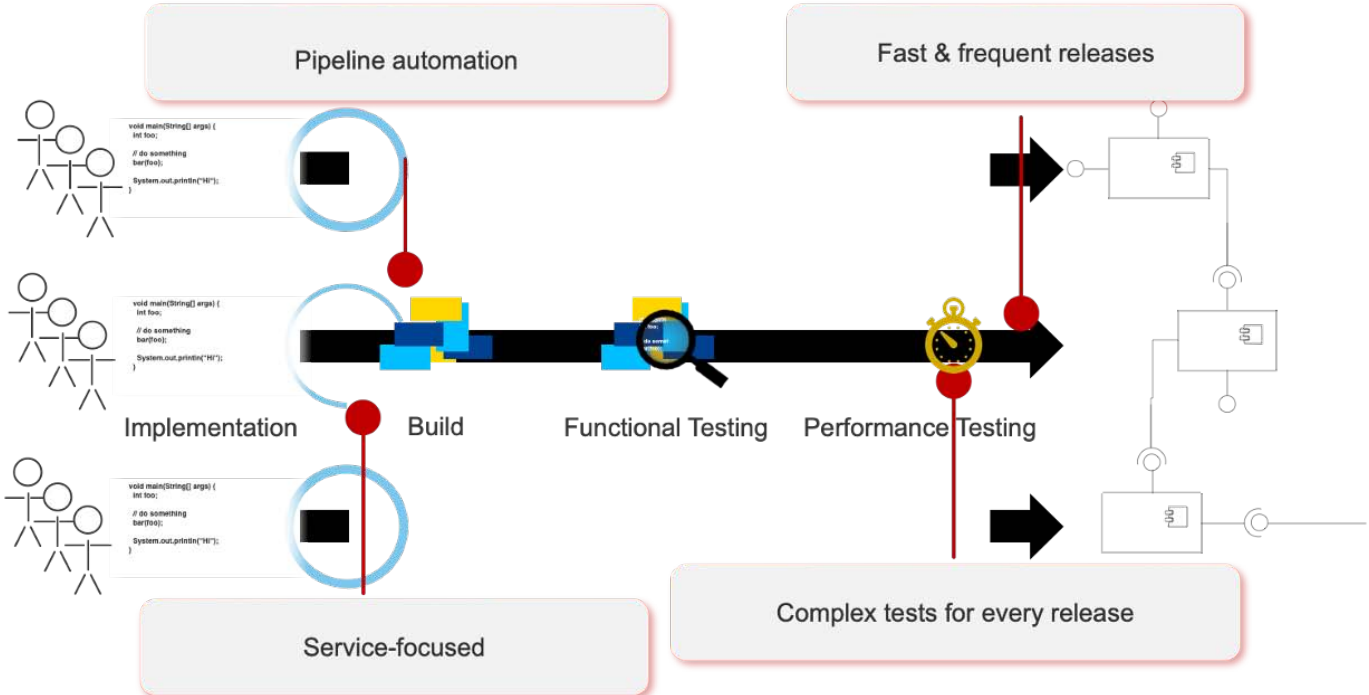
Processo di sviluppo moderno

- DevOps
  - Test automatizzato in pipelines
  - Rilasci frequenti
  - Installazione diretta nell'ambiente dell'utente



Processo di sviluppo moderno

- DevOps
  - Test automatizzato in pipelines
  - Rilasci frequenti
  - Installazione diretta nell'ambiente dell'utente



Sicurezza durante lo sviluppo

Gli sviluppatori e gli operatori sono esperti di sicurezza?

Gli sviluppatori sono consapevoli delle vulnerabilità del loro software?

Pur sapendo che il loro codice non è perfetto, sanno che il codice non è nemmeno sicuro?

Sviluppo software - Ricerca

- Per rispondere abbiamo analizzato il SATD

Self-admitted technical debt (SATD)

Gli sviluppatori sono consapevoli del technical debt e lo ammettono nei loro commenti al codice

Codice non del tutto corretto per il quale lo sforzo di renderlo corretto aumenta con il tempo come una sorta di interesse.

Funziona ma potrebbe essere migliore

Funziona in questo caso ma non sempre

E' scritto male ma funziona

Commenti nel codice

```
//FIXME isn't this a nice mess of a client? read input, write
splits, read splits again.

//FIXME? YarnRuntimeException is for runtime exceptions only.

//TODO: don't repeat this ugly cast logic (maybe use isCastable in
the last else block).
```

SATD sopravvive a lungo in un programma

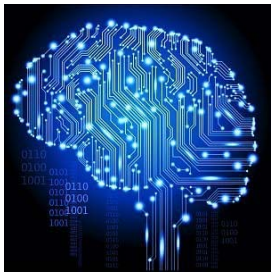
Bavota, G. and Russo, B., 2016 A large-scale empirical study on self-admitted technical debt.
In *Proceedings of the 13th international conference on mining software repositories (MSR 2016)* pp. 315-326.

Il codice relativi a SATD contiene debolezze che possono influire sulla sua sicurezza

Russo, B., Camilli, M., Mock, M.
WeakSATD: detecting weak self-admitted technical debt, the 19*th International conference on mining software repositories (MSR2022)*

Più a lungo un codice SATD rimane nel programma, più alto è il rischio che le debolezze possano essere sfruttate

Machine Learner codeBERT



MITRE

ATT&CK®

CWE

CVE®

EXPLOIT DATABASE

```
/* FIXME: this code assumes that sigmask is an
     even multiple of the size of a long
     integer. */
{
unsigned long *src = (unsigned long const *)
    set;
unsigned long *dest = (unsigned long *)&(
    thread.p->sigmask);
switch (how)
{
    case SIG_BLOCK:
    for (i = 0; i < (sizeof (sigset_t)
        sizeof (unsigned long)); i++)
    {
      /* OR the bit field longword-wise. */
      *dest++ |= *src++;
    }
    break;
    case SIG_UNBLOCK:
    for (i = 0; i < (sizeof (sigset_t)
        sizeof (unsigned long)); i++)
    {
      /* XOR the bitfield longword-wise. */
      *dest++ ^= *src++;
    }
    case SIG_SETMASK:
    /* Replace the whole sigmask. */
    memcpy (&(thread.p->sigmask), set, sizeof
        (sigset_t));
    break;
}
```

Circa 700K funzioni in C

CWE issue

Title	**CWE-242: Use of Inherently Dangerous Function**
Description	The program calls a function that can never be guaranteed to work safely
Extended Description	Certain functions behave in dangerous ways regardless of how they are used. ... .
Modes of Introduction:	Phase: Implementation
Applicable Platform	Languages: C, C++
Common Consequences	Technical Impact: Varies by *Context*
Likelihood Of Exploit	High
Demonstrative Example in C	

```
char buf[80];
void vulnerable() {
    int len = read_int_from_network();
    char *p = read_string_from_network()
        ;
    if (len > sizeof buf) {
        error("length too large, nice
            try!");
        return;
    }
memcpy(buf, p, len);
```

Potential mitigations	Phases: Implementation; Requirements ban the use of dangerous functions. Use their safe equivalent. Phase: Testing; Use grep or static analysis tools to spot usage of dangerous functions.

void *memcpy(void *dest, const void *src, size_t n);
Definition of size_t n is an unsigned int size_t
If len is negative, may copy huge amounts of input into buf as neg values are set to MAXINT+1: len can also be set to exceed the bus capacity

SATD & SATD block

```
1  /* FIXME: this code assumes that sigmask is an
        even multiple of the size of a long
      integer. */
2
3  unsigned long *src = (unsigned long const *)
       set;
4  unsigned long *dest = (unsigned long *) &(
       thread.p->sigmask);
5
6  switch (how)
7  {
8      case SIG_BLOCK:
9      for (i = 0; i < (sizeof (sigset_t) /
           sizeof (unsigned long)); i++)
10     {
11       /* OR the bit field longword-wise. */
12       *dest++ |= *src++;
13     }
14     break;
15     case SIG_UNBLOCK:
16     for (i = 0; i < (sizeof (sigset_t) /
           sizeof (unsigned long)); i++)
17     {
18       /* XOR the bitfield longword-wise. */
19       *dest++ ^= *src++;
20     }
21     case SIG_SETMASK:
22     /* Replace the whole sigmask. */
23     memcpy (&(thread.p->sigmask), set, sizeof
           (sigset_t));
24     break;
25 }
```

Risultati

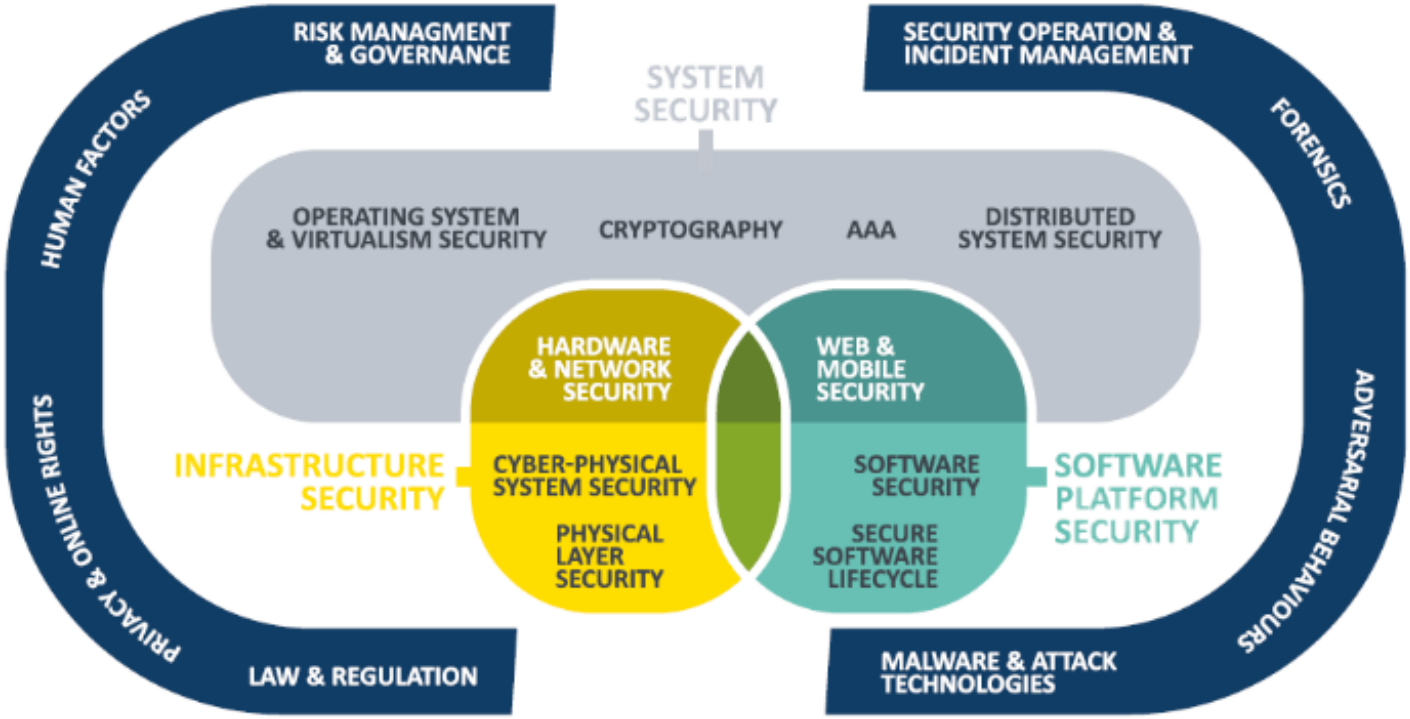
- La quantità di codice “brutto” aumenta nel tempo
  - Solo il 57% viene rimosso
  - In media, sopravvive a 1000 commit
- Analizzate più di 700,000 funzioni in C e i loro commenti
  - il 55% di SATD contiene almeno una vulnerabilità e di 14 tipi diversi (MITRE-CWE)

Il fattore umano

Il fattore umano

- *In Italia: rapporto Censis-DeepCyber 2022*
- Utenti
  - Mancanza di consapevolezza: **3 su 4 dei cittadini non ha un'idea** di cosa sia la sicurezza informatica
- Formazione
  - Oltre il 40% dei dirigenti **non ha ancora una formazione** specifica sulla sicurezza informatica

Competenze necessarie in sicurezza informatica



Fonte: GCSEC "Mind the Gap" - 2019, Studio condotto da GCSEC con l'Università di Oxford

Qualche parola sulla

Situazione nazionale

Situazione nazionale

- **ACN**: Agenzia per la Cybersicurezza Nazionale fondata nel 2021
  - *National Hyper Security Operations Center* (SOC)
  - 86 PNRR initiatives
- **CERT- PA** *Computer Emergency Response Team per la Pubblica Amministrazione*

Qualche parola su

UNIBZ

unibz - attività in corso

- Specialisti
  - Due corsi dedicati alla sicurezza informatica e alla sicurezza del software
  - Scuola ISE sulla cybersecurity in collaborazione con l'Università di Innsbruck
- Eventi:
  - Capture the flag con Würth Phoenix srl, University of Verona, University of Trento
  - Cybersecurity.it

Aperta a professionisti e studenti

Riepilogo

Come possiamo proteggere le nostre aziende?

- **Coordinamento**
  - SOC provinciale in rete
- **Infrastruttura**
  - **CSLab** *CyberSecurity Laboratory al NOI*- proposta di progetto bando *FESR 2021-2027* in collaborazione con *IAA* e *Würth Phoenix srl*
- **Specialisti e formazione**
  - Nuova *borsa di studio di dottorato in sicurezza informatica* cofinanziata dal **PNRR** e Würth Phoenix
  - Corsi ed eventi per studenti (Capture the Flag)

UNIBZ Team

Grazie per l'attenzione

brusso@unibz.it

ISE 2023@unibz - 10-12 luglio

- Laurie Williams, North Carolina State University, US
  - Software Supply Chain Security
- Rui Abreu, University of Porto, Portugal
  - On Making Software More Secure
- Mariano Ceccato, University of Verona, Italy
  - Security Testing of Mass Assignment Vulnerabilities in RESTful APIs
- Stefano Zanero, Politecnico di Milano, Italy,
  - When software vulns got real: security of cyber-physical systems

https://seschool-series.github.io/2023/