# Developing a UI and Automation Framework for a Cybersecurity Research and Experimentation Environment

Cody Butler, George Thompson
*Computer Science*
*Norfolk State University*
Norfolk, VA, USA
[c.butler74291,g.m.thompson74431]
@spartans.nsu.edu

George Hsieh, Mary Ann Hoppa
*Computer Science*
*Norfolk State University*
Norfolk, VA, USA
[ghsieh, mahoppa]@nsu.edu

Kevin S. Nauer
*Computer & Network Security*
*Sandia National Labs*
Albuquerque, NM, USA
ksnauer@sandia.gov

***Abstract*—Cybersecurity is critically important to nations and societies globally. However, it has been very difficult for cybersecurity researchers to analyze, compare and validate the effectiveness of various offensive and defensive mechanisms quantitatively and experimentally. The Cyber Analysis, Simulation and Experimentation Environment (CASE-V) project is focused on developing a flexible, feature-rich and user-friendly cybersecurity research and experimentation environment. The CASE-V includes two foundational platforms: OpenStack cloud and Hadoop big data platforms. A unified User, Application, Management and Orchestration layer on top of these platforms is needed to make the CASE-V user friendly and robust. This paper presents the architectural design and implementation of the key software components for a web-based user interface and a framework for automating the various configuration, orchestration and management tasks. This design is based on the MEAN (MongoDB, Express, AngularJS, and Node) full-stack JavaScript framework for developing responsive web applications. Bootstrap is also used for developing with HTML, CSS, and JavaScript. In addition, SPICE is used for remote access to virtual machines in a seamless manner. For automation, Python and Bash shell scripts are implemented to execute the relevant OpenStack API's.**

***Keywords—Cybersecurity Research and Experimentation Environment, OpenStack, MEAN Stack, Bootstrap***

## I. INTRODUCTION

Cybersecurity is critically important to nations and societies globally. In 2016 cybercrime was estimated to cost the global economy over $450 billion [1]. In a report published in 2017 [2], the cost of cybercrime was accelerating (with a 23 percent increase over the previous year) and was costing organizations, on average, $11.7 million that year.

To address this situation, cybersecurity research and practices require sound science that builds upon controlled, well-executed experiments, according to the 2015 NSF report titled *Cybersecurity Experimentation of the Future (CEF): Catalyzing a New Generation of Experimental Cybersecurity Research* [3]. Research infrastructure is a critical need of the cybersecurity community, especially to support new revolutionary approaches to experimentation and test.

To help address this critical need in cybersecurity research and experimentation infrastructures, a research project was initiated at Norfolk State University in 2015 to develop a cloud enabled and big data capable Cyber Analysis, Simulation and Experimentation Environment (CASE-V) for enhancing situational awareness and decision support, especially for countering new advanced persistent threat.

The CASE-V, as shown in Fig. 1, consists of three major architectural layers: applications, platforms, and enabling technologies which include cloud computing and big data analytics.

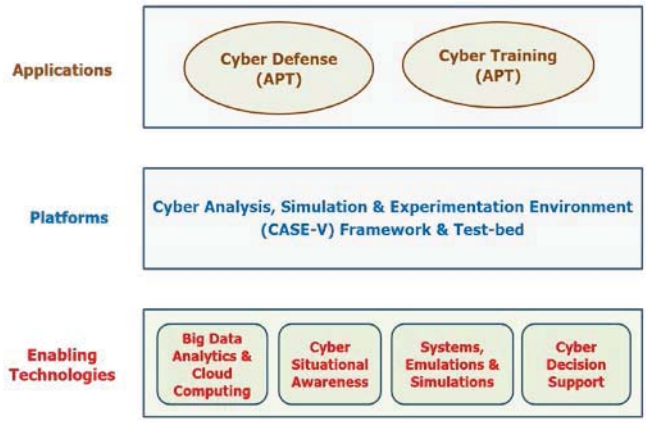


Fig. 1: CASE-V architecture

Cloud computing or virtualization infrastructures are needed for investigating large-scale networks and endpoints to model the real world. Big data analytics are also needed for situational awareness and decision support in cybersecurity research and experimentation, especially in intrusion detection, anomaly detection, incident response, and cyber intelligence all of which require dealing with big data and can benefit significantly from automated tools, machine learning and data analytic techniques.

The CASE-V testbed being implemented at NSU uses **`OpenStack`** [4] for the cloud platform, and **`Hadoop`** [5] for the big data analytics platform, respectively. There are multiple OpenStack and Hadoop platforms configured as self-contained systems with their own compute, storage, and networking hardware and resources. They are also interconnected through a high-performance backbone network, a client access network, a remote diagnostic and access network, and a shared storage network and disk arrays.

To provide a flexible, feature-rich and user-friendly cybersecurity research and experimentation environment, a

This work was supported in part by DoD (award #FA8750-15-2-0120), and DoE/NNSA (award #DE-NAOO02686).

unified User, Application, Management and Orchestration layer on top of the foundational platforms is needed to make the CASE-V user friendly and robust.

This paper presents the architectural design and implementation of the key software components for a web-based user interface and a framework for automating the various configuration, orchestration and management tasks. This design is based on the MEAN (MongoDB, Express, AngularJS, and Node) full-stack JavaScript framework for developing responsive web applications. Bootstrap is also used for developing with HTML, CSS, and JavaScript. In addition, SPICE is used for remote access to virtual machines in a seamless manner. For automation, Python and Bash shell scripts are used to execute the relevant OpenStack API's.

The paper also describes the Version 1 prototype implementation of the UI and automation framework. This initial implementation focused on the OpenStack cloud computing platform, and provided a collection of capabilities including multiple user privilege levels, admin capabilities, user registration and token-based authentication, verification of new users, cloud space request for new users, resource limits, etc.

The remainder of this paper is organized as follows. Section II presents the key technologies chosen for the web-based user interface framework for the CASE-V testbed. Section III discusses the feasibility study and early prototyping efforts performed during the architectural investigation phase. Section IV describes the Version 1 prototype implementation of the UI and automation framework. Section V provides illustration of a small sampling of the components and capabilities implemented in the Version 1 prototype. Section VI concludes the paper with a summary and discussion of future work.

## II. Web-Based User Interface Framework

### A. Goals

The following goals were set for the design and implementation of the UI and automation framework for the CASE-V testbed:

1) User-friendly, extensible, robust, secure, scalable.
2) Support different platforms, programming languages and API's.
3) Responsive web interfaces and dynamic information updates.
4) Use best-in-class technologies and architectures.

### B. MEAN Stack

To help achieve the goals listed above, the MEAN stack is chosen as the underlying technology for the web-based user interface framework.

**MEAN** is a free and open-source JavaScript software stack used for creating dynamic websites and web applications [6]. It is composed of four components:

1) **MongoDB**: a NoSQL database using JSON-like documents with schemas.
2) **Express.js**: a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.
3) **AngularJS**: extends the HTML vocabulary for declaring dynamic views in web applications, and provides an environment that are very expressive, readable, and quick to develop.
4) **Node.js**: cross-platform JavaScript run-time environment that executes JavaScript code server-side using an event-driven, non-blocking I/O model that makes it lightweight and efficient.

Although each MEAN component can be deployed and used individually without requiring the others, the MEAN stack when used together offers one major advantage. Because all components of the MEAN stack support programs written in JavaScript, MEAN applications can be written in one language for both server-side and client-side execution environments.

### C. Bootstrap

In addition to the MEAN stack, Bootstrap is chosen as the front-end component library for the CASE-V testbed.

**Bootstrap** [7] is a free and open-source front-end framework for faster and easier web development. It was developed by Twitter for building responsive and mobile-first projects on the web. The Bootstrap toolkit provides HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as JavaScript extensions.

## III. Feasibility Study and Early Prototyping

The initial research on the MEAN stack and Bootstrap identified many benefits of using this powerful combination. On the other hand, the project team did not have experiences in MEAN and Bootstrap. Thus, additional efforts were devoted by the project team on feasibility study and early prototyping to determine the architectural approaches for the following two aspects of integrating this web development and applications environment with the OpenStack platform:

1) Remote console access to VMs running on OpenStack.
2) Scripting language support for task automation.

### A. Remote Console Access

OpenStack allows users to remotely access VM's through remote connection protocols such as Virtual Network Computing (VNC) and SPICE.

**VNC** is a graphical desktop sharing system that uses the Remote Frame Buffer protocol (RFB) to remotely control another computer. The VNC protocol [8] is limited, lacking support for multiple monitors, bi-directional audio, reliable cut-and-paste, video streaming and more. **SPICE** [9] is a new protocol that aims to address the limitations in VNC and provide good remote desktop support.

The first feasibility study and early prototyping effort was focused on determining if both protocols were supported by the MEAN/Bootstrap framework and which one was more efficient.

### B. Remote Console Access using VNC Protocol

The VNC protocol was first tested to evaluate its efficiency using VMware Workstation, because this hypervisor supports the VNC protocol. A VM was created and launched using VMware Workstation on a Windows 7 host

machine. The only configuration step needed to provide VNC support for the VM was to specify the port number on the Windows host, which served as a VNC server, to be used for the VNC connection to this VM.

Next, a VNC client called Vinagre was installed on a separate host running Ubuntu 16.04 OS. The two hosts were connected to the same LAN switch and assigned to the same subnet, as shown in Fig. 2. This test demonstrated that VNC connections were successfully established and the response time performance was satisfactory.

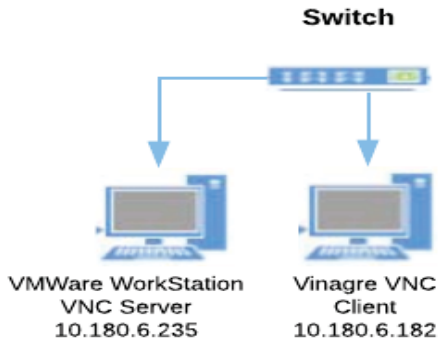


Fig. 2: VNC testing configuration

The next step involved finding a web-based VNC client that could be integrated into the MEAN/Bootstrap framework. However, this search was not very successful for a variety of reasons including the dependencies being out of date or packages containing bugs, etc. One attempt, using a bower package called angular-novnc to create a VNC client that worked with the MEAN stack, seemed to work. However, the resulting display appeared to have trouble rendering the pixels of the VM's GUI. Thus, a decision was made to start investigating the SPICE option and determine its efficiency.

### C. *Remote Console Access using SPICE Protocol*

To test the SPICE protocol, VMware Workstation hypervisor could not be used, since it did not support SPICE to connect to its guest VM's.

On the other hand, **`QEMU`** hypervisor supported the SPICE protocol. In addition, Vinagre could also be used as a SPICE client. Connection to a VM running in QEMU from another machine running Vinagre was successful in a setup like the one shown in Fig. 2, except this time with the use of SPICE as the protocol and QEMU as the hypervisor.

The next step in the feasibility study was to find a suitable web-based SPICE client instead of Vinagre, so it could be incorporated into the MEAN/Bootstrap framework. SPICE offered an open-source web-based client [10]. To use the SPICE web-based client to remotely connect to a QEMU VM, a proxy script was needed in between the SPICE server (QEMU) and the SPICE client. This was because the SPICE server used a **`WebSocket`** protocol that the SPICE client could not interpret in the web browser.

An open-source Python script called **`Websockify`** was used as a WebSocket to TCP proxy/bridge. The script was configured so that it would listen on the port that the SPICE server is running and forward the output to a second port that was specified. The SPICE client listened on the second specified port. Through this setup, as shown in Fig. 3, remote connection to a VM in QEMU was achieved in a web browser.

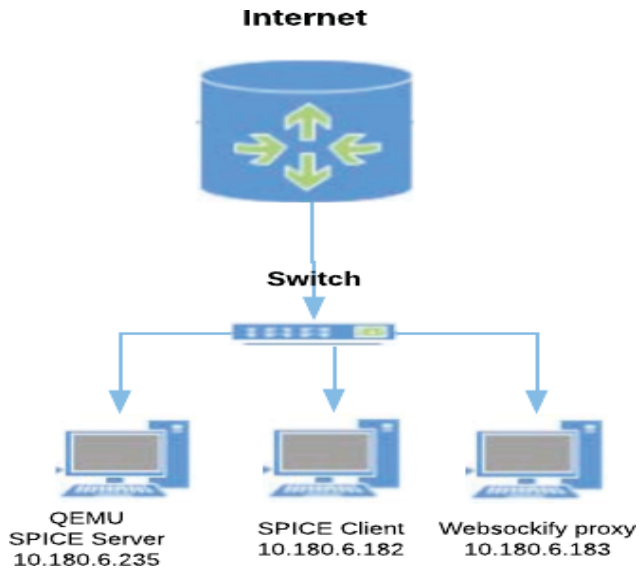


Fig. 3: SPICE testing configuration

### D. *Scripting Language Support for Task Automation*

Python is the preferred scripting language for writing task automation code for the CASE-V testbed. Thus, it was very important to investigate how Python code could be executed within the MEAN/Bootstrap framework.

To execute Python code, the Node package named **`python-shell`** was installed [11]. The python-shell is a simple way to run Python scripts from Node.js with basic but efficient inter-process communication and better error handling. It reliably spawns Python scripts in a child process running on the Node web-server.

A sample code to run a Python script using the python-shell is shown below:

```
var PythonShell = require('python-shell');

PythonShell.run('my_script.py', function (err) {
  if (err) throw err;
  console.log('finished');
});
```

## IV. Version 1 Prototype Implementation

The CASE-V project team proceeded to implement the Version 1 Prototype, after the architectural design was completed and its technical feasibility established.

The Version 1 prototype development involved the setup of OpenStack, MEAN stack, and Bootstrap, and implementing a collection of UI and task automation capabilities.

### A. *System Configuration*

To expedite learning, experimentation and development, the Version 1 prototype was implemented on a very small-scale system, as shown in Fig. 4 below.

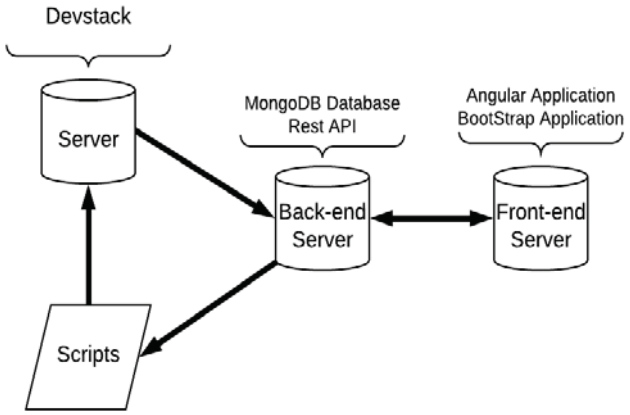


Fig. 4: Version 1 prototype system configuration

Two Dell Precision 7910 workstations were configured for the Version 1 prototype. Each workstation was equipped with 12 Intel Xeon 64-bit Cores, 128 GB of RAM, and 1 TB hard disk storage. The two workstations were connected to the same LAN switch and assigned to the same subnet.

Both workstations were loaded with Ubuntu 16.04 Desktop release. The first workstation was configured to run DevStack, which was a system for quickly installing an OpenStack cloud from upstream git for testing and development, the automation scripts, and the back-end components of the MEAN/Bootstrap framework: MongoDB and REST API.

The second workstation was configured to run the front-end components: AugularJS, Bootstrap, and Node to manage the information displayed to the user.

### B. DevStack

**DevStack** is a series of extensible scripts used to quickly build a complete OpenStack environment based on the latest versions of the software from git master [12]. For all intents and purposes, a DevStack deployment of an OpenStack cloud has the same capabilities as a full-scale deployment, including the ability to launch VM's and configure internal networks.

With a DevStack deployment, all cloud services run on a single node which limits its capacity and scalability. DevStack is primarily used for development and operational testing. It is much easier to install, configure, operate, troubleshoot, and reinstall a DevStack implementation than a multi-node OpenStack implementation.

### C. MEAN/Bootstrap

To facilitate early-stage learning and experimentation, all MEAN/Bootstrap components were installed and configured to run on a single host. Later, the MEAN stack components were installed and run on two separate physical hosts in the Version 1 prototype system to provide better scalability.

MongoDB with the REST API were installed on a back-end server, while AngularJS with Bootstrap application were installed on a front-end server. These servers needed to be able to communicate with each other, so they could send request/response information when needed. The back-end server could invoke Python scripts to call OpenStack's API to accomplish operational tasks.

MongoDB, Node.js and the Node package manager (**npm**), Express, and Angular were installed along with additional packages like Yeoman and Bower. **Yeoman** and **Bower** facilitate the installation and management of web applications, components, and projects. In addition, a Node package named **Satellizer** was installed and used for AngularJS authentication. This package provided the ability to create users within the MEAN stack to have unique tokens for identification which would allow different privileges to be assigned to different users.

### D. DevStack VM Remote Console Access

After DevStack was successfully installed and configured on the first workstation, an effort was launched to test remote console access to VM's created by OpenStack/DevStack. It turned out that OpenStack did provide its own SPICE client as well as a proxy [13].

To use these features, the Ubuntu packages **SPICE-html5** and **openstack-nova-SPICEhtml5proxy** were installed first. Since OpenStack uses VNC as its default remote console access protocol, some configuration files had to be changed to disable VNC and enable SPICE. Once that was done, an VM instance on the DevStack cloud was launched to test how to remotely access the VM using SPICE.

The way DevStack supported remote console access was different from other hypervisors like QEMU and VMware Workstation. Those hypervisors required specifying a separate port for each VM to provide remote console access to that VM. With DevStack, only one port was need for all VM's running in the same cloud. The nova-SPICEhtml5proxy would listen on that port for any incoming SPICE client request. The proxy would identify which VM the SPICE client was trying to connect by the VM's token. If this token was not specified, the proxy would not allow access to that VM.

The SPICE client could be embedded into the front-end of the MEAN/Bootstrap framework using an Iframe tag which allowed embedding another HTML page into the current page. Python scripts were developed to use OpenStack **Nova** commands to find the token for the VM dynamically. This approach successfully provided the capability to remotely access a VM running on DevStack from a client running within the MEAN framework on another host, as shown in Fig. 4 above.

### E. Version 1 Prototype Software Layers

Conceptually the Version 1 prototype software was organized into 4 layers, as shown in Fig. 5 below.

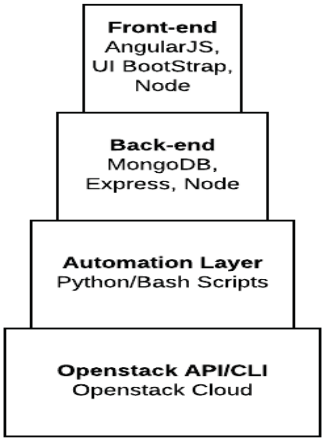


Fig. 5: Version 1 prototype software layers

The bottom layer consisted of the OpenStack cloud environment. The next layer up, the automation layer, consists of the Python and Bash scripts that made calls to OpenStack's API and CLI. This was how cloud operations were automated through these scripts.

The third layer, the back-end layer, consisted of MongoDB and Express which was installed as a Node package. Users and their information were stored in a MongoDB database. Express was used to set up API endpoints for handling HTTP requests. The Python scripts in the automation layer were called and executed from the back-end server using the python-shell package which served as a bridge between the back-end layer and the automation layer.

The topmost layer, the front-end layer, consisted of AngularJS, UI-Bootstrap and Node. The front-end layer was responsible for managing the user interface. It would also make HTTP requests to the back-end server, which would then call the Python scripts to perform operational commands in the OpenStack cloud. The results and responses would be sent from the cloud to the scripts, which would then return them to the back-end server, and consequently displayed it on the front-end user interface.

## V. Sample Components and Capabilities

### A. User Interface

Fig. 6, at the end of the paper, shows a sample screen implemented with the MEAN/Bootstrap framework. It also provides an illustration of the remote console access via this UI to a VM instance running on DevStack.

### B. User Registration

Fig. 7 shows a sample screen for a user to register with the CASE-V testbed, request a certain level of privilege, and cloud resource allowance. A user registration request needed to be reviewed and approved by an administrator who could adjust the privilege level and resource quotas granted to this user.

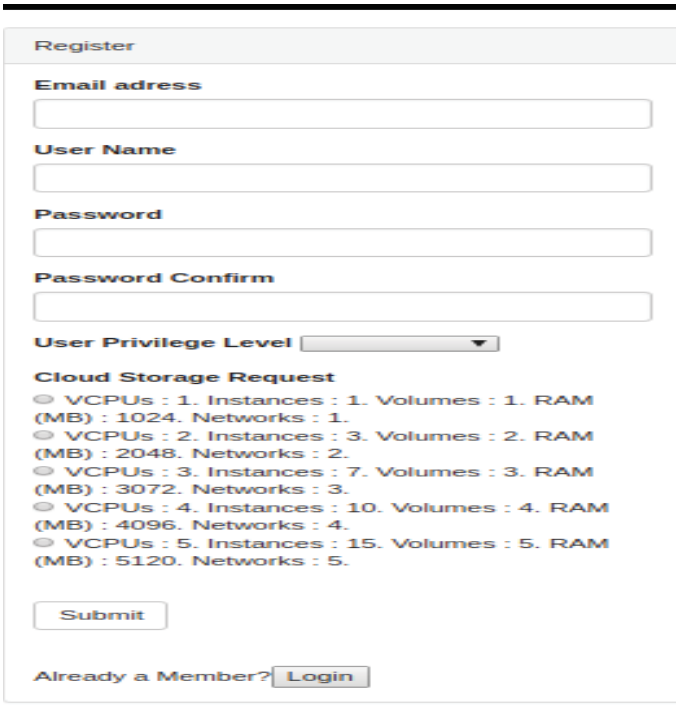


Fig. 7: User registration screen

Fig. 8 illustrates the steps and components involved in registering and creating a new user for the CASE-V testbed.

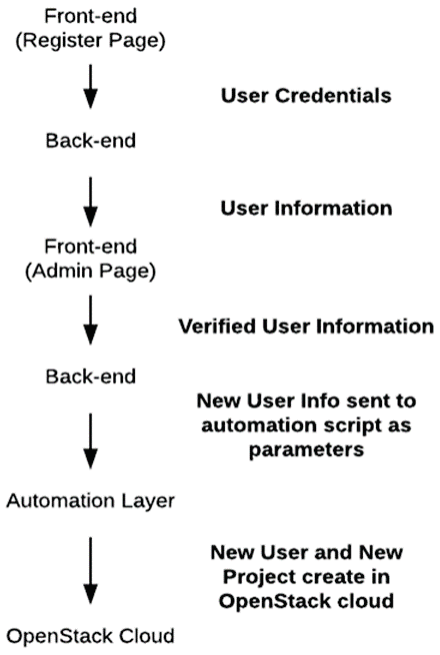


Fig. 8: Register and create a new user

## VI. Summary

This paper presented the architectural design and implementation of the key software components for a web-based user interface and a framework for automating the various configuration, orchestration and management tasks.

This design is based on the MEAN (MongoDB, Express, AngularJS, and Node) full-stack JavaScript framework for developing responsive web applications. Bootstrap is also used for developing with HTML, CSS, and JavaScript. In addition, SPICE is used for remote access to virtual machines in a seamless manner. For automation, Python and Bash shell scripts are implemented to execute the relevant OpenStack API's.

It also described the Version 1 prototype implementation of the UI and automation framework. This initial implementation focused on the OpenStack cloud computing platform, and provided a collection of capabilities including multiple user privilege levels, admin capabilities, user registration and token-based authentication, verification of new users, cloud space request for new users, resource limits, etc.

The Version 1 prototype is functional, albeit its functionality is very limited at this moment. Nonetheless, its design and development established a solid foundation for the CASE-V testbed upon which the project team plans to continue developing new features, capabilities, enhancements, and expanded capacity.

The first major area of future work is to incorporate the user interface and automation framework on a large-scale and enterprise-grade OpenStack cloud system which is a part of the CASE-V testbed.

The second major area of future work is to incorporate the framework on the Hadoop big data systems, which are also part of the CASE-V testbed, to provide a unified user interface and automation framework for both OpenStack and Hadoop systems.

Other areas of enhancement include changing from the token-based authentication scheme to an OAuth based scheme to provide support for single-sign-on, allowing users to upload custom VM images using this UI and automation framework, and automating network configurations (e.g., IP addresses) for basic networks using scripts and this UI and automation framework.

## REFERENCES


[1] L. Graham, *Cybercrime costs the global economy $450 billion: CEO*, CNBC, Feb 7, 2017, https://www.cnbc.com/2017/02/07/cybercrime-costs-the-global-economy-450-billion-ceo.html [accessed April 17, 2018].

[2] Ponemon Institute, *2017 Cost of Cyber Crime Study*, Oct 1, 2017, https://www.ponemon.org/blog/2017-cost-of-cyber-crime-study [accessed April 17, 2018].

[3] D. Balenson, L. Tinnel, and T. Benzel, Cybersecurity Experimentation of the Future (CEF): Catalyzing a New Generation of Experimental Cybersecurity Research, 2015, http://www.cyberexperimentation.org/report/ [accessed April 17, 2018]

[4] OpenStack. https://www.openstack.org/

[5] Hadoop. http://hadoop.apache.org/

[6] S. Holmes, Getting MEAN with Mongo, Express, Angular, and Node, 1st ed. Shelter Island, NY: Manning Publications Co., 2016.

[7] T. Richardson, Q. Stafford-Fraser, K. R. Wood and A. Hopper, "Virtual Network Computing," IEEE Internet Computing, vol. 2, no. 1, p. 33, 1998.

[8] Bootstrap. https://getbootstrap.com/

[9] SPICE. https://www.spice-space.org/

[10] SPICE HTML5 Client. https://www.spice-space.org/spice-html5.html

[11] python-shell - npm. https://www.npmjs.com/package/python-shell

[12] "DevStack," OpenStack, [Online] https://docs.openstack.org/devstack/latest/. [Accessed April 20, 2018].

[13] "Configure Remote Console Access," OpenStack, [Online] https://docs.openstack.org/nova/pike/admin/remote-console-access.html. [Accessed April 20, 2018].


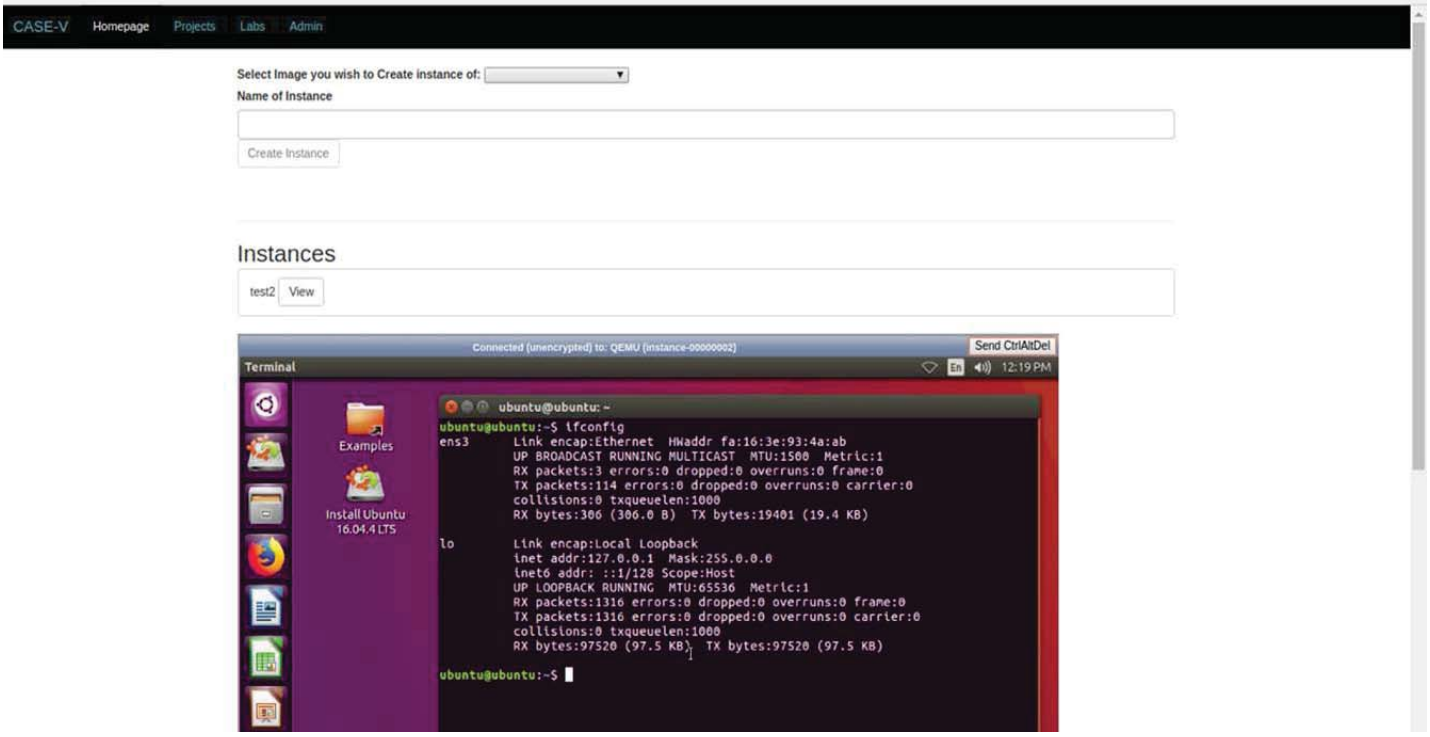


Fig. 6: UI and remote console access