| | |
|---|---|
| NAME | G.DHIVYA |
| REG. NO. | 420121106301 |
| DEPARTMENT | ECE |
| YEAR | III |
| COLLEGE NAME | AKTMCET |
| GROUP | IBM GROUP-5 |
| NM I'D | 4C1092622F22920B98C4BC20CA36D80B |

# SMART WATER MANAGEMENT

# PHASE-4

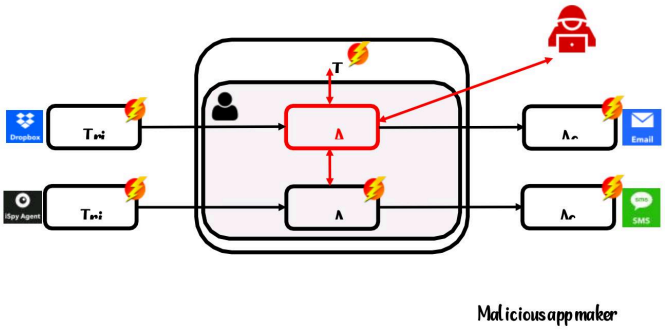
## Innovation

- In this phase you need to put your design into innovation to solve the problem.
- Explain in detail the complete steps that will be taken by you to put your design thatyou thought of in previous phase into transformation.
- Create a document around it and share the same for assessment.

## Module 9: RED SERVICE

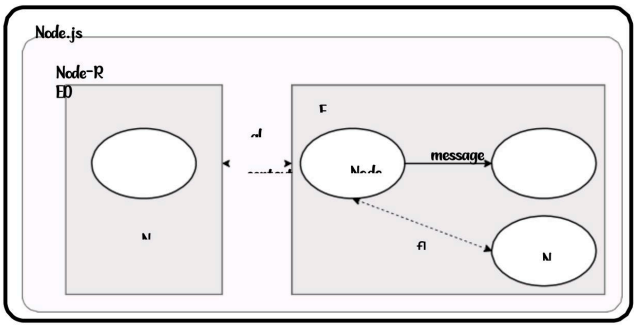
# 1 Introduction

*Trigger-Action Platforms (TAPs)* play a vital role in fulfilling the promise of the Internet of Things (IoT). TAPs empower users by seamlessly connecting other- wise unconnected *trigger* and *action* services.




Malicious app maker

A TAP is effectively a "person-in-the-middle" between trigger and action services. While greatly benefiting from the possibility of apps to run third-party code, TAPs are subject to critical security and privacy concerns.




Motivated by SandTrap, this work is a step toward formally understanding how to monitor Node-RED apps. We present a sound and transparent moni- toring framework for Node-RED for enforcing fine-grained allowlist policies at module-, API-, value-, and context-level. In the following, we discuss Node-RED along with overviewing platform- and app-level vulnerabilities and attacks (Section 2); propose an essential model for Node-RED, suitable to reason about nodes and flows, be they benign, vulnerable, or malicious; and present a mon- itoring framework to express and enforce fine-grained security policies, proving its soundness and transparency (Section 3).

# Node-RED Vulnerabilities

Node-RED is "a programming tool for wiring together hardware devices, APIs and online services", which provides a way of "low-code programming for event- driven applications" [36]. As an open-source platform, Node-RED is mainly tar- geted for deployment as a single-user platform, although it is also available on the IBM Cloud platform [23].

```
module.exports = function(RED){
  function NodeName(config){
    RED.nodes.createNode(this, config); var
    node = this;
    // register a callback when a message is received...
    node.on("input", function(msg){
      ... // functionality of node
      node.send(msg); // or an array of messages for multiple
                      outputs
    });
  }
  RED.nodes.registerType("type-name", NodeName);
}
```

# Node-RED platform

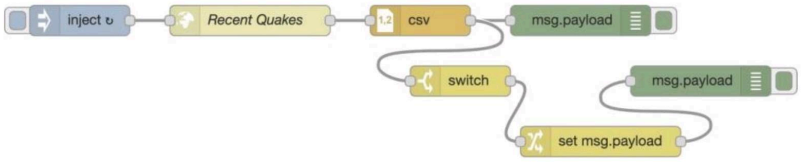
A node is a reactive Node.js application triggered by receiving messages on at most one input port (dubbed *source*) and sending the results of (side-effectful) computations on output ports (dubbed *sinks*), which can be potentially multiple, unlike the input port. Figure 3 illustrates the code structure of a Node-RED node. A special type of node without sources and sinks, called *configuration* node, is used for sharing configuration data, such as login credentials, between multiple nodes.

A flow is a representation of nodes connected together. End users can either create their own flows on the platform's environment or deploy existing flows pro- vided by the official Node-RED catalog [33] and by third parties.

```
[                               // list of nodes
  {                             // node 0
    /* parameters of interest in every node */
    id: NODE0,                  // unique ID of node, string
    type: function              // type of node, string
    wires: [                    // array of array of strings
      [ NODE1 ],                // first output port to node 1
      [ NODE2, NODE3 ] // second output port to nodes 2 and 3
    ],
    ...                         // other parameters
  },
```

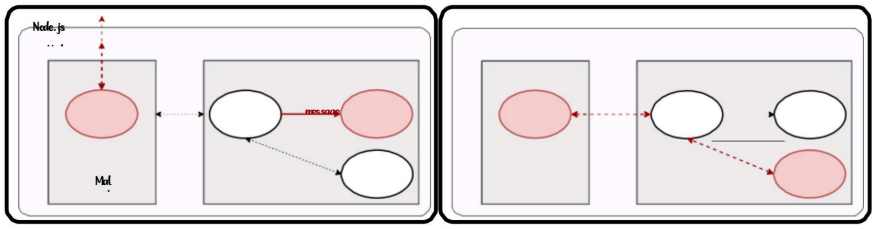
```
    ...                                 // other nodes
]
```

Figure 4: Node-RED flow structure.




In Node-RED, *contexts* provide a shared communicationchannel between different nodes without using the explicit messages that pass through a flow [40]. Therefore the node wiring visible in the user interface reflects only a part of the information flows that are possible in the flow. It introduces an implicit channel that is not visible to the user via the graphical interface of a flow. Node-RED defines three scope levels for the contexts

(a) (b)




The provided policies can later be vetted by the platform and the user, before deploying the node. SandTrap [3] offers a pol- icy generation mechanism to aid developers in designing the policies, enabling both baseline and advanced policies customized by developers or users to express fine-grained app-specific security goals.

In the following, we discuss Node-RED attacks and vulnerabilities that mo- tivate enriching the policy mechanism with different granularity levels. These policies will further be formalized in Section 3.

# Platform-level isolation vulnerabilities

All APIs provided by the underlying runtimes, Node-REDand Node. js, are accessible for node developers, as well as the

incom- ing messages within a flow. As shown in Figure 6a, there are various attack sce- narios for malicious nodes [3]. At the Node.js level, an attacker can create a ma- licious Node-RED node including.

# Formalization

Our runtime framework formalizes the core of the flow-based programming model of Node-RED and was the basis when developing the JavaScript monitor SandTrap [3].

This section presents a security model for Node-RED apps and characterizes the essence of a fine-grained access control monitor for the platform. We show how to formalize and enforce security policies for nodes at the level of APIs and their values, along with the access rights to the shared context. Our main formal results are the soundness and transparency of the monitor.

## Language syntax and semantics

**Syntax** We define a core language to capture the reactive nature of nodes and flows. Nodes are reactive programs triggered by input messages to execute the code of an event handler and potentially produce an output message.

Expression evaluation is standard and records the sequence of events pro- duced during the evaluation, where $M_k$ denotes the ⟨me⟩mory $M$ in $c, M, I, O_k$. Command evaluation models the execution of a node's handler. The handler executes whenever there is a message
in the input channel $I$ by consuming the message and updating the memory accordingly. Assignments operate in a similar manner and record the trace of events produced by variable reads and writes. An assignment updates the memory $M_k$ to $M_k'$

## Security condition and enforcement

We leverage our trace-based semantics to define a semantics-based security con- dition. The condition is parametric on node- level security policies, represented as allowlists of API calls and accesses to the shared context. Then, we present the semantics of a fine-grained node-level monitor and prove its soundness and transparency with respect to the security condition.

# Related work

We discuss the most closely related work on Node-RED security and modeling, monitor implementation, and securing trigger-action platforms in general. We refer the reader to surveys on the security of IoT app platforms [7,13] for furtherdetails.

**Node-RED security and modeling** Ancona et al. [5] investigate runtime monitoring of parametric trace expressions to check the correct usage of API functions in Node-RED. Trace expressions allow for rich policies, including tem- poral patterns over sequences of API calls. By contrast, our monitor supports both coarse and fine access control granularity of modules, functions, andcon- texts. Schreckling et al.

# Conclusion

We have investigated the security of Node-RED, an open-source JavaScript- driven trigger-action platform. We have expanded on the recently-discovered critical exploitable vulnerabilities in Node-RED, where the impact ranges from massive exfiltration of data from unsuspecting users to taking over the entire platform. Motivated by the need for a security mechanism for Node- RED, we have proposed an essential model for Node-RED, suitable to reason about nodes and flows, be they benign, vulnerable, or malicious. We have formalized a prin- cipled framework to enforce fine-grained API control for untrusted Node-RED applications. Our formalization for a core language shows how to soundly and transparently enforce global security properties of Node-RED applications by local access checks, supporting module-, API-, value-, and context-level policies.

*Acknowledgments* This work was partially supported by the Swedish Foundation for Strategic Research (SSF), the Swedish Research Council (VR), and Digital Futures.

# Module 10: MOBILE APPLICATION DEVELOPMENT

---

## Platform

The software development packages needed to develop, deploy, and manage mobile apps aremade from many components and tools which allow a developer to write, test, and deploy applications for one or more target platforms.

### Front-end development tools

Front-end development tools are focused on the user interface and user experience (UI-UX)and provide the following abilities:

- UI design tools
- SDKs to access device features
- Cross-platform accommodations/support

Notable tools are listed below.

**First-Party**

First party tools include official SDKs published by, or on behalf of, the company responsiblefor the design of a particular hardware platform (e.g. Apple, Google, etc) as well as any third-party software that is officially supported for the purpose of developing mobile apps for that hardware.

| Platform | Programming language | Debuggers available |
|---|---|---|
| Android | Java but portions of code can bein C, C++,<br><br>Kotlin | Debugger integrated in Eclipse, standalone debuggingmonitor available |
| BlackBerry | Java | Debugger integrated in IDE |
| iOS SDK | Objective-C, Swift | Debugger integrated in Xcode IDE |
| iOS SDK | Object Pascal | Debugger integrated in Xcode IDE |

Third Party

| Platform | Programming language |
|---|---|
| Accelerator | HTML5, C# |
| MobileTogether | XPath/XQuery, Action Trees visual programming language |
| App Inventor for Android | Visual blocks-based programming language, with Interface designer |
| Appcelerator | JavaScript |
| Basic4android | Visual Basic similar syntax |
| Codename One | Java |
| Solar2D | Lua |
| DragonRAD | Visual drag & drop tiles |
| GeneXus for Mobile andSmart Devices | Knowledge representation and declarative programming-modeling for easydevelopment, then code is automatically generated for each platform |
| IBM MobileFirst Studio | HTML5, CSS3, JavaScript, and native SDK languages w/ Native WorklightAPI |
| Lazarus | Object Pascal |

## Security add-on layers

With bring your own device (BYOD) becoming the norm within more enterprises, IT departments often need stop-gap, tactical solutions that layer atop existing apps, phones, andplatform component. Features include

- App wrapping for security
- Data encryption
- Client actions
- Reporting and statistics

## System software

Many system-level components are needed to have a functioning platform for developingmobile apps.

| Platform | Programming language |
|---|---|
| Adobe AIR | ActionScript, HTML, CSS, JavaScript |
| BREW | C; the APIs are provided in C with a C++ style interface |
| Firefox OS | HTML 5, CSS, JavaScript |
| .NET Compact Framework | C#, VB.NET, Basic4ppc |
| OpenFL | Haxe (similar to Actionscript and Java) |
| Palm OS | C, C++, Pascal |
| Python | Python |
| Symbian | C++ |
| Tizen | *Web-based:* HTML 5, CSS, JavaScript *Native:* C, C++ |
| Ubuntu Touch | *Web-based:* HTML 5, CSS, JavaScript *Native:* QML, C, C++ |

| Platform | Programming language |
|---|---|
| webOS | JavaScript, CSS, HTML, C and C++ through the PDK |
| Windows Mobile | C, C++ |
| Windows Phone | C#, Visual Basic, C, C++ |

Criteria for selecting a development platform usually contains the target mobile platforms, existing infrastructure and development skills. When targeting more than one platform with cross-platform development it is also important to consider the impact of the tool on the user experience. Performance is another important criteria, as research on mobile apps indicates astrong correlation between application performance and user satisfaction.

## Mobile app testing

Mobile applications are first tested within the development environment using emulators andlater subjected to field testing. Emulators provide an inexpensive way to test applications onmobile phones to which developers may not have physical access. The following are examples of tools used for testing application across the most popular mobile operating systems.

- **Google Android Emulator** - an Android emulator that is patched to run on a Windows PCas a standalone app, without having to download and install the complete and complex Android SDK. It can be installed and Android compatible apps can be tested onit.
- **The official Android SDK Emulator** - a mobile device emulator which mimics all of thehardware and software features of a typical mobile device (without the calls).
- **TestiPhone** - a web browser-based simulator for quickly testing iPhone web applications.This tool has been tested and works using Internet Explorer 7, Firefox 2 and Safari 3.
- **iPhoney** - gives a pixel-accurate web browsing environment and it is powered by Safari. Itcan be used while developing web sites for the iPhone. It is not an iPhone simulator but instead is designed for web developers who want to create 320 by 480 (or 480 by 320) websites for use with iPhone. iPhoney will only run on OS X 10.4.7 or later.
- **BlackBerry Simulator** - There are a variety of official BlackBerry simulators available toemulate the functionality of actual BlackBerry products and test how the device software, screen, keyboard and trackwheel will work with application.
- **Windows UI Automation** - To test applications that use the Microsoft UI Automationtechnology, it requires Windows Automation API 3.0. It is pre-installed on Windows 7,Windows Server 2008 R2 and later versions of Windows.
- **MobiOne** Developer - a mobile Web integrated development environment (IDE) for Windows that helps developers to code, test, debug, package and deploy mobile Webapplications to devices such as iPhone, BlackBerry, Android, and the Palm Pre. MobiOne Developer was officially declared End of Life by the end of 2014.[8]

- eggPlant: A GUI-based automated test tool for mobile app across all operating systemsand devices.
- Ranorex: Test automation tools for mobile, web and desktop apps.