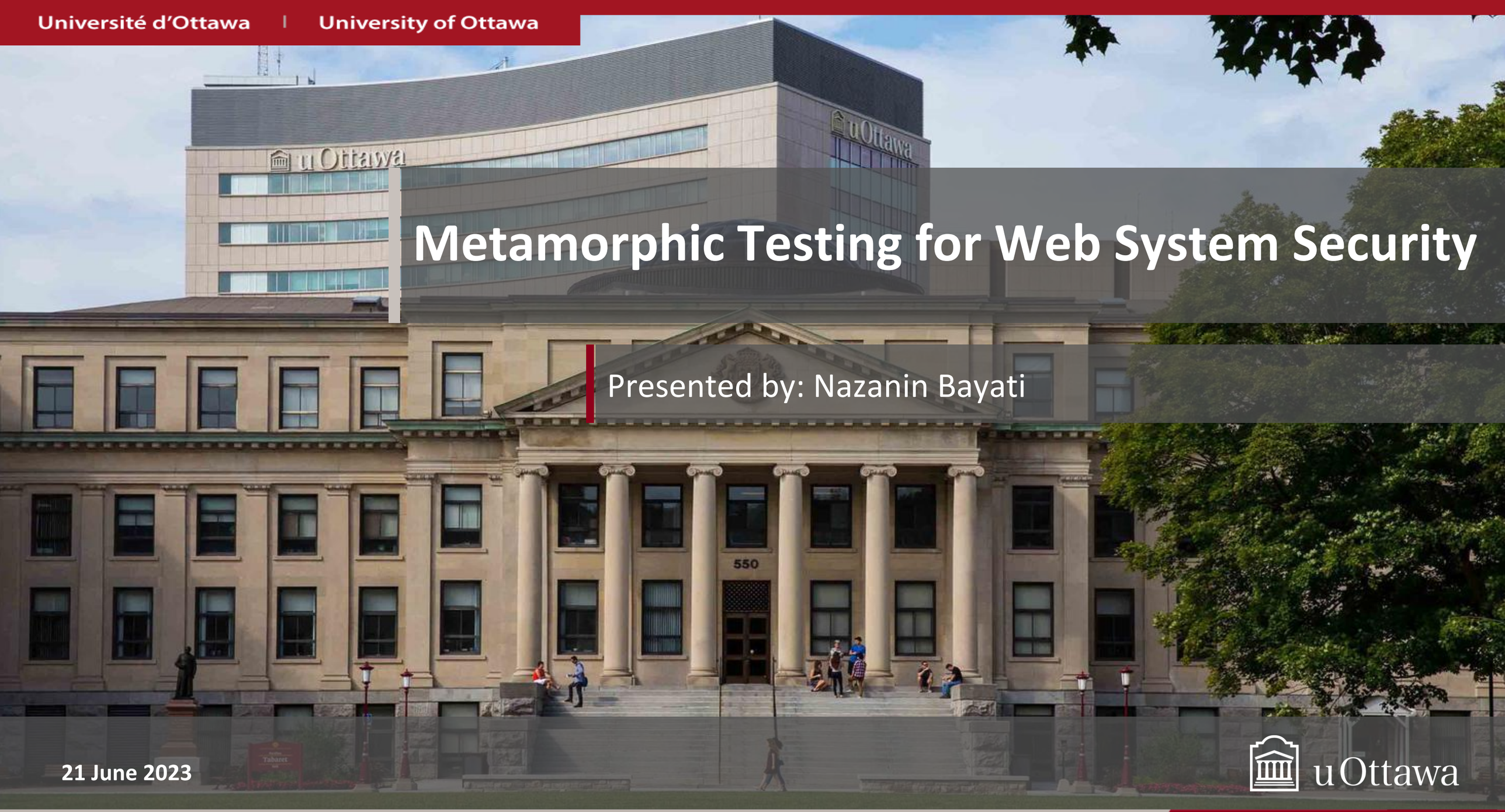


Metamorphic Testing for Web System Security

Presented by: Nazanin Bayati

21 June 2023

uOttawa

About me

- I am Nazanin Bayati
- Ph.D. Candidate in Computer Engineering at uOttawa
- Research field: Web system security testing
- Supervisor: Prof. Lionel Briand

Web Systems

- **Web systems** are one of the main means of delivering online services
  - E.g., e-commerce and online banking
- These systems:
  - manage critical assets (e.g., card transactions)
  - and often store sensitive information (e.g., customer data)
- It is essential to verify that the web systems are **secure**

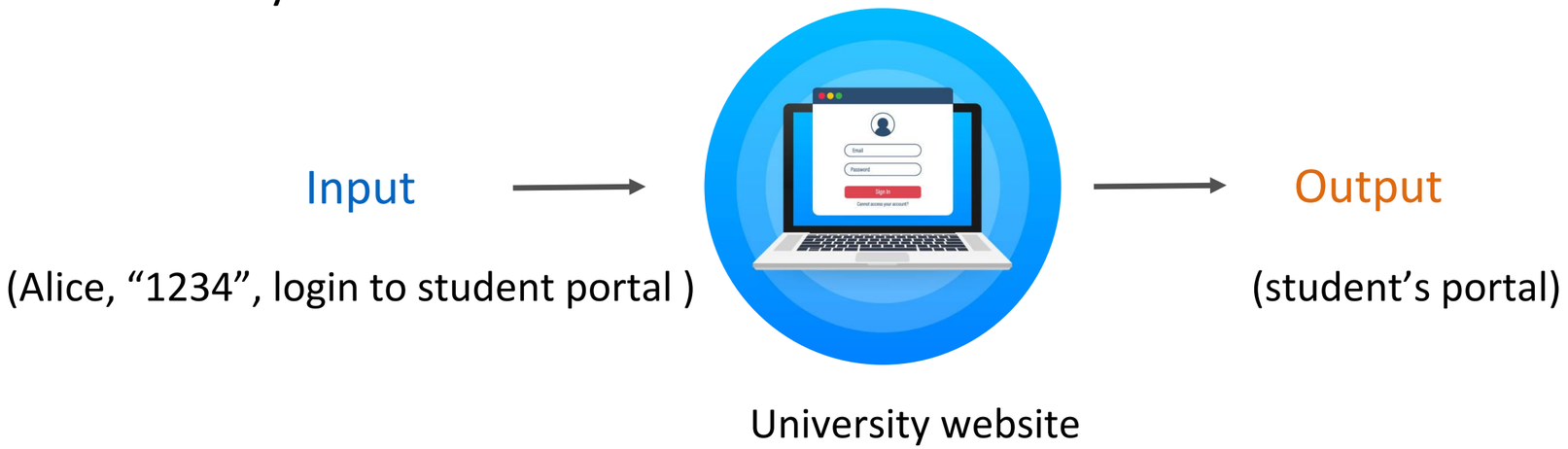
In 2016, hackers stole $81 million from the central bank of Bangladesh

In 2021, hackers were able to gain access to customer data of the e-commerce platform Shopify

Security Testing

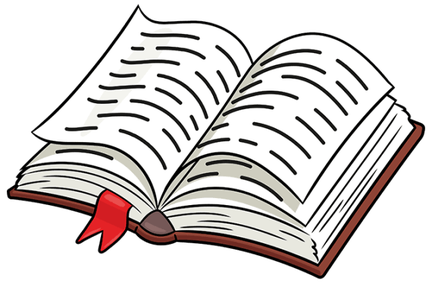
- **Security testing** aims at verifying that the software meets its **security properties**
  - E.g., on a university website, a student should not be able to change her grade
- Software testing consists of:
  - providing inputs to the software
  - verifying that the software outputs are valid according to specifications

- Example: on a university website



Security Testing - Oracle

- **An oracle** is a procedure to determine, for a set of inputs to the software, the correctness of the corresponding outputs, according to specifications.



Input

(Alice, “1234”, login student’s portal)

(Admin, “2023”, login to admin’s portal)

(Nazanin, “007”, login to admin’s portal)

Oracle

Output

(Student’s portal)

(Admin’s portal)

(Error page)

Security Testing - Challenge

- **Oracle problem:** the infeasibility of defining a procedure to automatically determine the correctness of any output produced by the software under test

- Existing automated security testing approaches do not address or partially address the oracle problem
  - Target security issues that are easy to detect (e.g., when the system crashes)
  - Most approaches focus on a particular security issue (e.g., SQL injection)

- **Goal**: address the oracle problem in security testing for a large number of security issues

uOttawa

Metamorphic Testing

- **Metamorphic Testing (MT)** is based on the idea that

  - it may be simpler to reason about **relations between outputs** of multiple test executions,

  called **Metamorphic Relations** (MRs), **than** to specify the output of the system for a given input

- **Example**: on a university website, to test if students can access the student's portal:

  - Input (initial input): select a student who accesses the student's portal

    - (Alice, "1234", login to student's portal)

  - Follow-up input: select a different student who performs the same action

    - (Bob, "0317", login to student's portal)

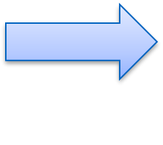
  - MR:

```
Equal(
        Output( Alice, "1234", login to student's portal )
      , Output( Bob, "0317", login to student's portal )
      )
```

MST-wi: Metamorphic Security Testing for Web Interactions

- **MST-wi** method uses **Metamorphic testing** to test the **security properties** of a Web system
- In MST-wi, the system's security properties are captured as MRs

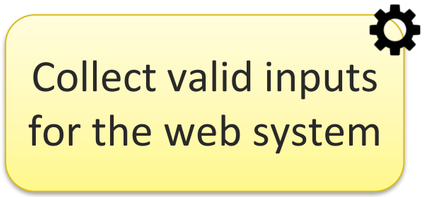
Web System



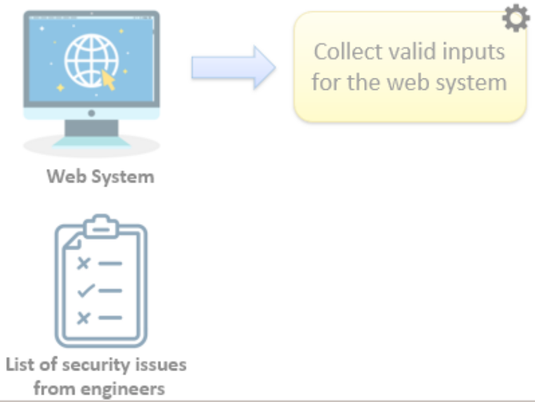
Collect valid inputs for the web system

List of security issues from engineers

MST-wi – Data collector



- We rely on an extension of the Crawljax.
- Crawljax explores the user interface of a Web system and it generates a graph whose
  - nodes: the system states reached through the user interface
  - edges: the action performed to reach a given state
- Crawljax detects the system states based on the content of the displayed page.
- Our extension relies on the edit distance to distinguish the system states
- Crawling stops when no more states are encountered, or when a timeout is reached



MST-wi

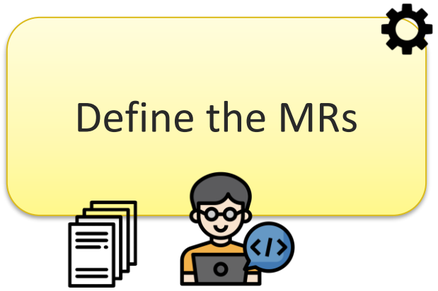
Web System

Collect valid inputs for the web system

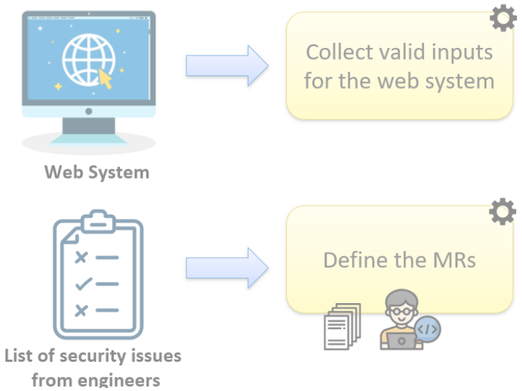
List of security issues from engineers

Define the MRs

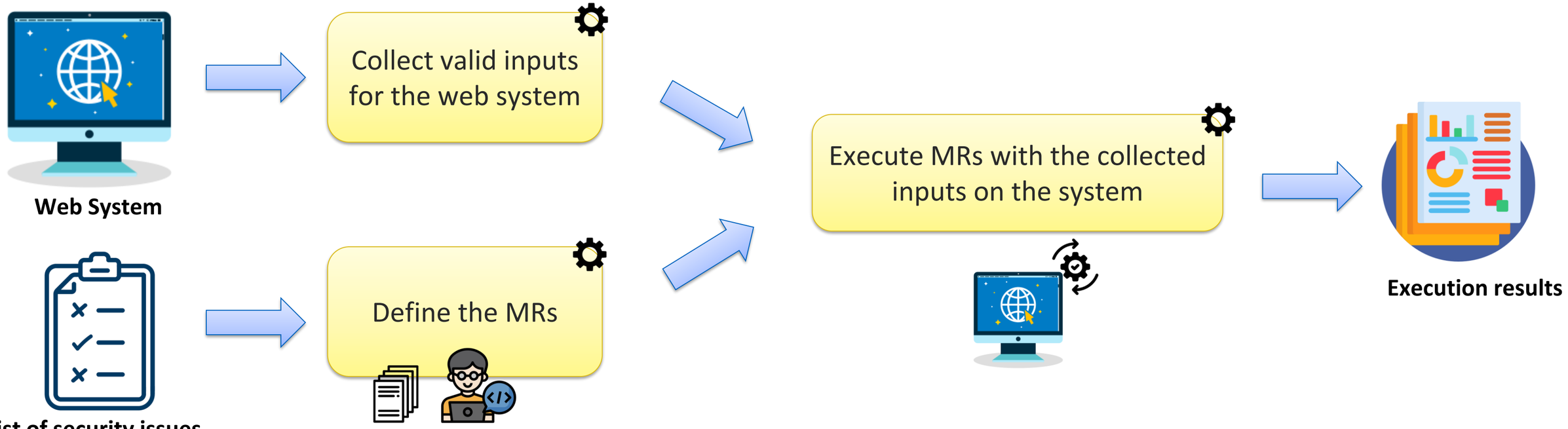
MST-wi – DSL



- SMRL is an extension of Xbase, an expression language provided by Xtext.
- Xbase specifications can be translated to Java programs and compiled into executable Java bytecode.
- SMRL extends Xbase by introducing:
  - Data representation functions
  - Boolean operators to specify security properties
  - Web-utility functions to express data properties and transform data
- We can extend these functions by defining new Java APIs invoked in MRs.



MST-wi


Web System
Collect valid inputs for the web system
List of security issues from engineers
Define the MRs
Execute MRs with the collected inputs on the system
Execution results

MST-wi – MR Example

- **Security issue**: An unauthorized user should not be able to access the admin's information.
- MR:
  - Select the initial input:
    - *A user who can access the admin's portal: (Admin,"2023")*
    - *Set the action to login to the admin's portal*
    - *(Admin,"2023", log in to admin's portal)*
  - Generate the Follow-up input:
    - *Modify the user to someone who should not be able to access the admin's portal:*

      (Billie,"321", login to admin's portal)
  - Define the relation:

```
Not Equal (
        Output(Admin,"2023", login to admin's portal)
      , Output(Billie,"321", login to admin's portal))
```

uOttawa

MST-wi – MR Example

- **Security issue**: A user should not be able to reuse old session IDs for authorization.
- MR:
  - Select the initial input:
    - *A user who performs a logout action: (user1,"u01", logout)*
  - Generate the Follow-up input:
    - *Check for an action that user1 performed after login: action_y*
    - *Copy all information of user1 and insert logout action before action_y:*

    CREATE(user2, copyActionTo(user1, logout, action_y))
  - Define the relation:

```
Not Equal (
        Output(session(user2), logout)
      , Output(session(user2), action_y))
```

MST-wi - Evaluation

- Our empirical results with two open-source web systems, Jenkins and Joomla, show that the approach requires limited manual effort and detects 85.71% of the targeted security issues

- MST-wi can detect 102 different types of security issues,
  - 45% of the security issues reported in MITRE CWE database

- MST-wi can automate 39% of the security testing activities, reported in OWASP top10, not currently targeted by state-of-the-art techniques

MST-wi vs Static and Dynamic Application Security Testing tools

- Static Application Security Testing (SAST) tools: SonarQube and SA2
- Dynamic Application Security Testing (DAST) tools: OWASP Zap and DA2
- The set of weaknesses targeted by MST-wi is larger than what can be targeted by applying all four competing approaches together.

Table1. Comparison of MST-wi with state-of-the-art approaches for the verification of Weaknesses in the CWE Security Design Principles

	MST-wi	OWASP Zap	DA2	SonarQube	SA2
Addressed Weaknesses	101 (45%)	17(7%)	17(7%)	16(7%)	67(30%)

84 (37%)

uOttawa

Vulnerabilities that can be detected by MST-wi

- The feature under test is accessible via a URL/path
- The testing framework supports modifying parameter values
- It is possible to log-in with a predefined list of credentials
- System settings or configuration elements can be controlled by the test engineer
- The testing framework can control the Web-browser (e.g., click on back button)
- The type of the parameters of the request (in URL or post-data) is known or can be easily determined
- It is possible to access system artifacts (e.g., log files)
- The system under test provides a feature to configure the system time
- The testing framework supports handling multiple user sessions in parallel
- The testing framework has a feature to select certificates

Vulnerabilities that cannot be detected by MST-wi

- The weakness concerns a system that is not Web-based or mobile-based.
- The weakness can be discovered only by means of program analysis.
- It is not possible to distinguish valid and invalid behaviour based on system output; a human needs to inspect it.
- The weakness can be discovered only by controlling a third-party component.

uOttawa

Achievements

- MST-wi is the only approach that:

  - **includes** a **Language** to **express** MRs

  - **targets** a **wide** range of **security issues**

  - **includes** a data collection **framework** to **derive** input data **automatically**

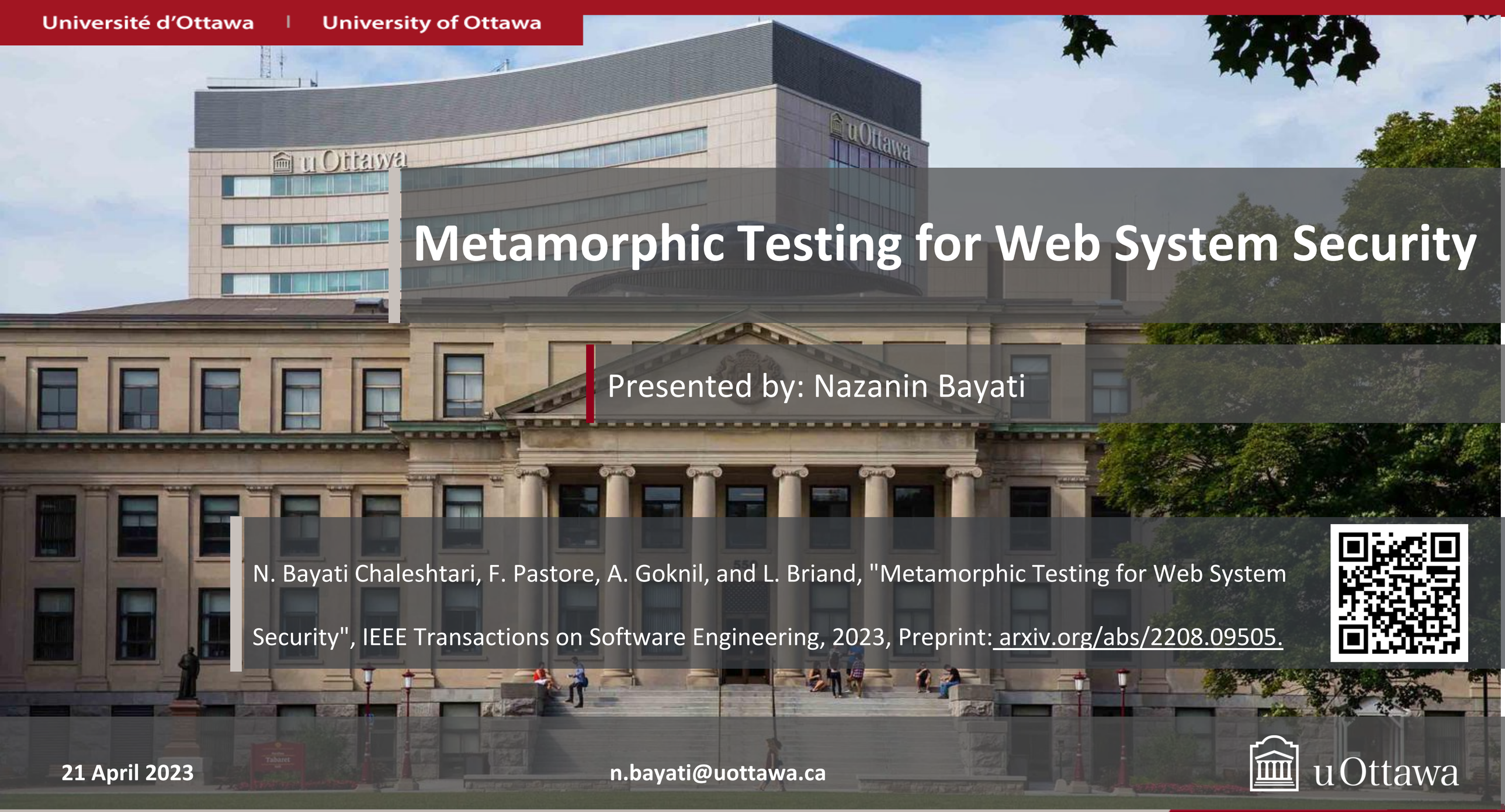
  - **automates** the **execution** of MRs in the Web system

  - **automatically detects** various security issues based on MRs

uOttawa

Future Direction

- It is time-consuming to execute all inputs on our MRs.
- Currently, we are working on reducing the execution time of MST-wi by selecting the most relevant inputs.
- Solution based on AI: clustering and meta-heuristic search



Metamorphic Testing for Web System Security

Presented by: Nazanin Bayati

N. Bayati Chaleshtari, F. Pastore, A. Goknil, and L. Briand, "Metamorphic Testing for Web System Security", IEEE Transactions on Software Engineering, 2023, Preprint: arxiv.org/abs/2208.09505.

21 April 2023

n.bayati@uottawa.ca

uOttawa