Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

SMESEC

Protecting Small and Medium-sized Enterprises digital technology through an innovative cyber-SECurity framework

D4.2 Final integration report on e-Voting SME pilot

Document Identification			
Status	Final	**Due Date**	31/05/2019
Version	1.0	**Submission Date**	14/06/2019

Related WP	WP4	**Document Reference**	D4.1, D4.3, D4.5, D4.7
Related Deliverable(s)	D2.1, D3.1, D3.2 D3.4, D3.5	**Dissemination Level (*)**	PU
Lead Organization	SCYTL	**Lead Author**	Noemi Folch
Contributors	CITRIX, FORTH, ATOS	**Reviewers**	Manos Athanatos (FORTH)
			Kostas Lampropoulos (UOP)

Keywords:
security, system, design, architecture, integration, WP4, requirements, goals, innovation, use case, e-voting, protection, defence, management.

This document is issued within the frame and for the purpose of the SMESEC project. This project has received funding from the European Union's Horizon2020 Framework Programme H2020-DS-SC7-2016 under Grant Agreement No. 740787 and supported by Swiss State Secretariat for Education, Research and Innovation (SERI) under contract number 17.00067. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the European Commission.

Document Information

List of Contributors	
Name	Partner
Noemi Folch	SCYTL
Jordi Cucurull	SCYTL

Document History			
Version	Date	Change editors	Changes
0.1	11/04/2019	Jose Fran. Ruiz (Atos)	Table of contents template for all use case partners
0.2	15/05/2019	Noemi Folch (Scytl)	Use case information
0.3	20/05/2019	Christos Tselios (CITRIX)	Contribution
0.4	23/05/2019	Alireza Shojaifar (FHNW)	Changing in Figure 7, adding CYSEC in 4.1 and 4.2
0.5	24/05/2019	Manos Athanatos (FORTH)	Quality Assurance process
0.6	27/05/2019	Noemi Folch (Scytl)	Comments review
0.7	29/05/2019	Jordi Cucurull (Scytl)	
0.8	5/6/2019	Manos Athanatos (FORTH)	Added deployment information of the EWIS Honeypot
0.9	6/6/2019	Fady Copty (IBM)	Added information about training of AngelEye
0.10	6/6/2019	Jordi Cucurull (Scytl) Carolina Rueda (Scytl)	Anonymised IPs Additional information to configure hosts to send logs to XL-SIEM
0.11	12/6/2019	Kostas Lampropoulos (UOP) Jose Fran. Ruiz (Atos) Jordi Cucurull (Scytl) Noemi Folch (Scytl)	Last review
1.0	14/06/2019	ATOS	Quality Check + Submission to EC

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable leader	Noemi Folch (SYCTL)	14/06/2019
Technical manager	Christos Tselios (Citrix)	14/06/2019
Quality manager	Rosana Valle Soriano (Atos)	14/06/2019
Project Manager	Jose Fran. Ruíz (Atos)	14/06/2019

Table of Contents

List of Figures

List of Acronyms

Abbreviation / acronym	Description
API	Application Programming Interface
AWS	Amazon Web Services
BCP	Business Continuity Plan
CLI	Command Line Interface
CWE	Common Weakness Enumeration
DB	Database
DMZ	Demilitarized Zone
DNS	Domain Name System
DRP	Disaster Recovery Plan
Dx.y	Deliverable number y belonging to WP x
EC	European Commission
EC2	Elastic Compute Cloud
EWIS	Electrical Wiring Interconnection System
FE	Frontend
FTP	File Transfer Protocol
GDPR	General Data Protection Regulation
GHDB	Google Hacking Database
GSLB	Global Server Load Balance
HA pair	High Availability Pair
HTTP	Hyper Text Transfer Protocol
HTTPS	HTTP Secure
IP	Internet Protocol
IT	Information Technology
JSON	JavaScript Object Notation
MIP	Mobile Internet Protocol
NetBIOS	Network Basic Input/Output System
NTP	Network Time Protocol

Abbreviation / acronym	Description
OWASP	Open Web Application Security Project
PKCS	Public Key Cryptography Standards
RCP	Remote Copy Protocol
REST	Representational State Transfer
ROTI	Report of Test and Inspection
SIEM	Security Information and Event Management
SMB	Server Message Block
SME	Small Medium Enterprise
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SSH	Secure Shell
TelNet	Telecommunication Network
TFTP	Trivial Files Transfer Protocol
VM	Virtual Machine
VP	Vice-President
VPC	Virtual Private Cloud
VPN	Virtual Private Network
WAF	Web Application Firewall
XML	Extensible Mark-up Language

Executive Summary

This deliverable describes the final work done, and reported in M24, of the integration of the SMESEC Framework in the e-Voting pilot. The report is based in the initial version provided at M18 and we build on top of it the following iterations done in the project, both from a technical and awareness point of view. Together with the advancements and updates done in the system we also report the work done in the awareness and training area in order to cover the needs of the employees identified at the beginning of the project.

Additionally, we also report the final analysis and next steps to be done in the project for the work with the SMESEC Framework and how so far it fulfilled the objectives of the use case. We also describe the business development and the impact SMESEC has in this area, as business improvement is a topic for SMESEC as critical as the technical development.

Finally, this document describes in detail the specifics of the e-Voting use case: scenarios, testing, impact of SMESEC in the use case, etc.

In summary, this document describes the current version of the e-Voting pilot. The work described here will be continued in WP5 for further testing, analysis, and improvement using the enhancements done incrementally in SMESEC in the third year and taking advantage of the large testing and feedback provided by the open call.

1 Introduction

1.1 Purpose of the document

This is the second deliverable of WP4 "Integration of SMESEC security framework to e-Voting, Smart City, Industrial Services and Smart Grids pilots" related to e-Voting pilot. The role of this WP in the SMESEC project is to adapt the SMESEC security framework prototype to the different pilots proposed in the project.

Specifically, D4.2 provides an in-depth description of the integration of SMESEC in the e-Voting use case, the impact in the use case and organization (also from an organization point of view), the cybersecurity training and awareness performed in the scope of the project, fulfilment of objectives as described in the first year and next steps, which will be followed in WP5.

1.2 Relation to other project work

As described before, this document covers the advanced efforts carried out to integrate the SMESEC security framework into the e-Voting pilot. The work described here will be used for other deliverables and Work Packages such as:

- D5.1 testing of the scenarios for validation
- D5.2: specification of the integrated products and services in the four use cases
- D5.3: execution of trials in the pilots
- WP6: the results of this deliverable will be used for exploitation and dissemination activities

1.3 Structure of the document

This document is structured in 6 major chapters

Chapter 1 presents an introduction to the use case, objectives and its integration with SMESEC

Chapter 2 describes review of the requirements and needs identified in the first year

Chapter 3 presents characteristics of the use case: update of the architecture, description of the scenarios used, and impact of SMESEC in the use case from a technical and business point of view

Chapter 4 presents the technical integration of SMESEC in the use case, updated from the last version presented in M18

Chapter 5 describes the cybersecurity awareness and training plan used in the use case

Chapter 6 presents the conclusions at M24 of the integration of the SMESEC platform in the use case

2 Requirements and needs: from planning to action

The requirements and needs identified in the document D2.1 "SMESEC security characteristics description, security and market analysis report", section 2.3.2 "List of requirements for e-Voting Pilot" still reflect the reality and necessities of our organization, and there is no need for update.

3 Scenarios and usability

This chapter presents the characteristics of the use case: update of the architecture, description of the scenarios used, and impact of SMESEC in the use case from a technical and business point of view.

3.1 Updates and enhancement

The updates for the e-Voting use case between M18 and M24 are the following:

- Configuration of the Honeypot at the secure zone: The Honeypot has been deployed and configured in the secure zone as a standalone EC2 instance, which is devoted to detect attacks in the secure zone area.
- Detailed and optimized internal configuration of a standalone Citrix ADC VPX instance in AWS cloud: all the traffic between the voter's device and the voting server passes through this component, which inspects the data ensuring it is correct.
- Definition and setup of application level firewall rules and policy files for traffic categorization in the aforementioned Citrix ADC VPX instance: This allows the inspection of the traffic and rejection of it in case it does not comply with the rules defined.
- Development of the scripts to generate the samples to train Angeleye: These scripts create many different requests for the different voting endpoints and evaluate the response of the voting server, categorizing each request as good or bad depending on the output.
- Generation of most of the dataset of samples to train AngelEye: Using the mentioned scripts we generated millions of samples for training and testing the tool.
- Preliminary setup of AngelEye evaluation client in Apache webserver: an AngelEye tool has been adapted to be deployed in the host of the web server to be run periodically and detect possible attacks from the HTTP POST Requests logged by the server.
- Configuration of XL-SIEM to report on the activity of Apache, Tomcat and Internal EWIS Honeypot servers: an XL-SIEM agent has been deployed within an EC2 instance and the other components configured to send their syslog based logs to the agent. The XL-SIEM agent was also configured with the appropriate plugins for each type of log to be supported.
- Evaluation of the CYSEC tool, giving feedback in terms of usability, functionality and overall user experience: the cloud version of the tool has been used to evaluate the level of security of the company and to receive feedback about it.

3.2 Architecture

The following diagram shows an overview of the online voting solution architecture, representing the main components of the system and their interaction with the different actors:

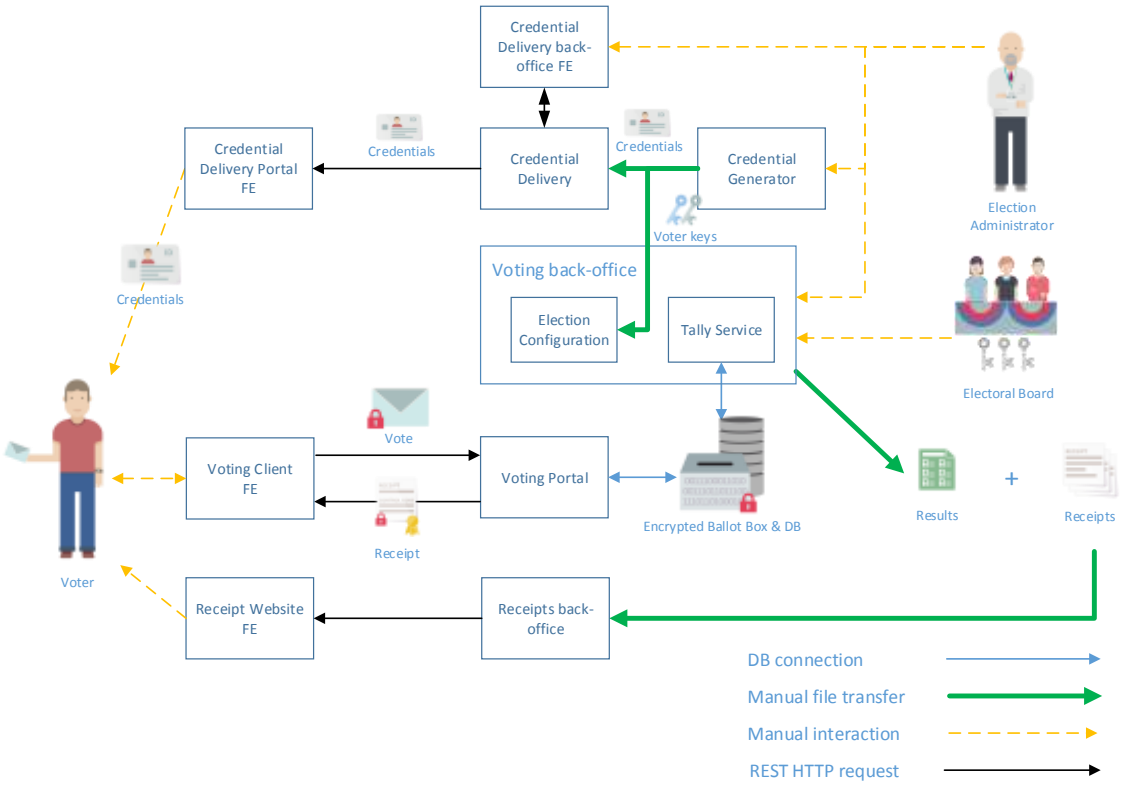

Figure 1: online voting solution architecture

There are two types of modules, those residing on the backend (Credential Generator, Voting back-office, Voting Portal and Receipts back-office) and the ones that are located on the client-side (Credential Delivery Portal, Credential Delivery back-office, Voting Client and Receipt Website). The interaction of these types of modules, hence the communication between the backend and the frontend is performed via HTTP-REST APIs. The remaining interactions are conducted manually or through the database. In this case we also deployed the Credential Delivery components, part of the online voting solution, which will be used during the pilot in order to facilitate the delivery of credentials to the voters. We also included the Receipt Website for allowing the voters to verify their votes at the end of the election.

3.2.1 Credential generation

The credential generator is a command-line tool that creates a pre-defined number of voter credentials and their corresponding voter keys. To generate credentials for an Election, the Credentials Generator requires a series of information, such as the Electoral census, the number of credentials to be generated, the expiration time of the credentials, etc. depending on the election.
The Credentials Generator is automatic. After requesting the parameters or input files and validating the configuration, it will start generating credentials in the output folder configured before.

Figure 2: Credential generation process

The credentials and voter keys are stored in a number of files:

1. List of voter credentials: file that contains the voter credentials to be delivered to the voters. There is one version in plain-text and another one encrypted for the credential delivery (see below).
2. XML with voter authentication and voter keys: files to load the voter identities with protected passwords and voter keys to the voting back-office. This data is used to authenticate the voters and, later, to provide the voters with their voter keys (which are stored within PKCS#12 keystores protected with a derivation of the voter credentials). A PKCS#7 file is also generated to guarantee the authenticity of the credentials and it is used at the time of importing them to the Online Voting System database via the Voting back-office.

This information is directly uploaded to the corresponding modules using the output files. The credentials assignation, authorization, activation and delivery will depend of the project and can vary in function of the project necessities.

3.2.2 Credential delivery

This is a service used to deliver the generated voter credentials to the voters in a secure manner (e.g. via e-mail using a One Time Link (OTS)). The service is implemented through the following components:

- Credential Delivery: it is a web application which offers a REST-API to manage and deliver the credentials.
- Credential Delivery back-office FE: it is a set of Javascript files that allow a manager to configure an election and import the credentials generated by the Credential Generator module. The file with the credentials is encrypted with a secret key shared between the Credential Generator and the Credential Delivery.
- Credential Delivery Portal FE: it is a set of Javascript files that allow a user to retrieve their credentials pointed by the OTS file received.

3.2.3 Voting back-office

The voting back-office is the component of the system that is used to configure, manage and finalize the election. It is a web application that offers a web interface to the administrators (Figure 3) and it is composed of the parts described in the next subsections. A REST-API and a new web interface with partial functionality are also being implemented. This component is only accessible by the system administrators.

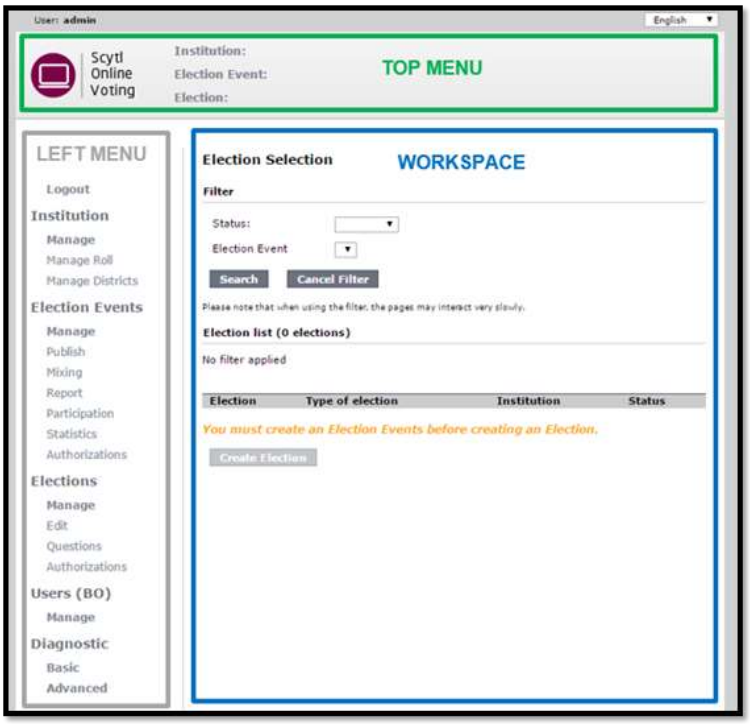

Figure 3: Voting platform back-office

The voting back-office generates immutable logs for all critical operations, i.e. logs that are cryptographically protected against manipulation.

3.2.3.1 Election Configuration

The election configuration part allows the election administrators to configure and manage the election. The following actions should be performed to configure a typical election:

- Generation of institution
- Generation of Election Event
    - Configuration of Election Event
    - Generation of Electoral Board Key (used to encrypt the votes and distributed in shares in smartcards)
    - Generation of Administrator Board key (used to sign the election configuration and distributed in shares in smartcards)
- Configuration of electoral roll (import list of voter credentials)
- Generation of election
- Publish the election

The election configuration service makes use of a database to store and share the data with the Voting Portal and the Tally Service.

3.2.3.2 Tally server

The tally server is the part of the voting back-office that allows the election administrators to finalize the election by verifying, shuffling, decrypting and tallying the votes and publishing the election results (see figure below). The main steps are:

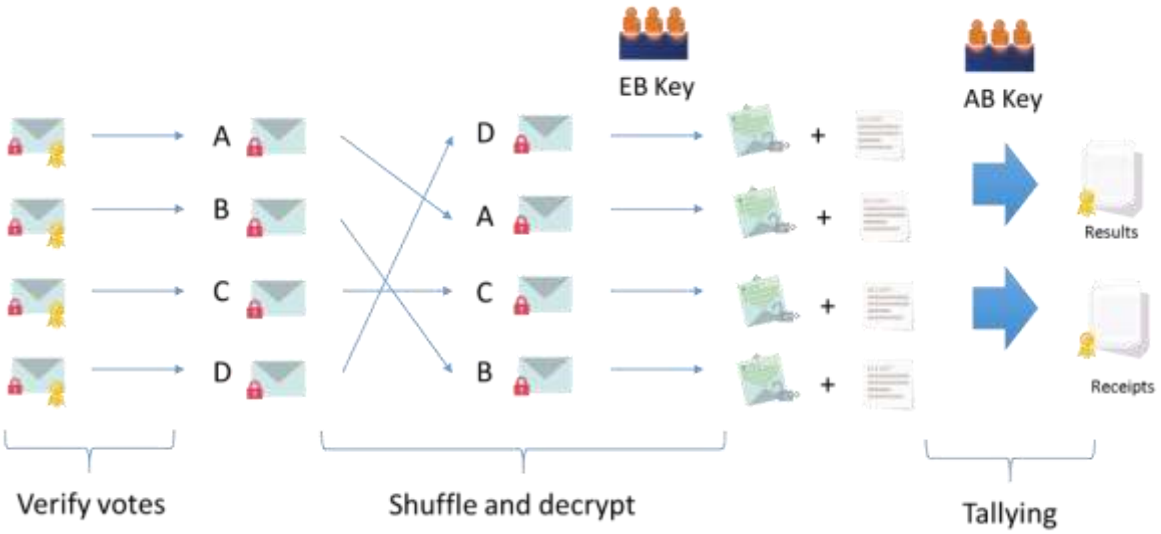

Figure 4: Tally server steps

- Verify votes: the signatures of the votes are verified, i.e. ensuring the integrity and authenticity of each vote, and it is checked that the voters that issued them are present in the electoral roll. Afterwards, the vote signatures are removed to separate the vote contents from the voter identity.
- Shuffle votes: the votes are shuffled to prevent that decrypted votes could be related to voter identities.
- Decrypt votes and receipts: votes and receipts are decrypted. Despite not represented on the picture for simplicity, vote and receipts are separately shuffled again.
- Tally votes: decrypted votes are counted in order to compute the election results.
- Publish results and receipts: results and receipts are published in a website; thus anybody can check them and voters can be sure their votes were processed.

3.2.4 Voting Portal

The voting portal is the component that, together with the Voting Client component, allows voters to cast a vote during the election. It also provides authentication facilities if no third-party authentication service is used. This component is connected to a network accessible to voters (usually Internet).

3.2.5 Voting Portal backend

The component is implemented as a web application and offers a REST-API. The following operations are supported:

- Authentication: Used by the voter to authenticate to demonstrate its identity. A JSON web token and an authentication token are returned in exchange.
- Provision of voter keys: Used by the voter to request the keystore that contains her set of voter keys.
- Cast a vote: Used to cast a signed and encrypted vote. Once received, it is checked that the voter is allowed to cast a vote and the vote is stored in the ballot box. A signature of the hash of the vote receipt is returned as a proof that the vote has been registered by the system.

3.2.6 Voting Client FE

The voting client is developed as a set of HTML and JavaScript files which run on the client side and more specifically in the voter's web browser. The voting client implements most of the cryptographic operations performed to protect the vote, achieving the end-to-end encryption previously mentioned. This component presents a graphical interface to the voter and interacts with the Voting Portal through the REST API. The most relevant operations performed are:

- Authentication: Requests the voter credentials and performs the corresponding derivations in order to authenticate the voter in front of the Voting Portal. In case of using an Identity Provider, it redirects the browser to this service.
- Vote encoding: Encodes the voter selections in a ballot.
- Vote receipt generation: Randomly generates a voting receipt that is delivered to the voter if the vote cast is successful.
- Vote encryption and signature: Encrypts the vote and its receipt with the election key and signs the encrypted vote with the voter key. Afterwards, the vote is sent to the Voter Portal.
- Receipt signature validation: Once the vote is cast, a signature of the hash of the receipt is returned, and this is validated before delivering the receipt and signature to the voter.

3.2.7 Receipts Website FE

This is a website that shows the list of receipts obtained after performing the counting of the votes. The receipts are presented in a single list that contains all the receipts. The voter has to search his receipt among the ones presented in order to be sure that his vote was processed by the e-voting system.

3.2.8 Receipts backoffice

This is the web server that supports the website that contains the voting receipts.

3.3 Scenarios of SMESEC

The scenario of application of e-Voting pilot was defined in document D4.1 "SMESEC Preliminary integration report on e-Voting SME pilot", section 3.2 "Scenarios of application". The only difference of the scenario, as it is shown in the architecture graphic of the same section in the current document, is that we have deployed the Credential Delivery and the Receipt WebSite. These two components will be used during the pilot execution. The Credential Delivery will be used in order to facilitate the delivery of the voter credentials to their owners. And the Receipt Website will be used by the voters in order to validate their receipts after the election closes.

3.4 Impact of SMESEC in the use case

In the e-Voting use case, SMESEC framework has increased the security at the infrastructure level that, until now, unless specific tools were installed, it was at application level only. In addition, the SMESEC framework allows us to monitor the system during its operation. Scytl's voting system already implements advanced cryptographic protocols and algorithms to protect the privacy and integrity of votes and results. It also implements end-to-end verifiability mechanism that provides transparency and auditability of the voting system. But, in addition to that, SMESEC provides the security layer for hardening, monitoring, attack detection and prevention as well as a method to ensure the availability of the election process. The integration of both technologies provides a joint solution that allows implementing secure online voting process to limited budget entities at the highest levels of security, availability and transparency.

3.5 Business impact

Scytl will be able to offer its e-Voting service combined with a robust security framework that will allow SMEs and public authorities to implement security measures in their election processes without requiring a large budget.

SMESEC will provide the security layer for hardening, monitoring, attack detection and prevention as well as a method to ensure the availability of the election process. The integration of both technologies will provide a joint solution that will allow entities with limited budget to implement secure online voting processes with the highest levels of security, availability and transparency.

Additionally, to the advanced cryptographic protocols and algorithms that Scytl uses to protect the privacy and integrity of votes and results, SMESEC framework provides a security layer at infrastructure level that allows hardening, monitoring, attack detection and prevention. All of this enhances the confidence of both SMEs and public authorities on the security of the electoral process, ensuring even more its availability. Such approach will help these entities to carry out secure consultation processes even with limited budgets. All of this will solve security and cost barriers that

prevent local authorities and small public entities to implement direct democracy practices and e-government.

In addition to all that, SMESEC framework becomes a tool to improve security training and awareness within the company.

4 Technical integration of SMESEC

4.1 Integration of SMESEC in the use case

The electronic voting system integrated with the SMESEC Framework has been deployed in an environment composed of several networks with different security privileges and policies. The different components and setup can be seen in the following picture:

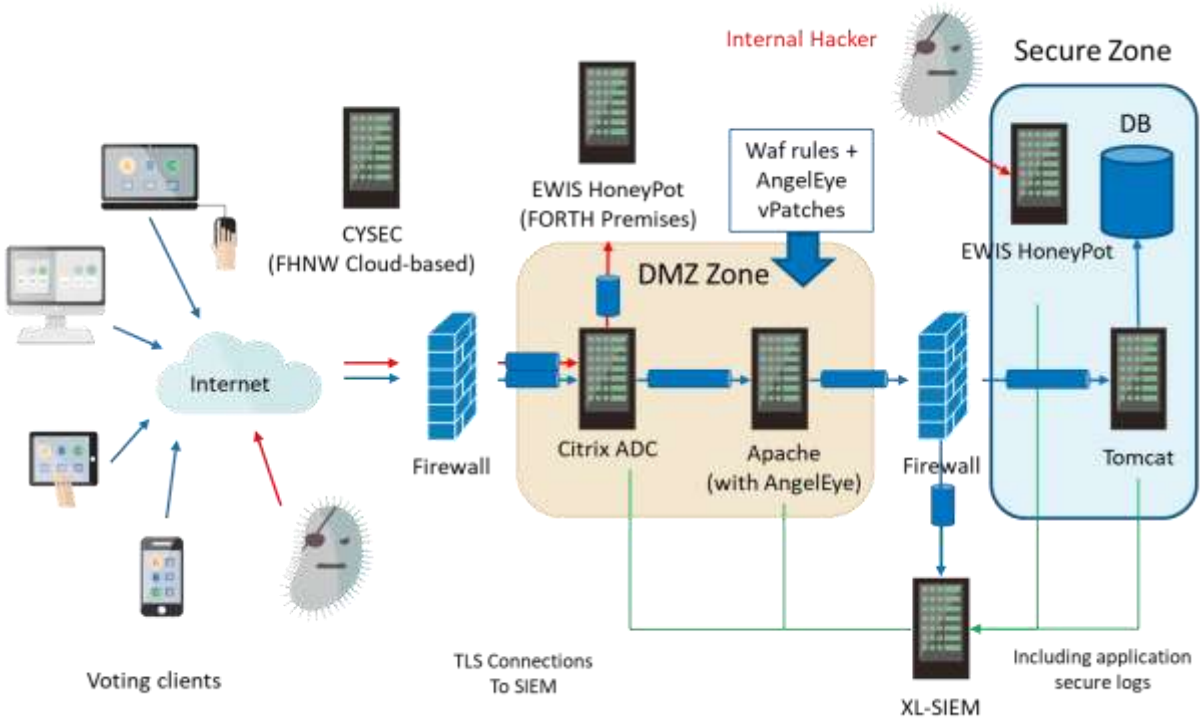

Figure 5: Online voting pilot components and setup

The voting system is composed of three main components: the web server (Apache), the application server (Tomcat) and the database (DB). The web server is deployed in the DMZ network, which is accessible through Internet. The application server and the database are deployed in the Secure Zone network, which is not directly accessible through Internet. In addition, when the voters connect to the system, a Javascript Voting Client is locally executed in their computers.

The integrated components are Citrix ADC, Angel-Eye, XL-SIEM and two instances of the EWIS HoneyPot. Citrix ADC is used as an application firewall; thus it is configured to be the first element that process the incoming connections that arrive from the Javascript Internet Voting Clients to the web server. The EWIS HoneyPot that, for the pilot, is externally deployed, is used as a system to

receive redirected connections rejected by Citrix ADC, i.e. connections that Citrix ADC have determined that are not compliant with the voting REST API. Angel-Eye is used as a periodically analyser of the HTTP requests received in order to detect attacks. The EWIS HoneyPot that is installed in the Secure Zone is a regular honeypot system used to attract attackers that are trespassing into this private network. And, finally, the XL-SIEM agent, deployed in a dedicated subnet, listens for Syslog connections from the other components deployed, e.g. web server, web application server, etc. The syslog of these components is forwarded to this agent that, in turn, forwards it to the external XL-SIEM server. The TaaS tool has not been finally deployed, because the information on how to use the tool was provided too late and the eVoting use case showed to be too complex to be integrated with this tool given the available time. In addition, Scytl already has a tool implemented by its security department that brings almost the same functionality than TaaS.

The components shown in the picture have been deployed in the EC2 of Amazon Web Services, although the same scenario is valid to be deployed in physical networks. From a perspective of the SMESEC Framework, the components selected can be used both in virtual and physical environments.

The tools integrated in this use case were Citrix ADC, EWIS HoneyPot, XL-SIEM and AngelEye. The following sections describe their integration.

4.1.1 Citrix ADC

4.1.1.1 Short description and characteristics

Formerly known as NetScaler ADC, this is an application delivery controller that performs application-specific traffic analysis to intelligently distribute, optimize, and secure Layer 4-Layer 7 (L4 - L7) network traffic for web applications. In our case it is used as a highly optimized Web Application Firewall (WAF) to filter the HTTP REST connections issued by the Javascript Voting Client. The deployed node analyses the content and the format of inbound connections and accepts or rejects them. Citrix ADC was integrated in our system by deploying a virtual machine instance in AWS cloud that contained the software bundled in the AWS-native AMI format. This instance was configured to have three network interfaces in three different subnets: client, server and management. The node was configured and managed through the management network interface using both the dedicated Command Line Interface (CLI) as well as a highly sophisticated Graphical User Interface (GUI) which allowed easier configuration. All inbound traffic arrives through the client interface and after the necessary inspection for verifying its validity is forwarded to the server interface, located inside the same subnet of the web server. Upon its configuration, Citrix ADC would provide dedicated security services to the associated Scytl Voting server, through a tailor-made, optimized traffic inspection service.

This tool was directly installed using the AMI provided by CITRIX in the AWS MarketPlace following their instructions. Later, they install the appropriate license and configured the service. No written instructions were provided at the moment of integrating the tool, thought we would have

preferred it to avoid depending on them for the installation. Such instructions are currently included in the Appendix of D3.4. Later we configured the WAF rules according to the REST API used by the Voting Client.

4.1.1.2 Usage in the pilot

The actual Citrix ADC deployment on AWS is a well-documented process, analysed in full detail in D3.2. In addition, D3.2 also provides a step by step guide for configuring Citrix Secure Web Gateway service. For additional information regarding these processes, readers are encouraged to refer to the aforementioned sources.

The Citrix ADC VPX is available as an Amazon Machine Image (AMI) in the AWS marketplace and enables customers to leverage AWS Cloud computing capabilities and use our solution's load balancing and traffic management features for their business needs. Citrix ADC VPX, the virtualized flavor of Citrix ADC, supports all the traffic management features of a physical appliance, and can be deployed as standalone instances or in HA pairs.

For the specific use case, we opted for deploying a standalone instance in Scytl's predefined Virtual Private Cloud (VPC) domain, located inside a specific AWS Region. Citrix ADC was deployed in a special availability zone as illustrated in Figure 10.

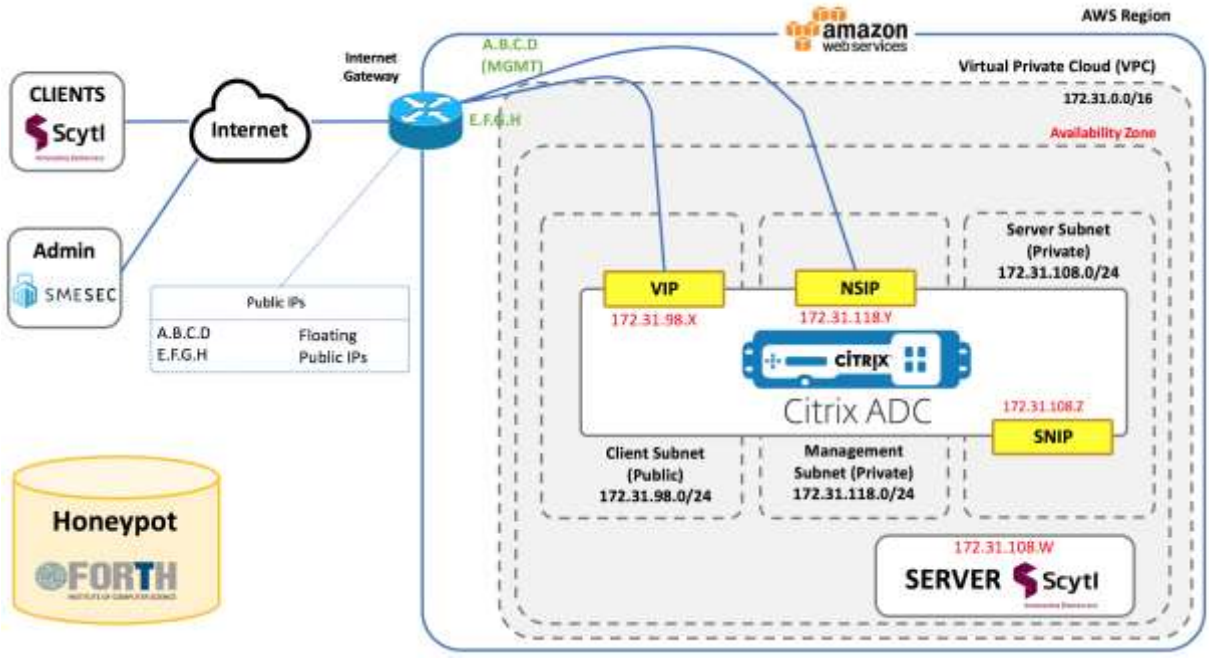

Figure 6: Current IP/Networking Configuration

For properly deploying Citrix ADC and enabling the full spectrum of its features, three distinct network interfaces must be available. These interfaces are linked to different subnets to avoid data leakage and each plays a different role in the overall Citrix ADC functionality. More specific, each Citrix ADC instance requires at least three IP subnets:

- A management subnet
- A client-facing subnet (VIP)
- A back-end facing subnet (SNIP, MIP, etc.)

Citrix recommends three network interfaces for a standard Citrix ADC instance on AWS installation and this is the approach we also followed for the specific use case. The Management subnet is a private subnet associated with the NSIP interface and is only used for providing administrative access to the deployed Citrix ADC node. For executing configuration commands users must obtain access to the Citrix ADC's Command Line Interface (CLI) through the NSIP and the Management subnet. This dictates certain connectivity of the NSIP to the AWS Internet Gateway preferably via hardcoded routes.

However, after the configuration of Citrix ADC in AWS, and despite establishing proper connectivity with Scytl Voting Server via the dedicated SNIP interface, no traffic traversed the node. The reason was an existing direct line of communication between the AWS Internet Gateway and the Scytl Server which virtually led traffic to bypass Citrix ADC and all its security features. Once discovered, this problem was tackled by configuring Citrix ADC with a Context Switching Server and registering in the DNS the IP of the VIP interface as the one the voters access to load the Voting Client.

The management route (connection to the NSIP interface in Figure 10) is used for accessing Citrix ADC CLI. The SMESEC administrator connects to the AWS Internet Gateway using a .pem file for enhanced security. Once obtain access to the AWS Region, the Internet Gateway routes the management requests to the Citrix ADC NSIP. This dictates the existence of a dedicated public IP associated with the management subnet of the Citrix ADC, therefore an additional one must also be allocated for traffic purposes. The traffic route (connection to the VIP interface in Figure 11) is used by voters to reach the Scytl Voting server which traverses Citrix ADC. As already stated, a dedicated public IP must be allocated for granting access to the AWS region infrastructure in which the Scytl VPC is deployed.

A high-level description of message exchange which takes place between the Scytl Client and the Voting Server, is presented in Figure 7. Once the voter logs into the Client (1), is able to cast a vote, thus initiate an interaction with the e-Voting server (2). When the vote is cast, the Scytl Server receives the inbound traffic (3) and then issues a response that informs the client that the vote is properly cast (4). This process relies on seamless communication between the Voting server and the Client and is augmented by the introduction of Citrix ADC and the FORTH EWIS as shown in Figure 8.

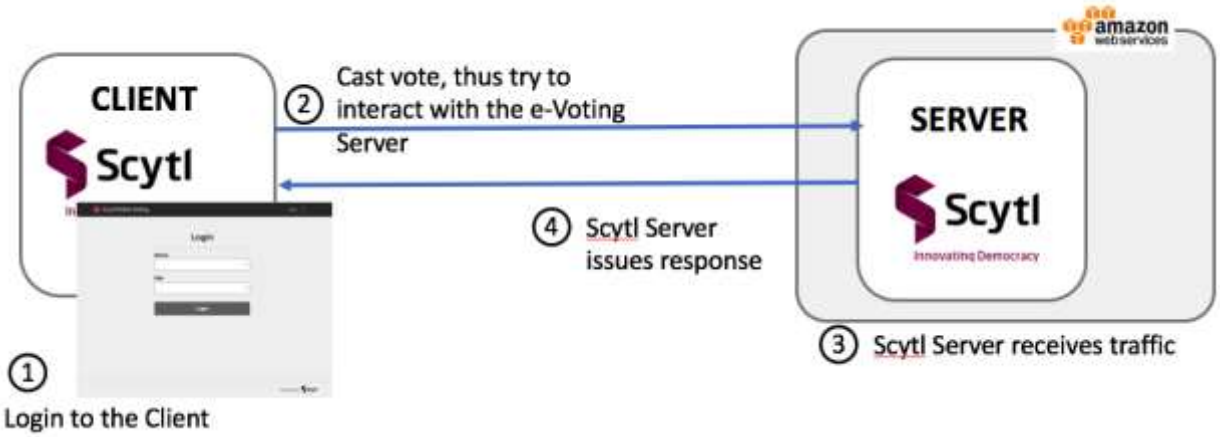

Figure 7: High-level e-Voting Platform Message Flow

Like the previous case, once the voter logs into the Client (1), is able to cast a vote, thus initiate an interaction with the e-Voting server (2). When the vote is cast, it is the Citrix ADC which is a-priori equipped with the proper certification/keys that receives the inbound traffic (3). After examining its content based on a set of rules provided by the administrators (4) decides whether or not the traffic is appropriate to be forwarded to the Scytl Voting server for further processing (5a) or be forwarded to the FORTH EWIS (5b) to be logged as malicious. Both Scytl Server and FORTH EWIS issue responses to forwarded traffic which are properly handled by Citrix ADC.

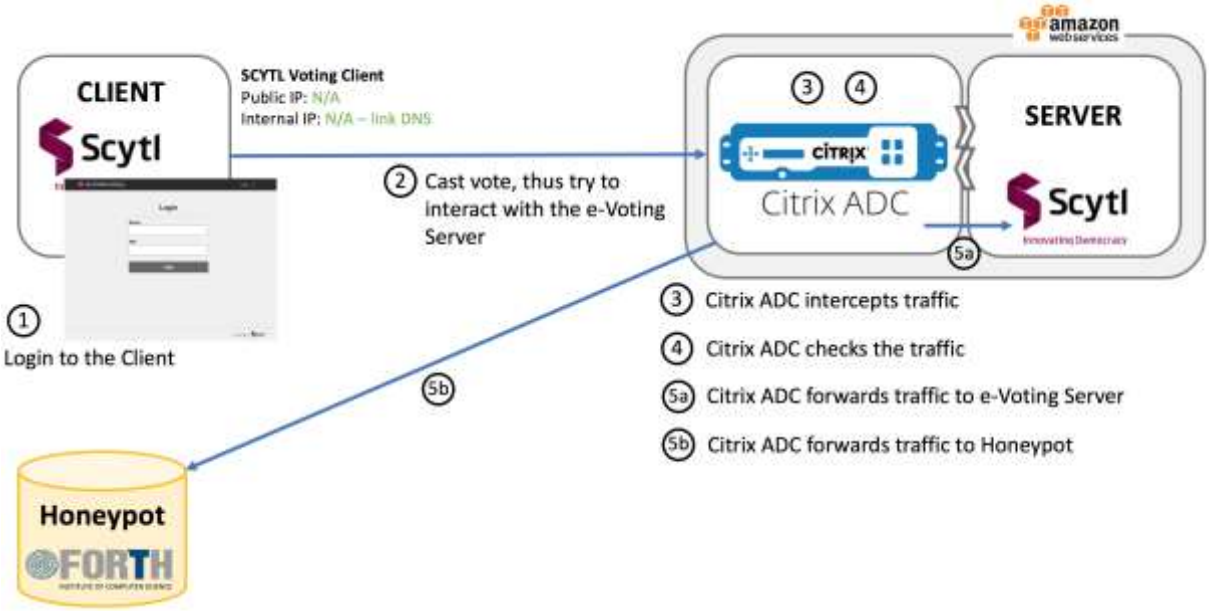

Figure 8: Demonstration Packet Flow

Citrix ADC VPX is attached to the overall networking topology as shown in Figure 9

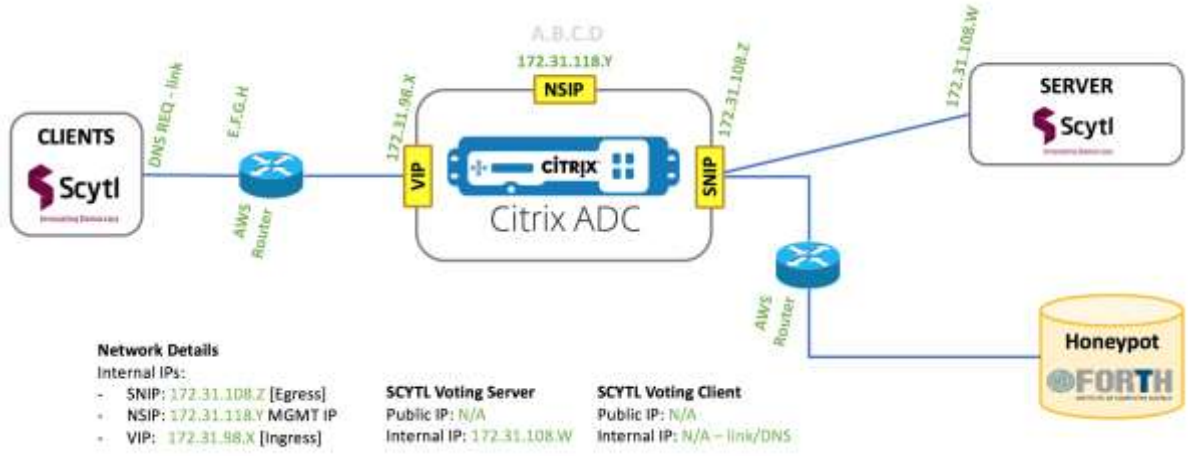

Figure 9: Abstract SMESEC e-Voting Pilot Network Topology Provisioning

Moreover, for the e-Voting Pilot we have deployed additional internal entities for Citrix ADC VPX, namely a Content Switching vserver and two distinct Load Balancing vservers, as shown in Figure 101. These internal entities are responsible for the overall functionality offered by SWG feature, as described in D3.2, however all irrelevant functionality of SWG that would not be utilized in the specific Pilot was discarded. Additional information on vservers and their functionality can be found in the Appendix of D3.2.

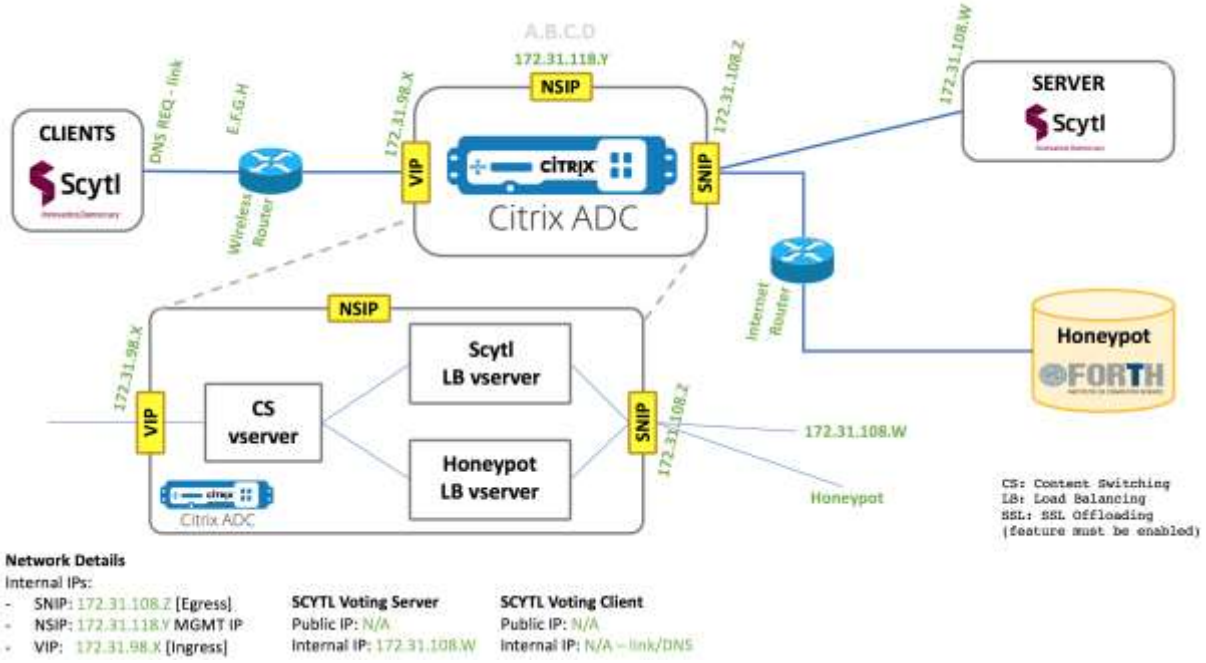

Figure 10: Internal Citrix ADC Networking Topology in respect to the overall networking of the Pilot

Citrix ADC Configuration

<H.IP>

Enabling Basic Parameters

Go to the Citrix ADC CLI and execute the following commands

```
> enable ns feature LB CS RESPONDER SSL
> enable ns mode FR L3 USIP Edge USNIP PMTUD
> set lb parameter –preferDirectRoute NO –vServerSpecificMac ENABLED
–sessionThreshold 450000
> set ns param –useproxyport DISABLED
```

Internal Citrix ADC Networking Configuration

```
> add ns ip 172.31.98.X 255.255.255.0 –type VIP -vServer DISABLED
> add ns ip 172.31.108.Z 255.255.255.0 –type SNIP -vServer DISABLED
> add route <H.IP> 255.255.255.255 GW_to_<H.IP>
```

SSL Certificates

Step 1

Upload the certificate [cert_name.crt] and the signup key [key_name.key] to the Node, using the GUI or to the /var/tmp folder using Linux CLI commands

Step 2

Go to the Citrix ADC CLI and execute the following command to add both files and combine them with the name sslckey

```
> add ssl certKey sslckey -cert "/var/tmp/ cert_name.crt" -key
"/var/tmp/ key_name.key"
```

Communication with the Scytl Server

Go to the Citrix ADC CLI and execute the following commands in order to

(i) notify Citrix ADC that a service called *scytl* must be associated with IP 172.31.108.W and since it involves encrypted traffic must also be associated with port 443. (ii) Add a Load Balancing vserver called *scytl-ssl* that handles all traffic (iii) Bind the previously created LB vserver to the *scytl* service and (iv) bind the LB vserver with the appropriate certificate and key pair combined in the previous step under the name sslckey.

```
> add service scytl 172.31.108.W SSL 443 -usip NO
> add lb vserver scytl-ssl SSL 0.0.0.0 0
> bind lb vserver scytl-ssl scytl
```

```
> bind ssl vserver scytl-ssl -certkeyName sslckey
```

Communication with the HoneyPot

Go to the Citrix ADC CLI and execute the following commands in order to

(i) notify Citrix ADC that a service called *honeypot* must be associated with IP <H.IP> and since it involves encrypted traffic must also be associated with port 443. (ii) Add a Load Balancing vserver called *honeypot-ssl* that handles all traffic (iii) Bind the previously created *honeypot-ssl* LB vserver to the *honeypot* service and (iv) in the case of ssl, associate the *honeypot-ssl* LB vserver with the appropriate certificate and key pair combined in the previous step under the name sslckey.

```
> add service honeypot <H.IP> SSL 443 -usip NO
> add lb vserver honeypot-ssl SSL 0.0.0.0 0
> bind lb vserver honeypot-ssl honeypot
> bind ssl vserver honeypot-ssl -certkeyName sslckey
```

Content Switching Server Configuration

Go to the Citrix ADC CLI and execute the following commands in order to

(i) notify Citrix ADC that a Content Switching vserver called cs-ssl must be associated with IP 172.31.98.X and since it involves encrypted traffic must also me associated with port 443. (ii) add a policy file called *legal_UA* which categorizes traffic based on a specific rule. (iii) If the policy rule is met, then traffic will be forwarded to the LB vserver *scytl-ssl* (as defined by the targetLBVserver parameter), otherwise (iv) traffic will be forwarded to the LB vserver *honeypot-ssl* and (v) in the case of ssl, associate the *cs-ssl* CS vserver with the appropriate certificate and key pair combined in a previous step under the name *sslckey*.

```
> add cs vserver cs-ssl SSL 172.31.98.X 443
> add cs policy legal_UA -rule "HTTP.REQ.HEADER(\"User-Agent\").SET_TEXT_MODE(IGNORECASE).STARTSWITH(\"Scytl\")"
> bind cs vserver cs-ssl -policyName legal_UA -targetLBVserver scytl-ssl -priority 100
> bind cs vserver cs-ssl -lbvserver honeypot-ssl
> bind ssl vserver cs-ssl -certkeyName sslckey
```

Optional Step for debugging purposes:

It is possible to create a specific policy that allows all traffic to pass through Citrix ADC and reach the Scytl Server entity. This policy does not offer any type of protection in terms of redirecting malicious or improper traffic to the honeypot, but could be used in cases where all traffic is considered appropriate and needs to be directly forwarded to its destination. In such a scenario, traffic inspection (if any) must have been conducted on an earlier phase since Citrix ADC does not make any type of

distinction of the actual content it forwards. The new policy is called *all_traffic_is_legal_UA* and the necessary commands to enforce it are the following:

```
> add cs policy all_traffic_is_legal_UA -rule TRUE
> bind cs vserver cs-ssl -policyName all_traffic_is_legal_UA -targetLBVserver scytl-ssl -priority 100
```

It should be stated here that the new policy must be applied to the Content Switching vserver, which acts as the first point of entry/decision entity of the internal Citrix ADC topology of vservers.

Generating e-Voting Policy

For generating a policy that will introduce the necessary intelligence in the Content Switching vserver and will allow proper redirection of traffic toward the appropriate Load Balancing vserver it is advised to use the GUI, as shown in the following figures.

After accessing the GUI, go to the *Configuration* tab and navigate to the *Traffic Management* drop down menu. From all available options, select *Policies*. This will bring up a page called *Content Switching Policies* from which we need to click the **Add** button.

Figure 11 Content Switching Policies list

Selecting the **Add** button, will bring forward another tab in which we need to insert the *Policy Name*. In the example of the following figure, the name *cs-encrypted-policy* is inserted, however there is absolutely no guideline for naming the available policies.

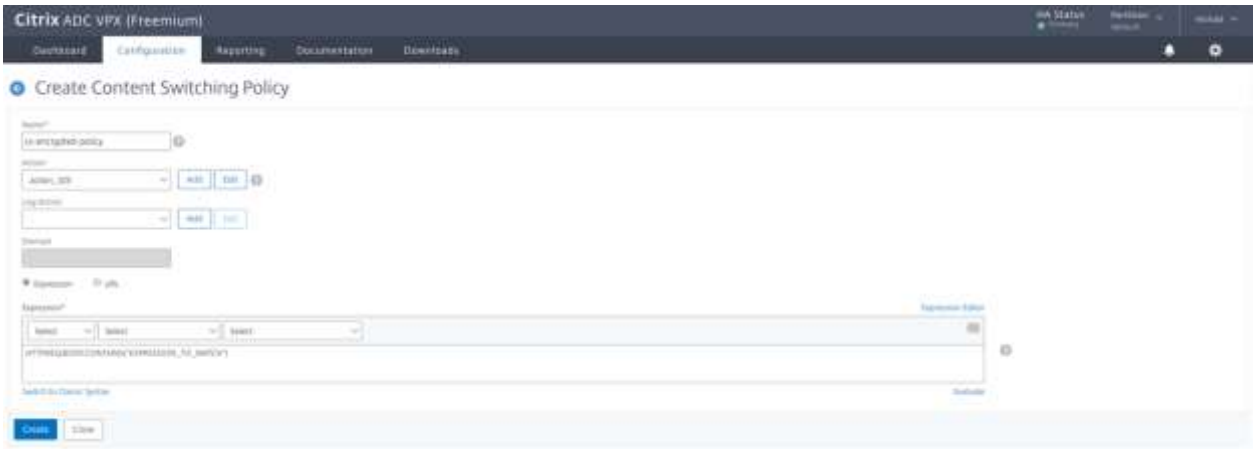

Figure 12 Creation of policy

After adding the name, navigate to the rightmost part of the tab and select the *Expression Editor* link. The *Expression Editor* link will bring up a panel in which all types of different configurations can be selected for each part of the Expression. In the specific example, we have selected **HTTP/REQ/BODY/CONTAINS(String)** for the first four parts of the Expression. This results into the expression preview shown in the previous figure: **HTTP.REQ.BODY.CONTAINS(“”).**

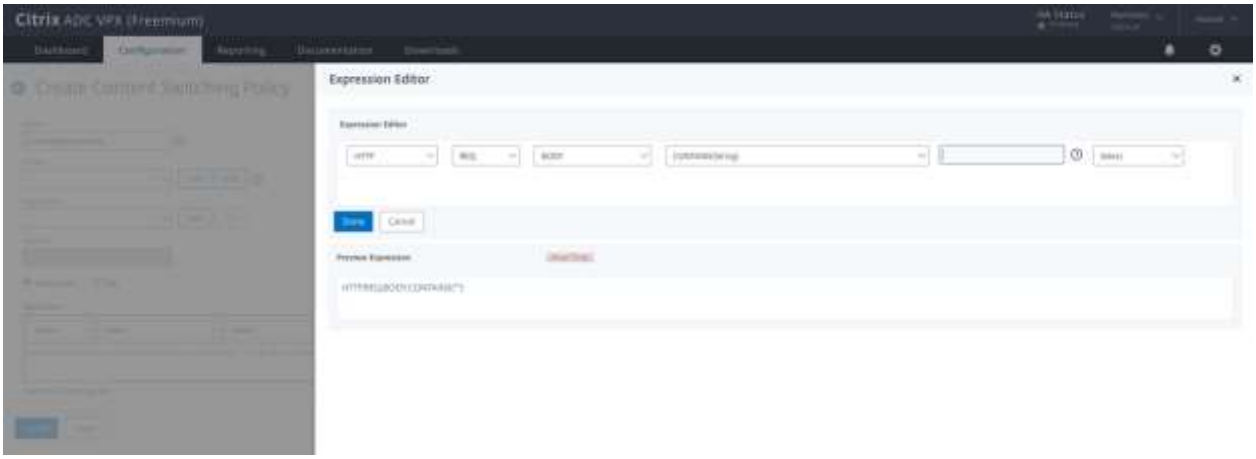

Figure 13 Expression Editor of policies

In the next part of the Expression Editor, we need to insert the actual string based on which all traffic will be categorized as appropriate or not. This string may be an Expression, a User ID, a Token, etc. As long as there is a match in the decrypted message body, Citrix ADC will identify it, categorize it as legit and forward it to the proper entity for further actions. After inserting the string, the Expression becomes **HTTP.REQ.BODY.CONTAINS(“EXPRESSION_TO_MATCH”)** and we should hit the *Done* button.

After hitting the *Done* button, we are forwarded back to the Create Content Switching Policy tab in which we should decide regarding the actual action Citrix ADC will perform when the inspected traffic meets the previously selected criteria. Since we intend to redirect the traffic to the Load Balancing vserver responsible to forward it on the Scytl Voting Server, we should go to the *Action* drop down menu and select the appropriate lbvserver name, in our case *scytl-ssl*.

Adding one more inspection control over the existence or not of a certain string is possible by inserting a second check in the policy file using the || as shown in the following Figure 22.

Policies can become complex enough to perform all types of control and act accordingly.

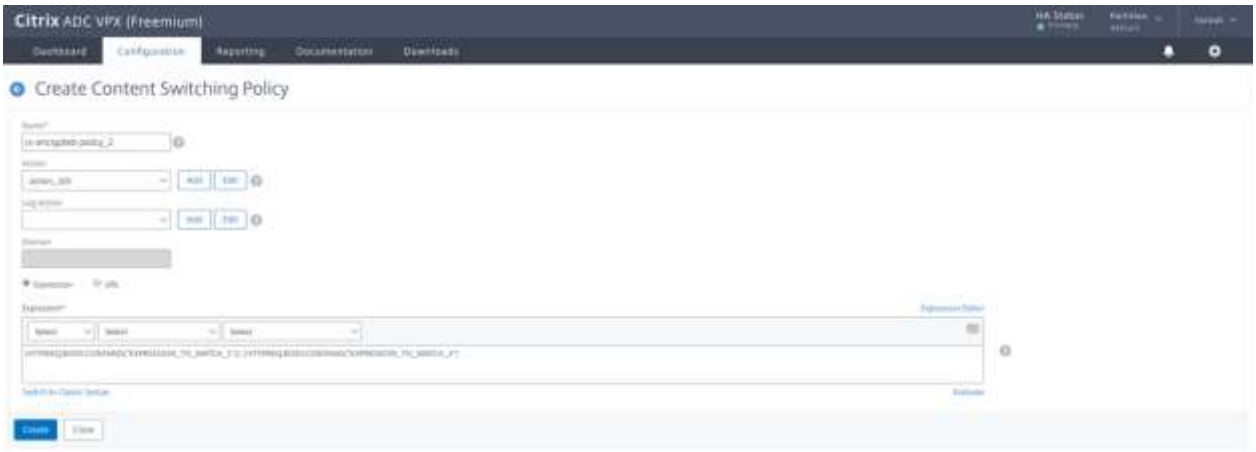

Figure 14 Creation of policy with complex rules

Once finalizing the policy generation, hit the *Create* button to conclude the process to create the policy. The policy can be then activated by going back to the *Content Switching Policies* page, tick the square box before the newly created policy and select the appropriate button.

After the policies are defined, go to the CS server to start the Policy Bind process that will associate policies with a given Context Switching Virtual Server. In order to do so, go to the Traffic Management -> Context Switching -> Virtual Servers section of the administrator website and right click the server where the rules have to be applied and select the edit option. From there in the area called “Context Switching Policy Binding”, after clicking on it, it is possible to add bindings. If the button Add Binding is clicked the following will appear:

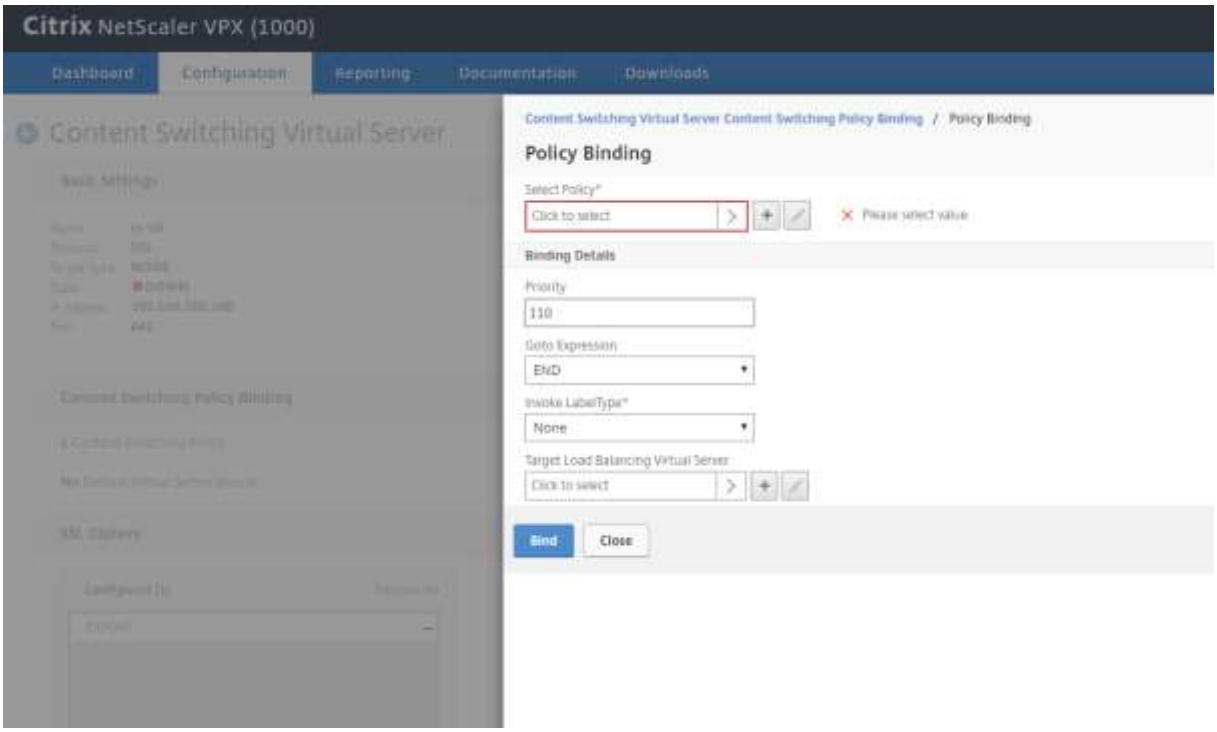

Figure 15 Policy binding to a Content Switching Virtual Server

Then the policy name has to be selected (in the example called "final") and the priority has to be setup (from the default 110 to 90) and in the Goto Expression drop down menu the priority of the other rule that is desired to go has to be written by selecting the "More" option:

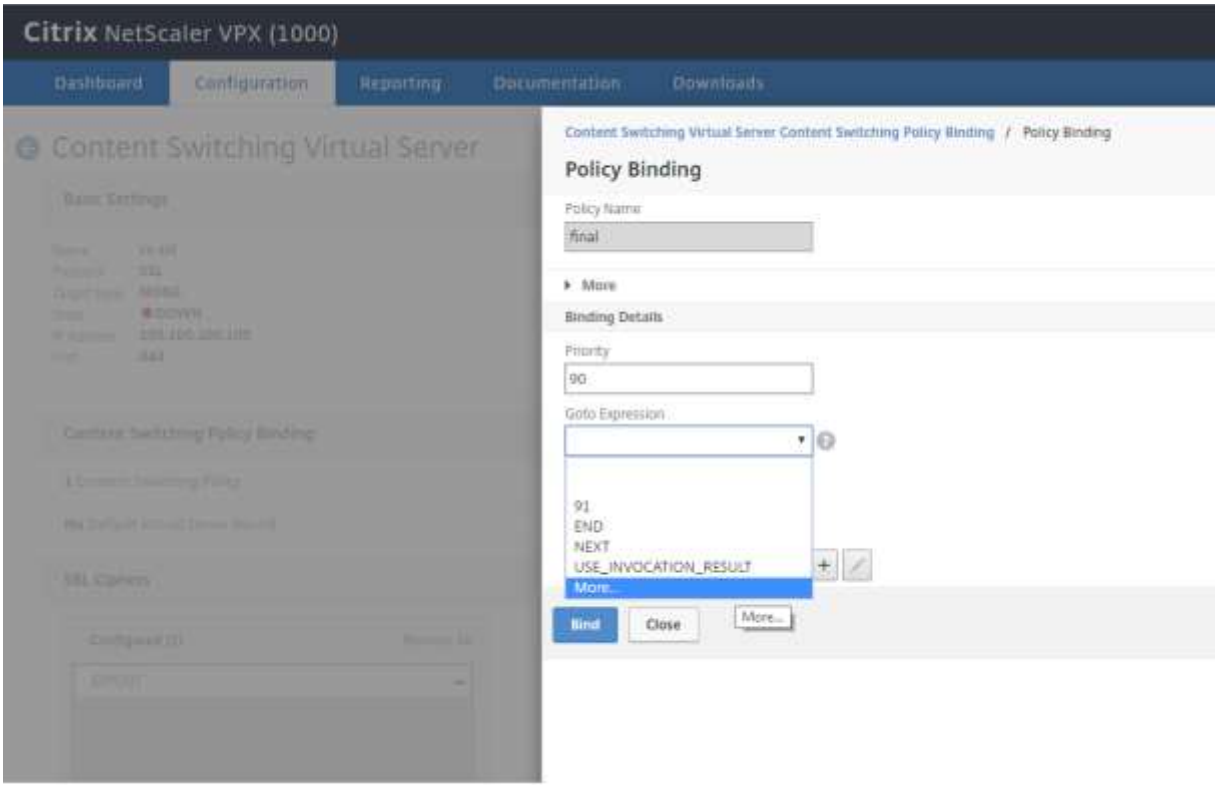

Figure 16 Usage of Goto Expression when binding policies

After selecting "More" the system redirects the user to following box where a custom numeric expression can be inserted. This custom numeric expression corresponds to the priority of the policy in which the system check must jump into once the current check results TRUE:

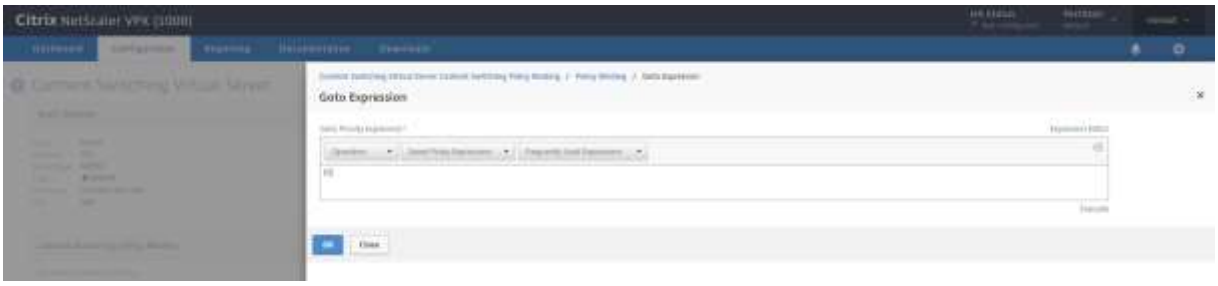

Figure 17 How to add priority expression using the Goto Expression editor

Then the policy is bound by clicking the "Bind" button:

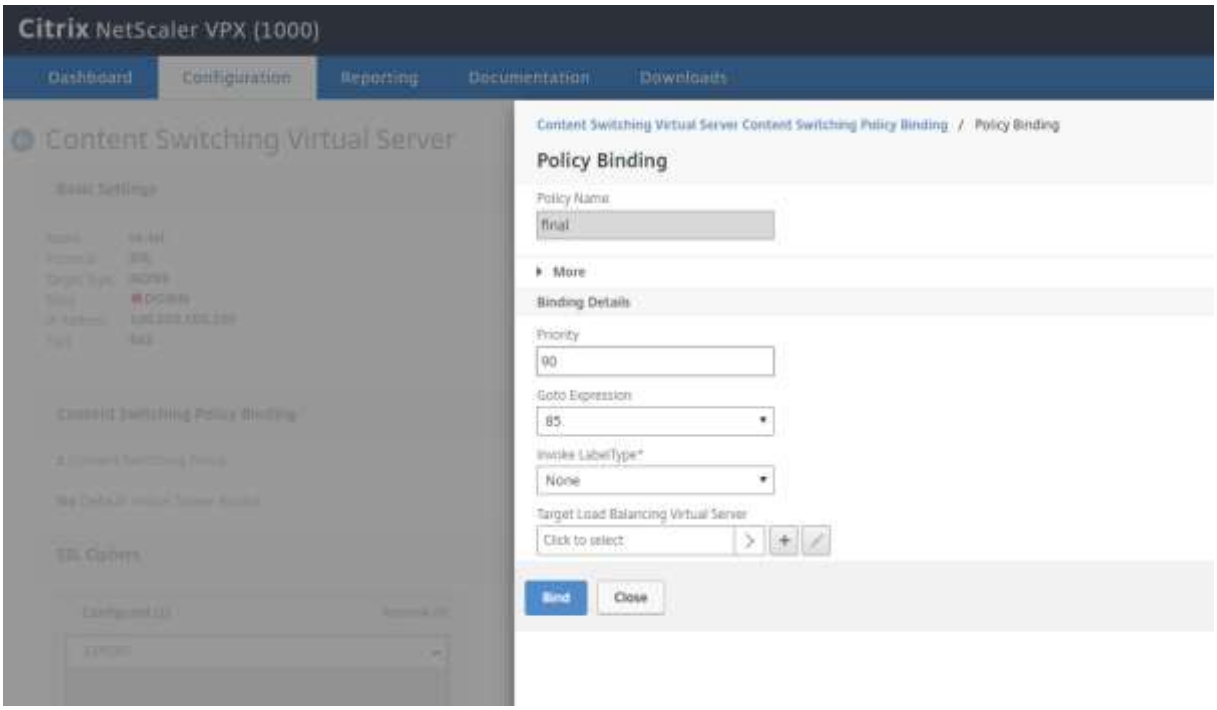

Figure 18 Policy binding example of parameters set

Finally, it can be verified that the new Policy has the specific Priority as well as the specific Goto Expression.

Policies generated

In order to filter at application level, the data transmitted from the Voting Client to the Voting Server, Scytl has generated the following rules:

Priority	Rule	Bind to
100	HTTP.REQ.URL.CONTAINS("clientLogin.html")	300
120	HTTP.REQ.URL.CONTAINS("clientGetCertificates.html")	500
140	HTTP.REQ.URL.CONTAINS("clientGetBallot.html")	700
180	HTTP.REQ.URL.CONTAINS("clientCastVote.html")	900
200	HTTP.REQ.URL.CONTAINS("index.html")||HTTP.REQ.URL.CONTAINS(".js")||HTTP.REQ.URL.CONTAINS(".css")||HTTP.REQ.URL.CONTAINS(".pem")||HTTP.REQ.URL.CONTAINS(".html")||HTTP.REQ.URL.CONTAINS(".png")||HTTP.REQ.URL.CONTAINS("custom")	
300	HTTP.REQ.BODY(135).REGEX_MATCH(re/^username=[0-9,a-f]{32}&password=password&electionId=[0-9,a-f]{32}&canDecrypt=0&loginType=advanced$/)	
500	HTTP.REQ.BODY(1000).REGEX_MATCH(re/^token=[0-9a-zA-Z%-.=]+&electionId=[0-9,a-f]{32}$/)	
700	HTTP.REQ.BODY(1000).REGEX_MATCH(re/^token=[0-9a-zA-Z%-.=]+&electionId=[0-	

	9,a-f]{32}&messageType=VOTE$/)	
900	HTTP.REQ.BODY(7000).REGEX_MATCH(re/^message=[0-9a-zA-Z%-.=_]+&token=[0-9a-zA-Z%-.=]+/)	

The first four rules detect the HTTP requests that are sent to one of the four endpoints used during the authentication and casting of a vote (login, get certificates, get ballot and cast vote). If one of these HTTP requests are detected, Citrix ADC is instructed to jump to the corresponding rule that will analyse if the HTTP POST Request is correctly formatted using a regular expression (i.e. one of the last 4 rules). If none of these HTTP requests are detected, then the fifth rule is applied, that checks if the connection is asking other types of data that are also required to load the Javascript Voting Client.

These rules have been configured in Citrix ADC following the procedure previously described in this section. A screenshot can be seen with the rules setup:

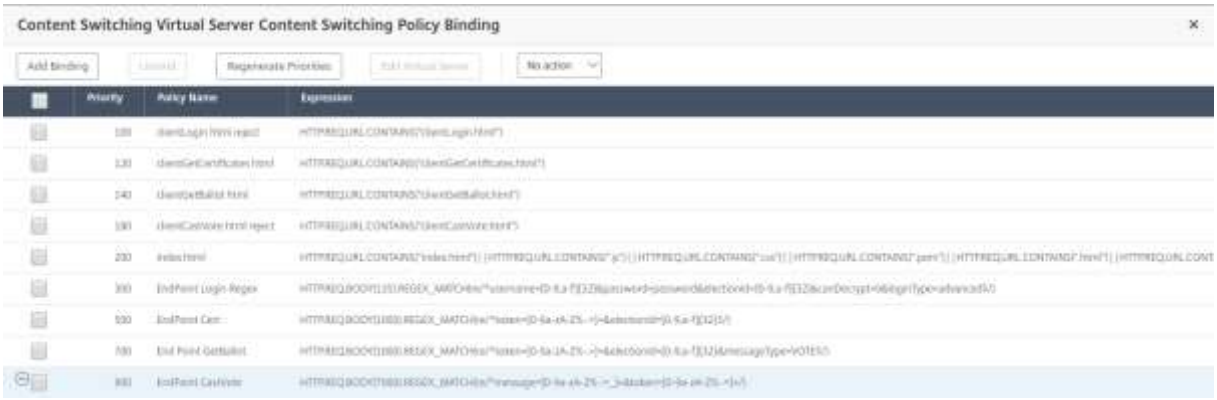

Figure 19 Rules defined for filtering the traffic of the Voting Client

4.1.2 EWIS Honeypot

4.1.2.1 Short description and characteristics

The honeypot is a software used to receive and register attacks. It works as an Early Warning Intrusion Detection System, luring attackers into attacking it instead of the production system. In that way, we are able to gain insight about the ongoing attacks in the network and provide valuable information to the system administrator. Currently, there are two instances of the EWIS Honeypot that have been deployed, one in the FORTH premises and another in the Secure Zone area of our scenario. The first instance, deployed in FORTH is connected to the Citrix ADC instance that is deployed in our premises. When an external connection arrives, it is firstly examined by Citrix ADC, then, based on ruleset created for Scytl, if it is not a valid request it is forwarded to the honeypot, for further examination and interaction with the potential attacker. If it is a valid request it is forwarded to the e-voting production system. The setup of this behaviour was directly configured by CITRIX and

FORTH. The honeypot that is deployed in the Secure Zone, is a honeypot that offers a set of services and tries to attract attackers to explore them. Both honeypots were deployed using a virtual machine provided by FORTH and directly configured by them. The honeypots included in the solution are able to emulate a variety of services, which are common targets for cyberattacks. So, we are able to detect attacks against FTP, TFTP, HTTP, HTTPS, TELNET, DNS, SMTP, MS Windows RPC, SMB, SSH, DNS, NTP, SNMP, NetBIOS and more. The events detected by the honeypots of EWIS solution, are sent to the ATOS cyber-agent that resides in Scytl's premises. The same information is also delivered to FORTH's EWIS backend databases. Moreover, the data collected to FORTH's databases are visualized by the EWIS Visualization platform. In that way, the system administrator can receive more detailed reports for each event captured, focusing on specific services, IP or date ranges. General statistics, can also be used, for assessing the overall cybersecurity state of the organization.

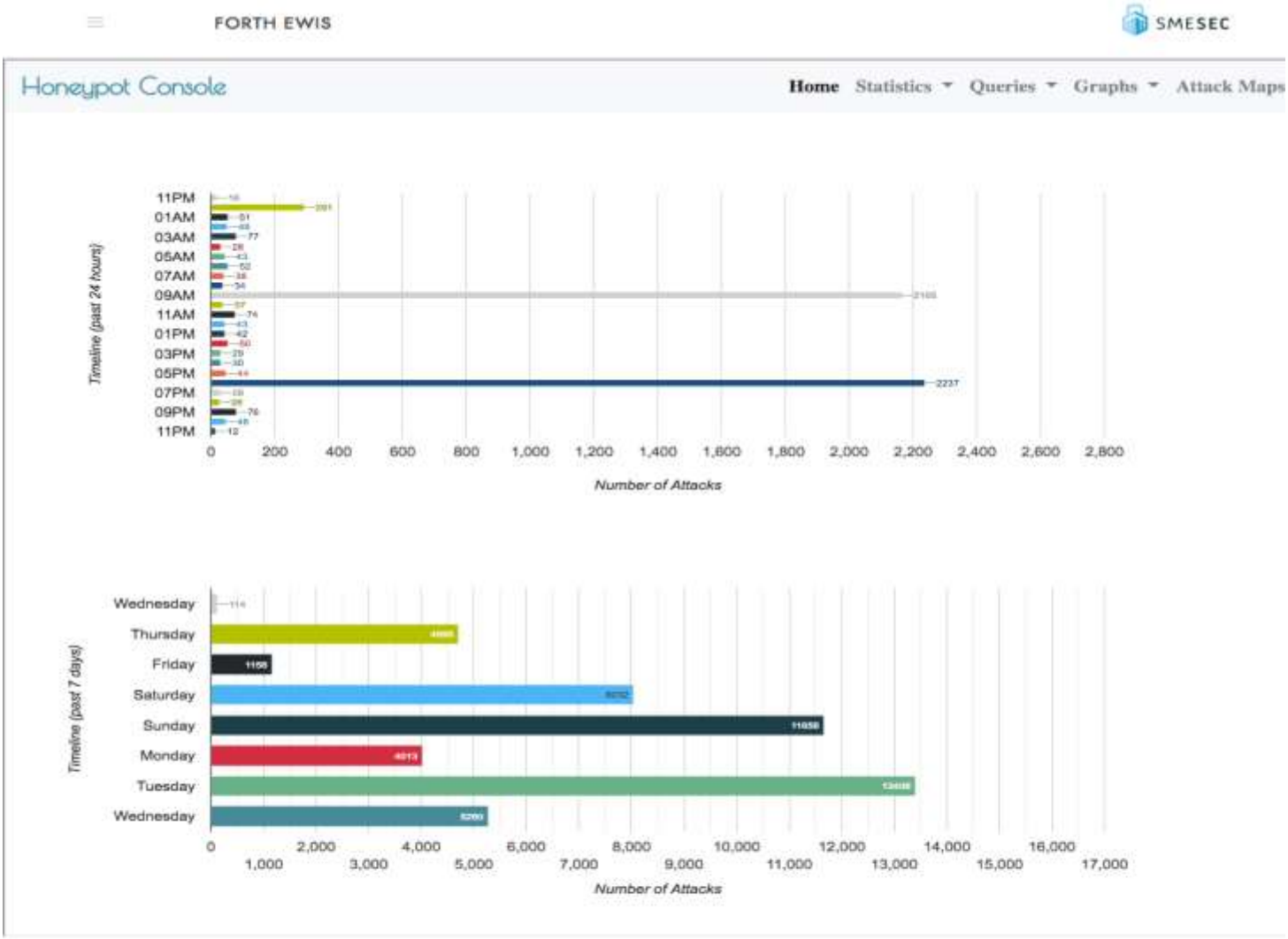

Figure 20 EWIS general attack statistics, as seen in SMESEC platform

Finally, EWIS has already been unified and incorporated in the SMESEC Framework and can be accessed in a unified way by all the users. In Figure 8, we can see a view with general attack statistics for the last 24h and the last 7 days. This view is taken from the incorporated SMESEC framework. In Figure 9 information about the top10 IP addresses that target MySQL service can be seen.

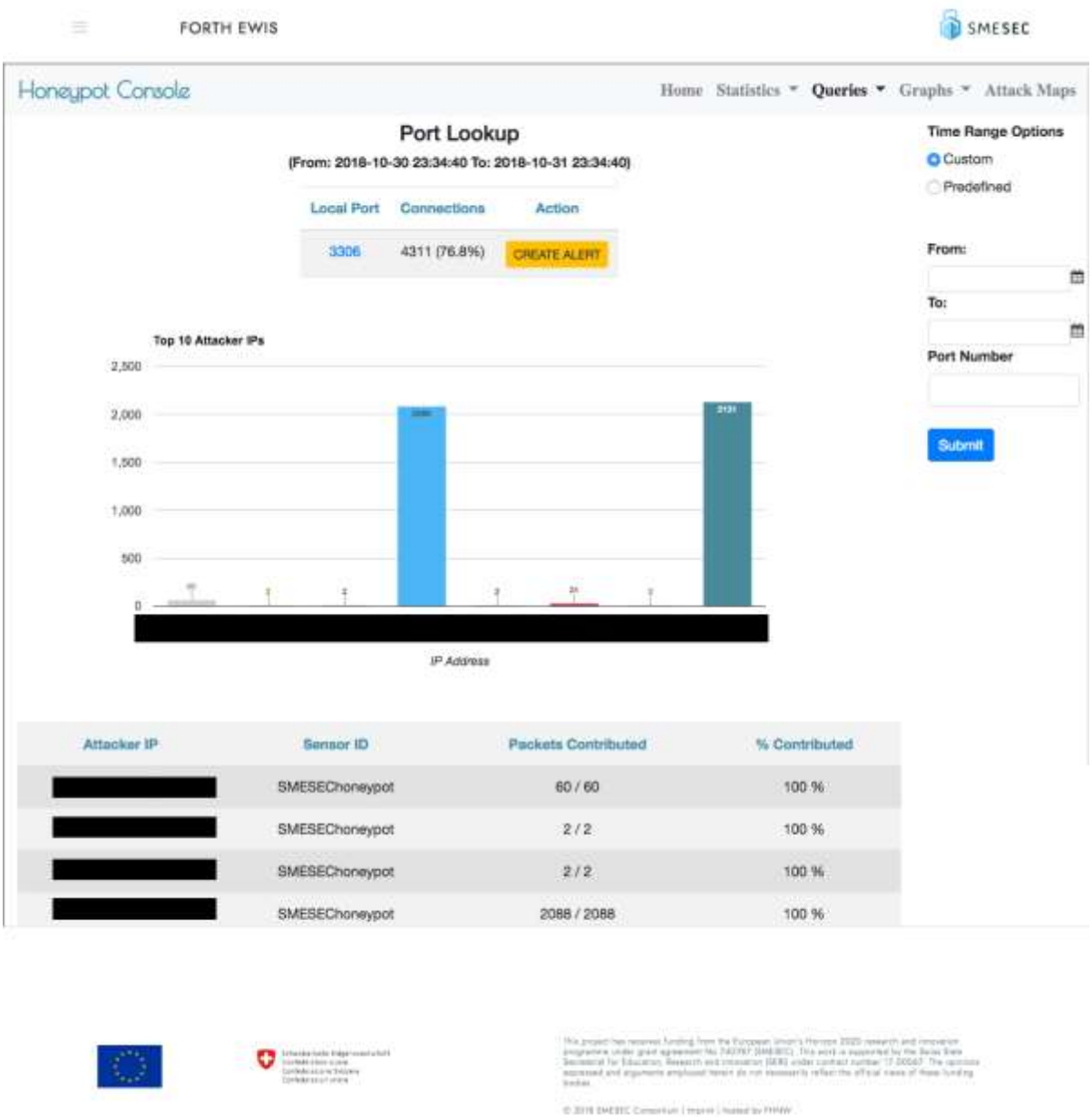

Figure 21 EWIS specific statics for a specific service (MySQL) based on the tcp port

4.1.2.2 Usage in the pilot

The EWIS Honeypot deployed in the Secure Zone of the AWS of this use case, has been deployed using an OVA image provided by FORTH that can be directly uploaded in AWS and deployed as an instance. Then, the instance has to be configured in order to start working.

Deployment of EWIS Honeypot in AWS

We have uploaded and deploy the image following this procedure:

1. Install AWS CLI:

```
$ sudo pip install awscli --ignore-installed six
```

2. Put your AWS credentials under `~/.aws/config`, see AWS CLI configuration reference for details

```
[default]
aws_access_key_id=foo
aws_secret_access_key=bar
```

3. Copy the OVA image into S3, assume you've already had an S3 bucket named `raw-images`

```
$ aws s3 cp example.ova s3
```

4. Create `vmimport` role and assign proper policy for the S3 bucket by following the AWS procedure here:

```
$ vim trust-policy.json

$ aws iam create-role --role-name vmimport --assume-role-policy-document file://trust-policy.json

$ vim role-policy.json $ aws iam put-role-policy --role-name vmimport --policy-name vmimport --policy-document file://role-policy.json
```

5. Create a JSON file `container.json` with the following content:

```
[
   {        "Description":  "FORTH  Honeypot  OVA",        "Format": "ova",                          "Url":          "https://s3.us-east-1.amazonaws.com/smesec/Honeypot.ova"  }

]
```

6. Import the image:
```
$ aws ec2 import-image --description "example image" --disk-containers file://container.json
```

7. Finally, you can launch an instance from the AMI created

Configuration of EWIS Honeypot

1. Edit configuration file /root/variables_script.sh

    a. Modify Network configuration in the file to that of your own network

    #configure network
    ipaddress=X.X.X.X
    netmask=255.255.255.0
    network=X.X.X.0
    broadcast=X.X.X.255
    gateway=Y.Y.Y.Y

    b. Then set the hostname of your honeypot and the IP address of the syslog server

    #configure hostname
    Hostname=honeypotX
    sensorIO=honeypotX

    #configure dianaea
    xmppserver=honeypot-console.domain.com
    xmppuser=username
    xmpppass=password

    #configure syslog
    syslogip: X.X.X.X
    syslogport: 514

    c. Make sure that the variables DIONAEA_run, KIPPO_run, DDOS_run are all set to 1

    DIONAEA_run=1
    KIPPO_run=1
    DDOS_run=1

2. Finish editing the configuration file, run /root/initialization_script.sh

3. Run netstat –lntp to make sure that the ports are correctly open

Later we had to install the plugin for XL-SIEM in order to process the logs of this component with the XL-SIEM event system (one plugin was installed on our XL-SIEM agent by Scytl and the other was installed in the XL-SIEM server by ATOS in their premises).

4.1.3 XL-SIEM

4.1.3.1 Short description and characteristics

This is a tool that collects events produced in the system and allows the study of them and also the generation of alarms based on the data collected. This tool is composed of a server that receives and process the data and also of agents that collect the data in place. In our case, we just have installed one agent in a dedicated subnet that collects the syslog of the different components of our scenario. The tool was provided as an installation package for Ubuntu/Debian. We deployed an instance of a Debian 9 in AWS and we installed the package with the instructions provided. Then, these servers were configured to send the listed information to the agent:

- Web Server: syslog data, ssh authentications
- Application Server: syslog data, secure logs data (data generated by the voting server Tomcat application) and ssh authentication
- Internal EWIS Honeypot: intrusion logs

Figure 22 XL-SIEM main dashboard

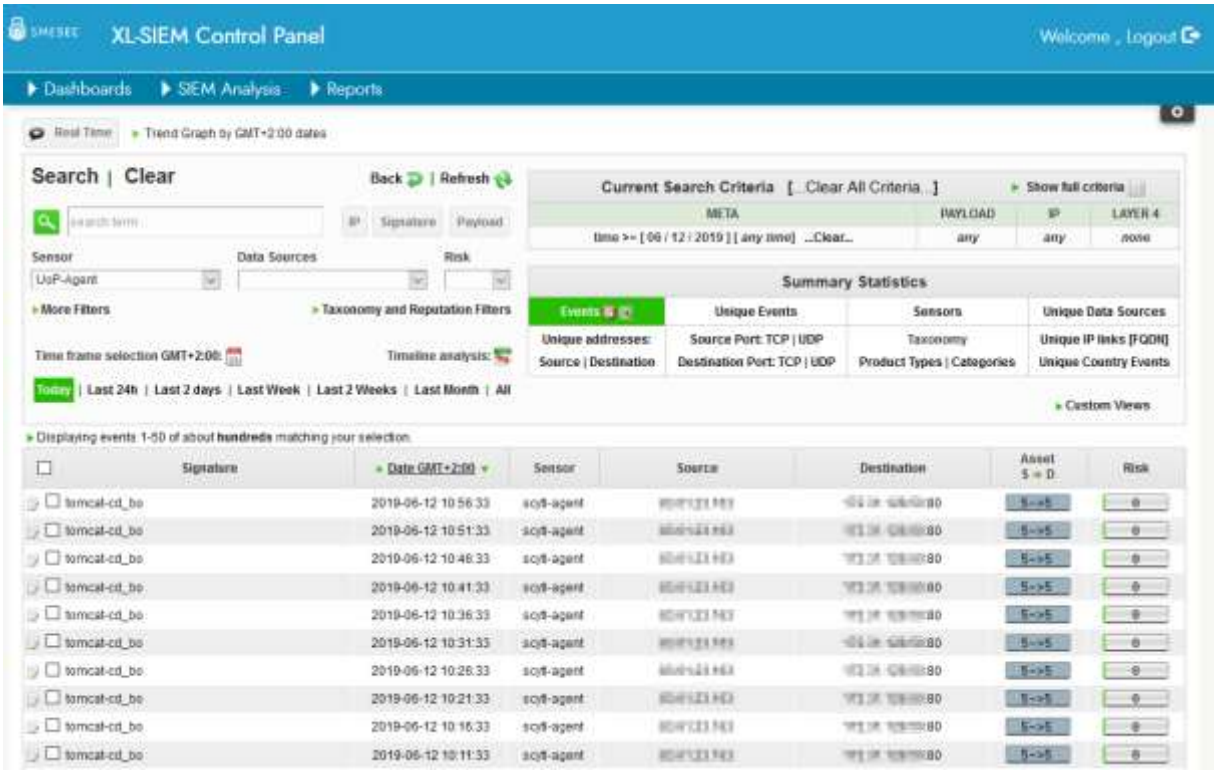

Figure 23 XL-SIEM dashboard to show events received

4.1.3.2 Usage in the pilot

The XL-SIEM agent is the part of this service that is installed in the customer premises. In the case of the pilot it was installed in the AWS together with the other components. An instance of a Debian image was created, and the packages provided installed. The instance was deployed in a dedicated subnet, since it is isolated from the rest of the hosts except for the syslog connections that it gathers. The agent installation was done as described in the next paragraphs.

Deploy AWS instance for XL-SIEM agent

In order to deploy the agent a Debian 9 was deployed in AWS using the Official Debian Amazon Machine Images (AMI)[1]. Then, a security group was created to open the syslog port in the instance (by default 514).

[1] https://wiki.debian.org/Cloud/AmazonEC2Image

Install dependencies

NOTE: The following commands are tested in a fresh installation of Ubuntu Server 16.04. You might need to change the commands to execute (especially in the case of MariaDB dependency depending of your OS).

```
> sudo apt-get update && sudo apt-get upgrade
> sudo apt-key adv --recv-keys --keyserver hkp://keyserver.ubuntu.com:80 0xF1656F24C74CD1D8
> sudo add-apt-repository 'deb [arch=amd64] http://ftp.igh.cnrs.fr/pub/mariadb/repo/10.1/ubuntu xenial main'
> sudo apt-get update
> sudo apt-get install libmariadbclient-dev
```

Install cyber agent package

To install the cyber agent package, execute the following commands:

```
> sudo apt-get install gdebi
> sudo gdebi xl-siem_cyber-agent_1.0.0-2_all.deb
> sudo systemctl enable cyber-agent
> sudo systemctl start cyber-agent
```

Override default files

Override the files installed by default with the ones provided by ATOS. In particular, you must add or override:

- `/etc/xl-siem/agent/agentuuid.dat`
- `/etc/xl-siem/agent/config.cfg`
- `/etc/xl-siem/agent/plugins/test.cfg`
- `/etc/xl-siem/agent/SSL/server.crt`

Redirect logs

To make use of the agent, some configuration is needed in rsyslog. Modify the file in:

`/etc/rsyslog.conf`

In case you have TCP and UDP modules commented, you need to uncomment those lines:

```
# provides UDP syslog reception
module(load="imudp")
input(type="imudp" port="514")

# provides TCP syslog reception
module(load="imtcp")
input(type="imtcp" port="514")
```

Also, you need to add some lines at the end of the file:

```
if ($rawmsg contains 'testSensor') then /var/log/test_logEvent.log
if ($rawmsg contains 'dionaea-syslog') and not ($rawmsg contains 'xl-siem') then /var/log/dionaea_syslog.log
```

NOTE: A sample file of this configuration is provided in the zipped file but modify it with care to not overwrite any possible configuration you may have.

When you are finish, restart the rsyslog service and the cyberagent:

```
> systemctl restart rsyslog
> systemctl restart cyber-agent
```

Test the installation

Copy the file logger.sh file with the following command:

```
> ./logger.sh <agent ip> <agent syslog port>
```

NOTE: You should give the script execution permissions with `chmod +x logger.sh`

Select the first option. If everything worked fine, a new event should appear in the XL-SIEM server.

Send logs to XL-SIEM

In order to configure the machines to send the logs to the XL_SIEM, we need to install syslog in each one:

1) To install the syslog

- yum install rsyslog

2) Then we configure the rsyslog to send to the XL-SIEM on the file configuration "/etc/rsyslog.conf"

- *.* @172.31.138.A:514

3) For each different application on the machine we create a file on "/etc/rsyslog.d/" that defines the location of the logs that will be send.

```
[centos@ip-172-31-108-w  rsyslog.d]$ cat apache_bo_rest.conf
# File 1
input(type="imfile"
      File="/var/log/httpd/govlab_bo_rest/govlab_bo_rest_access_log"
      Tag="apache"
      Severity="info"
      Facility="local4"
      reopenOnTruncate="on"
      PersistStateInterval="1")


# File 2
input(type="imfile"
      File="/var/log/httpd/govlab_bo_rest/govlab_bo_rest_error_log"
      Tag="apache"
      Severity="error"
      Facility="local4"
      reopenOnTruncate="on"
      PersistStateInterval="1")
```

4) When we finish with the configuration, restart the syslog.

Configure the XL-SIEM

In order to configure the XL-SIEM to receive logs from each of the machines:

1) Create on XL_SIEM(aws) 10-remote.conf and define which range ip will lisent1- Create on XL_SIEM(aws) 10-remote.conf and define which range ip will lisent:

   nano 10-remote.conf

   $ModLoad imudp$UDPServerRun 514$AllowedSender UDP, 172.31.0.0/16
   $template RemoteStore,
   "/var/log/remote/%HOSTNAME%/%$YEAR%%$DAY%.log":source, !isequal, "localhost" -
   ?RemoteStore:source, isequal, "last" ~

2) Restart de service of syslog:

   "sudo service rsyslog restart"

Also, in the XL-SIEM agent instance there must be one plugin for each type of data to be received by the agent. For example, for the application server instance (the one that contains the Tomcat server), we had to write a plugin for the type of information that it produces:

```
;; tomcat

[DEFAULT]
plugin_id=30001

# default values for dst_ip and dst_port
```

```
# they can be overwritten in each rule
dst_ip=\_CFG(plugin-defaults,sensor)
dst_port=80

[config]
type=detector
enable=yes

source=log
##location=/var/log/%(process)s/access.log,/var/log/%(process)s/error.log
location=/var/log/secure_logger_tomcat.log
# create log file if it does not exists,
# otherwise stop processing this plugin
create_file=false

process=tomcat
start=no
stop=no
startup=/etc/init.d/%(process)s start
shutdown=/etc/init.d/%(process)s stop

# list of sids (comma separated) to be excluded by the detector
exclude_sids=200

# Custom logs formats defined in apache2.conf
#             To             see             variable             definition:
http://httpd.apache.org/docs/2.2/mod/mod_log_config.html#formats

[1 - tomcat-cd_bo]
#Dec      17      15:01:07      ip-172-31-128-H      tomcat      2018-12-17
15:01:04,162|172.31.128.H|I.J.K.L.|CD_BO|LOGIN|000|Succesfully      logged
in|guest||New              Secret              Key              generated.
{*SG::RBOSTLS1wwRbWu5HWmiT8w7SiilZfaJE/yYMTHoQEEqDQ6KZPeApzZITXbgGVBfdTDawF
Tm9FilCmeqE/UJ1jbBl0QBm4QKeLcaIaKy8ipb0kMzp60B1QqIGdg6DgAVON6H6ZhxV8ngBETOc
Sd0s1lqQGupYF0uw8kJIUtQH5VnVqJVcbZcWahbH01HkwV1Wql45qfGWxeIhjUcsT3JRd02R8gW
lAg1kBLsMnvoLbjZqxXMVFAKCao2i/kLT7ElUUtNjZx6wUj//H3Y182Dob3cqraPMbIl51kDj+5
+9boRbyH1N7DmS4eusmQZhFuG7WIKVu6XgjgkxFWkRybi4fg==,LSK::O7koAogLmnvwtzt05OQ
lIybOue3gEKPPSC0nyzVGK9U=,ESK::S77ntSQxyUKFKmISiA8FCDsADgUSS+F8kdHJkwoycUP9
3H7aiAw9BmZ/1c2UIjBjvbMCxgmTZXTxscQxZJKnlAflRp6dp73oaGdsxO9by8K/fxSgc/wyzjw
MEcp3tAgMDKufd4O4/3driZLUEq1OZoxVxyB8r0PHJiGu9B2fv5Jx66KD81KHFOToYlO3yQBonS
DRuFDm311HLWCLWvF2ED29qN2kkRtUNTZ+2JY1p3jyqqhPRmO60PLwwnXrdNoqOlEooUFs3DtVd
EKpmPUKxyy3VqsvKIQ9rdRyRnqr496DMaQkwufZkM38fK1QPKWmGFQzsod0JZqxCxeKjMgoqA==
```

```
,PHMAC::lhV7n/BWzedrYeBVU0Zxc6IRCkQXUKvc2PblazESorU=,HMAC::d6L4cQen/9XwElZZ
bb7byNHfZl/jsr9EO8gRex2V8y0=*}

event_type=event
regexp=(?P<month>\D+)        (?P<day>\d+)        (?P<hour>\d{2}:\d{2}:\d{2})
(?P<nameSrc>\S+)                (?P<sensor>\S+)                (?P<date>\S+
\d{2}:\d{2}:\d{2},\d{2,3})\|(?P<dst>\S+)\|(?P<src>\S+)\|CD_BO\|(?P<request>
\S+)\|(?P<id>\S+)\|(?P<message>\D+)\|(?P<user>\D+)\|\|(?P<message2>\D+)
\{(?P<key>\S+)\}
src_ip={resolv($src)}
dst_ip={resolv($dst)}
date={normalize_date($date)}
plugin_sid=1
username={$user}
userdata1={$request}
userdata2={$key}

[2-tomcat-vote_portal]
#Dec      17      15:02:49      ip-172-31-128-H      tomcat      2018-12-17
15:02:43,469|RP|172.31.128.H|VOTRE|000|Vote                          received|
|172.31.108.W|WEB|NORMAL|mk8hnyzn|2c9f00b266f840420166f8dbf9760001|2c9f00b2
66f840420166f9163ec400e9|2c9f00b266f840420166f916411200fb|New   Secret   Key
generated.
{*SG::Zm9ZIbnSwl9Y7QrbGeRSTAs2siKThscmlwaKC7A4RnHskeqxEQTnODUN4WV9fweyBXlYl
GTxdQO1JFy1kJRnCQ749AJl/Li4wNtMnuZt6ZHM44bA+dYmHQIBbAPyoj9aAuksoVRiM1rw9HaR
ZmQnqisvIjlsp58xxZjjm5wkARiM4PUmtpF9a+U2BAZyGN7cxAWOiwFrTkDXuy52KA1BCcYIT6X
LlfU6SwG4FKweN7qUATbEkUPm45qRv+l7MUPgBkmA6gR/p0uva2y7moM9w+Ypiw+QGkSYGylZko
bT4RhKV5op1PhJTHHzWqDhqp0V+UuPc882v9j/vG3caJV47w==,LSK::9QZ03aRz8J0KfgK3Qfi
DZVBixMZgvav+wCfxpmAgzQI=,ESK::AI5x+sYKxaiOGnCh5XFSOWJ9nnZ9mFcZcSGScj9UKf/O
uFObjL2Mrs0omqk95qZ44RtxsYYnf9oaSzULvYEkZD786lA7qC8TlORYeYAJS1jnJpwblMt9m7h
TjREsMNlWUf8UDItGzdhddCYemOa7nVaRgixPQ8JdNYdCRwnDjty+i2hdHwRUdi/cxS6AINHGW+
Dv4zjAg8YGlNExmYseldPBrqeLhV2PVQarEf7NOhN7KMfRwjEncmloVdWTZoGpybf2tklN6o9il
uyW1O4GPj8DLaksvLs2gVByuLXgL2/rLNGTnLGK+tH6m5Jw6hTHYIbzKZtxyYP1de+TSao+5e0W
+Nbbq29druTFW0OXE2brl+i1vwMV6ikdGxGAw8qZikFc3aC3qBgmwqc72DDNicwiiq1IlIK2dGU
HyDUmA4wA28wmnrSHgQ==,PHMAC::3bsrVw7J+xztTfxEbpL9uZncwOM/RArk/4i2szHVtVE=,L
S::10000,TL::300000,TS::1545058963469,HMAC::vpFjbf0zbtzrLzUhUNNPewjyTHuMxv5
ZOEpwW1WwewM=*}

event_type=event
regexp=(?P<month>\D+)        (?P<day>\d{2})        (?P<hour>\d{2}:\d{2}:\d{2})
(?P<nameSrc>\S+)                (?P<sensor>\S+)                (?P<date>\S+
\d{2}:\d{2}:\d{2},\d{2,3})\|RP\|(?P<dst>\S+)\|VOTRE\|(?P<id>\S+)\|(?P<messa
ge>\D+)\|
```

```
\|(?P<src>\S+)\|WEB\|NORMAL\|(?P<string>\S+)\|(?P<electionid>\S+)\|(?P<elec
tionevent>\S+)\|(?P<electionidw>\S+)\|(?P<message2>\D+)\. \{(?P<key>\S+)\}
date={normalize_date($date)}
plugin_sid=2
src_ip={$src}
dst_ip={resolv($dst)}
userdata1={$message}

[3-tomcat-info-cd_bo]
#Dec      17      15:45:59      ip-172-31-128-H      tomcat      2018-12-17
15:45:51,641|172.31.128.H|I.J.K.L.|CD_BO|MODIFY_ELECTION|000|Succesfully
modified   election|admin||#election_name_modified=test,   start_date=17-12-
2018,                                                  end_date=30-12-2018
{*HMAC::FCQL2w/bNsVbKfHVWOyLLwjWw99Jkap6/zUEOWYtGig=*}

event_type=event
regexp=(?P<month>\D+)        (?P<day>\d+)        (?P<hour>\d{2}:\d{2}:\d{2})
(?P<nameSrc>\S+)                (?P<sensor>\S+)                (?P<date>\S+
\d{2}:\d{2}:\d{2},\d{2,3})\|(?P<dst>\S+)\|(?P<src>\S+)\|CD_BO\|(?P<request>
\S+)\|(?P<id>\S+)\|(?P<message>\D+)\|(?P<user>\D+)\|\|(?P<info>\D+\S+   \S+)
\{(?P<key>\S+)\}
src_ip={resolv($src)}
dst_ip={resolv($dst)}
date={normalize_date($date)}
plugin_sid=3
username={$user}
userdata1={$request}
userdata2={$message}

userdata3={$key}
```

4.1.4 AngelEye

4.1.4.1 Short description and characteristics

This is a tool based on neural networks that learns data patterns and evaluates if they are present in the data to analyse. For the e-voting use case this tool has been adapted to analyse the body of HTTP Requests that are used to perform the basic operations of the Javascript Voting Client. These requests are evaluated in order to find out patterns that may indicate that they are used to perform a zero-day attack to the voting server. AngelEye is being integrated directly in the host that contains the Apache Webserver and it will periodically run, analysing a set of logs that will contain the HTTP Posts received. Thus, the output of the tool is information about possible zero-day attacks that have been performed against the voting service.

4.1.4.2 Usage in the pilot

AngelEye is a tool that is based on the use of machine learning techniques, specifically on neural networks. This means that the usage of it does not only require a deployment and configuration of a client, but to the training of a neural network.

The deployment has been a bit complex, Scytl had to create scripts to generate datasets to train AngelEye. The datasets contain one several thousands, and up to one million, entries in some cases with HTTP requests that are considered correct and other that are considered bad (basically calls that produce the server to answer is an uncontrolled manner and that might be used in zero-day attacks against the system). After generating these datasets, the AngelEye evaluator has been installed in the Web Server of the Voting System. The Web Server has been configured an Apache plugin to export the HTTP POST Requests in a log file, the purpose of which is to be periodically analyzed by the AngelEye tool. The procedure to integrate this tool has consisted in the following steps:

1. Generation of scripts to create training and testing datasets
2. Generation of training and testing datasets
3. Integration of the tool in the WebServer
4. Testing of the tool in place

Generation of scripts to create training and testing datasets

The datasets to train the neural networks of AngelEye need to have several thousand up to a million of entries classified as good or bad. In this use case, the data analysed are HTTP requests issued by the Voting Client to the Voting Server. These requests perform the following actions: login, get election certificates, get ballot and cast vote. In order to automatize the generation of these requests and to produce good and bad requests, we created some scripts to automatize the process.

In order to do these datasets, we took advantage of a tool that we have generated in Scytl that is to perform automatic security testing. This tool, called Roti, allows us to instrument the Voting Client and perform as many requests as we want with the changes in the requests that we desire. Basically, we created four tests, one for each action, that for a percentage of requests randomly modifies the HTTP Request parameters contained in the body of the POST.

In order to introduce random modifications, we used a tool called Radamsa, which is a test case generator for robustness testing based on fuzziness techniques. The following lines shown an example of a login request without and with modifications applied with Radamsa:

- Login call without modifications:

```
username=5725352bfe21827c18c4c29983c970ed&password=password&electionI
d=8a84842e679856790167986ad0f4002c&canDecrypt=0&loginType=advanced
```

- Login call with modifications (marked in bold):

```
username=f48ab544c6af1738ff4ab0df72ca528b&password=password&electionI
d=8a84842e679856790167986ad4470îbc&canDecrypt=0&loginType=advanced
```

After doing these requests, if the response received is correct or is an error correctly handled by the server (HTTP 200 responses), it is tagged as benign. Otherwise, if the error received is not correctly handled by the server (other responses) then the request is entry is tagged as potentially malicious (because it can be used to exploit some attack in the server). The entries generated looked like the following one:

```
1.2.Login,dXNlcm5hbWU9ZjQ4YWI1NDRjNmFmMTczOGZmNGFiMGRmNzJjYTUyOGImcGFzc3dvc
mQ9cGFzc3dvcmQmZWxlY3Rpb25JZD04YTg0ODQyZTY3OTg1Njc5MDE2Nzk4NmFkNDQ3MO5iYyZj
YW5EZWNyeXB0PTAmbG9naW5UeXBlPWFkdmFuY2Vk,1,LOGIN.electoral_scope_errors.INV
ALID_ELECTION_ID,16003,1
```

Where the columns have the following meaning:

A. Name of the test or endpoint we are testing (it is constant, nothing to learn here)
B. Request parameters encoded in Base64 (line to be learnt by AngelEye)
C. Boolean that indicates if we apply fuzziness to the parameters (i.e., if Radamsa was called or not).
D. Error description obtained in case of failure
E. Time spends for this particular request
F. Boolean that indicates if the request returned the expected data and is considered benign

Generation of training and testing datasets

In order to generate the datasets, we have created 5 different elections and 1000 different voter credentials that were used several times to generate the amount of entries required. A total of 1 million entries for training were created for each of the endpoints (login, get certificates, get ballot and cast vote) and a smaller amount for testing. Of the entries, approximately 30% were intentionally non-modified and in the rest modifications were applied with Radamsa. Due to the fact that the server already can face some of the malformed requests and handle the error properly, not all of the 70% of modified requests are tagged as malicious as previously described.

Then, the datasets are used to train one neural network for each of the endpoints. Each one of these data sets was fed into the AngelEye tool as an CSV file. From each line in the CVS file an entry and a label were extracted and used for training of the deep neural network. Each entry was decoded into original format and no feature extraction was applied on those entries.

After the completion of the neural network training, the network was tested by a hold-out-set that was never seen by the neural network before. In the figure below a ROC curve of the login requests is shown. Last, the neural network was configured by choosing a threshold that satisfies the requirements and concerns regarding each endpoint.

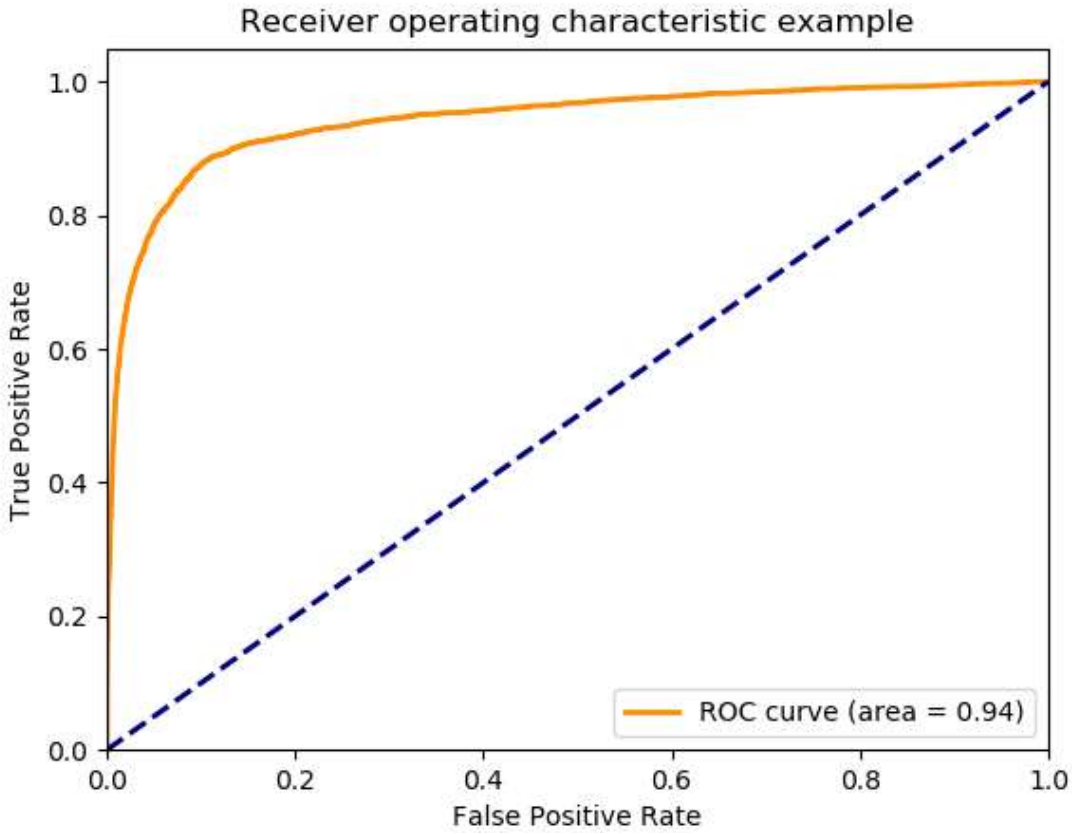

Figure 24: ROC curve for testing of login requests

Integration of the tool in the WebServer

In order to integrate the tool, we have modified the configuration of the WebServer to output a log with the HTTP POST Requests received. The log file was generated using the ModSecurity plugin including the following configuration:

```
<IfModule mod_security2.c>
    # ModSecurity Core Rules Set configuration
        #IncludeOptional modsecurity.d/*.conf
        #IncludeOptional modsecurity.d/activated_rules/*.conf

    # Default recommended configuration
    #SecRuleEngine On
```

```
    SecRequestBodyAccess On
    SecRuleEngine DetectionOnly
    SecRule REQUEST_HEADERS:Content-Type "text/*" \

"id:'200000',phase:1,t:none,t:lowercase,pass,log,ctl:requestBodyProcessor=X
ML"
    SecRequestBodyLimit 13107200
    SecRequestBodyNoFilesLimit 131072
    SecRequestBodyInMemoryLimit 131072
    SecRequestBodyLimitAction Reject
    SecRule        REQUEST_METHOD        "!^(?:GET|HEAD|PROPFIND|OPTIONS)$"
"phase:2,t:none,log,pass,msg:'Request',id:'966610',severity:'2',logdata:'%{
matched_var}'"
    SecRule REQBODY_ERROR "!@eq 0" \
    "id:'200001',  phase:2,t:none,log,deny,status:400,msg:'Failed  to  parse
request body.',logdata:'%{reqbody_error_msg}',severity:2"
    SecRule MULTIPART_STRICT_ERROR "!@eq 0" \
    "id:'200002',phase:2,t:none,log,deny,status:44,msg:'Multipart    request
body \
    failed strict validation: \
    PE %{REQBODY_PROCESSOR_ERROR}, \
    BQ %{MULTIPART_BOUNDARY_QUOTED}, \
    BW %{MULTIPART_BOUNDARY_WHITESPACE}, \
    DB %{MULTIPART_DATA_BEFORE}, \
    DA %{MULTIPART_DATA_AFTER}, \
    HF %{MULTIPART_HEADER_FOLDING}, \
    LF %{MULTIPART_LF_LINE}, \
    SM %{MULTIPART_MISSING_SEMICOLON}, \
    IQ %{MULTIPART_INVALID_QUOTING}, \
    IP %{MULTIPART_INVALID_PART}, \
    IH %{MULTIPART_INVALID_HEADER_FOLDING}, \
    FL %{MULTIPART_FILE_LIMIT_EXCEEDED}'"

    SecRule MULTIPART_UNMATCHED_BOUNDARY "!@eq 0" \
    "id:'200003',phase:2,t:none,log,deny,status:44,msg:'Multipart    parser
detected a possible unmatched boundary.'"

    SecPcreMatchLimit 1000
    SecPcreMatchLimitRecursion 1000
```

```
    SecRule TX:/^MSC_/ "!@streq 0" \
            "id:'200004',phase:2,t:none,deny,msg:'ModSecurity       internal
error flagged: %{MATCHED_VAR_NAME}'"
    SecStatusEngine On
    SecResponseBodyAccess On
    SecDebugLog /var/log/httpd/modsec_debug.log
    SecDebugLogLevel 1
    SecAuditEngine RelevantOnly
    SecAuditLogRelevantStatus "^(?:5|4(?!04))"
    SecAuditLogParts ABC
    SecAuditLogType Serial
    SecAuditLog /var/log/httpd/modsec_audit.log
    SecArgumentSeparator &
    SecCookieFormat 0
    SecResponseBodyMimeType       text/plain       text/html       text/xml
application/octet-stream
    SecTmpDir /var/lib/mod_security
    SecDataDir /var/lib/mod_security
</IfModule>
```

The log generated is similar to the following one:

```
--52998907-A--
[22/May/2019:14:39:18   +0000]   XOVfAvHwof3cExC1rBXTpAAAACo   172.31.108.Z   15562
172.31.108.W 443
--52998907-B--
POST /portal-webapp/clientGetCertificates.html HTTP/1.1
Host: portal-cruedav-devglab.amazon.innovascytl.com
User-Agent:  Mozilla/5.0  (X11;  Ubuntu;  Linux  x86_64;  rv:67.0)  Gecko/20100101
Firefox/67.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer:       https://portal-cruedav-devglab.amazon.innovascytl.com/js/js-lib/api-
worker.js
Content-Type: application/x-www-form-urlencoded
x-access-token:               [85ae242d/500a/3cdf/ac46/d8c6a5e77a71]-1553882416716-
8a84843169965fcc0169c48bcf330090-8a84843169965fcc0169c48bca9c001b-
OGE4NDg0MzE2OTk2NWZjYzAxNjljNDhhYzRlNDAwMDE=-
hKuAvOVEZgwNXWq/BDZMxqY24ha2uCV9otxpjbmRTGU=-
MGIxYWI1OTJkYjJmY2YzNGRmNjZhYTU0Zjc3OTVlZWM=-VOTER-0-
8a84843169965fcc0169c48ac4f50012-1.0-600000-webNormalKey-
Xre2DOhgAJ1E3WBAviAjzbIyrwIzppIZsfErlNq8Rrw3M5mUViOGxDOTwc0QPaM/OA4LmMGYg/qLNZeHdlf
```

```
achSFSl8Geq/a+dFp9yttQvZfbplueETCCKSpKpd+1mvi9fQn5ThLvzO97ZGQH4moo7sOVGbAx9L/WzfLyW
3ouH3EVrJs5W0O/hwPkGs6Ut//RQA67dncJYhjlnkqMIADfA9yR0UzFkoJ8a15i+DiF9VpDgW1LZRcKi1Ys
0zABw22Kn6axbjO6dl72IqXdwPVT4eW8+BL66fA6MT4D5Yqmm0jN2oNDmp1iepcTHT1q6B7S6hfhWL9F0NI
HyLIMd8NTQ==
Content-Length: 755
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache

--52998907-Z--

--52998907-A--
[22/May/2019:14:40:03 +0000] XOVfQ2Dxm3z4l9Es3TkYlQAAAEQ 172.31.108.Z 19170
172.31.108.W 443
--52998907-B--
POST /portal-webapp/clientLogin.html HTTP/1.1
Host: portal-cruedav-devglab.amazon.innovascytl.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:67.0) Gecko/20100101
Firefox/67.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://portal-cruedav-devglab.amazon.innovascytl.com/js/js-lib/api-
worker.js
Content-Type: application/x-www-form-urlencoded
INVOTE_API_KEY: webNormalKey
Content-Length: 135
Connection: keep-alive

--52998907-C--
username=5c0b4b87b0130d378dd2dfd94fa0b4f2&password=password&electionId=2c9f00b266f8
40420169574876a5025c&canDecrypt=0&loginType=advanced
--52998907-Z--
```

This log file is analyzed with the command line tool that, using the appropriate neural network, evaluates the requests that belong to the category to analyze.

The analyzer tool is a Python script that scans the request-logs according to the agreed format and logs out a report of prediction per unique entry. The tool was provided within a Zip file that included:

1. Pre-trained models for several endpoints
2. Configuration file
3. Python script to run the scanning of logs

For the installation it was required:

1- Python3.7

2- And the packages Keras, Tensorflow and numpy

3- Additional disk space for extracting the zipped request-log

Configuration file:

You can configure the path to pre-trained models, and the log output name.

Also you can configure the threshold used for tuning FPR-TPR rates

Script parameters:

1. --config <path to the above config file>
2. --input <path to the request-log zip file>
3. --output <path to the prediction file>

Testing of the tool in place

The tool was executed after issuing several requests that were considered malicious from the test data. The requests were issued using the Postman tool.

It is worth to explain that in order to perform the testing we disabled the Citric ADC rules of the tested endpoint. The reason is that some of the calls can be filtered directly by ADC if they don't comply with the rules defined in Section 4.1.1, which are very strict. Thus, AngelEye can be used without Netscaler and also as a second line of defense in case some of the malicious requests manage to bypass the application firewall rules of ADC .

4.1.5 CYSEC

4.1.5.1 Short description and characteristics

CYSEC is a tool that helps in monitor the adoption and adherence to good practices, while providing awareness training and self-evaluation methods for the staff that manage information security technologies in SMEs.

4.1.5.2 Usage in the pilot

CYSEC is integrated by the network hub but not with ICT network in the SME. Through SMESEC hub, the network administrator can have access to CYSEC, do cybersecurity self-assessment, see the recommendations (in the specific area based on the priorities), and communicate with the relevant staff in the company. Since CYSEC provides holistic SME-specific training and awareness content (cloud-

based or on-premise) for do-it-yourself cybersecurity assessment and capability improvement, it can integrate into the work process to improve the project. The version of CYSEC tool used was a prototype. When final and more user-friendly version of CYSEC tool will be available, it will be used to promote cybersecurity awareness within Scytl employees.

4.2 Analysis and evaluation of SMESEC

SMESEC has brought the possibility to complement the online voting system with a security framework that is adapted to the needs of SMEs. The following security benefits are obtained:

- Application level firewall: The connections to the Voting Portal, the most critical part of the service because it has to be online and available during all the election, are filtered at application level. Thus, in case an attacker tries to exploit this component with malformed requests to this service, the connection is redirected and send away from the server. This decreases the probability to compromise the Voting Portal.
- Intruders detection: If an intruder reaches the Secure Zone network, due to the actions exploring the network, we will receive alerts from the deployed HoneyPot included in the framework. This is useful to obtain an early detection of malicious activities within the system.
- Events and alarms: The framework allows to gather and show events that are generated by the security tools of the framework and/or directly by the online voting system components. Also, these events can trigger alarms when some of them happen. This can help to detect suspicious activities in the system that may indicate an attack is carried on or that is being prepared.
- Detection of zero-day attack: The framework provides mechanisms to detect malicious requests against the web servers, which may lead to zero day attacks, i.e. newly discovered attacks for which there is not fix available and the server is still unpatched. In our case we applied it to the Voting Portal in order to detect possible attacks that bypass the application level firewall.
- Security awareness: The framework provides online quizzes to evaluate the security of the company. In our particular case that we have a security team, this is not as necessary as other parts of the framework, but it can be useful to provide security self-assessment capabilities to our customers.

4.3 Testing and feedback provided

This section describes the tests that will be performed under WP5 test campaign.

4.3.1 Citrix ADC

This tool is being tested using a Scytl internal tool, ROTI, which simulates different parts of the voting process and tries to execute them automatically. The tool calls and modifies the logics of the Voting Client API in order to perform several security tests against the Voting Portal. These tests try to exploit common attacks to ensure the system cannot be compromised. We selected and adapted the following attacks to test Citrix ADC:

C655537 1.7.1 - Send a malformed request: Authentication request

C655538 1.7.2 - Send a malformed request: Certificates request

C655539 1.7.3 - Send a malformed request: Ballot request

C655980 1.7.4 - Send a malformed request: Web Service request

These tests issue an HTTP request to perform one of the actions described by their title with an error in the parameters of the request. Citrix ADC has to verify the HTTP requests received are compliant with the Voting Client API, thus in these tests it has to detect the modifications and discard the HTTP requests.

As described below for AngelEye, the same tool can be used to test Citrix ADC but instead of using preset tests, using tests that randomly modify the calls issued.

4.3.2 HoneyPot

There are two instances of the HoneyPot (external and internal). Both can be tested with synthetic tests generated with tools provided by FORTH. These tools execute the following synthetic attacks:

- Denial of Service attacks: A tool called HPing3 is used to generate UDP and ICMP flood attacks
- SQL injection attacks: A tool called Metasploit will be used to generate SQL Injection attacks to the HoneyPots.
- Brute-force attacks: A tool called Hydra will be used to perform brute-force attacks to the SSH services that are emulated by EWIS HoneyPots.

In addition, the External HoneyPot will be tested following the tests of Citrix ADC since these will redirect traffic to the HoneyPot that will be shown on the corresponding dashboards.

4.3.3 AngelEye

AngelEye can be tested with synthetic data generated with the same application used for training its neural network. The application issues HTTP Requests randomly modified that may behave similar to zero-day attacks. This tool includes four tests, one for each endpoint that AngelEye has to analyse:

- Login

- GetBallot
- GetCertificates
- CastVote

These endpoints are called by the Javascript Voting Client during a normal voting procedure, i.e. for authentication and casting a vote. Thus, the application just performs the same calls but applying another tool called RADAMSA, which randomly modifies the parameters that are passed as part of the HTTP POST Requests.

Running this application against our Apache WebServer (via Citrix ADC) we will be able to test both Citrix ADC and AngelEye. Most of the calls will be stopped by Citrix ADC and some of them, the ones that pass by this first filter, will be analysed by AngelEye (and eventually detected as zero-day attack). In case it is needed, Citrix ADC can be disabled to test more HTTP Requests that are usually filtered by this tool.

4.3.4 XL-SIEM

XL-SIEM is tested by using a synthetic tool to simulate events and by triggering real events. The synthetic tool provided by ATOS will be used to simulate all the possible events of the following sources:

- SSH authorizations
- Citrix ADC node
- Honeypot tool

The real events will be generated doing the following:

- SSH authorizations: A number of connections will be issued to the hosts monitored by XL-SIEM, i.e. Apache WebServer server, Tomcat server and Honeypot.
- Citrix ADC node: After executing the tests defined for Citrix ADC, we will check the information of the reported events is consistent with the tests.
- Internal HoneyPot tool: After executing the tests defined for the internal HoneyPot, we will check the information of the reported events is consistent with the tests.
- External HoneyPot tool: After executing the tests defined for Citrix ADC, we will check the information of the reported events is consistent with the tests (because Citrix ADC is redirecting connections that do not pass the validations to this HoneyPot).

5 Cybersecurity awareness and training

5.1 Training and awareness

At Scytl there is a Security department, led by the Director of Security and Data Protection Officer, and with 4 security analyst and researchers.

The Scytl Security department is reporting to the VP of Research & Development. Security is always a core priority in Scytl operations and products; by this way, not only the Security department is performing Security functions. The Security department is defining and leading the security activities, which are also coordinated with other departments, mainly the IT Department and the Testing Department.

The Security department is the core of the Security activities performed at Scytl, which can be executed by the Security Department itself, the IT Department - following the security policies defined by the Security department, or the Testing Department - executing the security tests defined by the Security department.

The Security department is in charge to promote and organize security trainings when the training necessity is identified.

They are also responsible to make Scytl personnel and partners aware of most recent security threats, attacks, and any security tendency which could be useful for Scytl activities.

A plan for training and awareness on the new SMESEC framework must be set-up. The Security Committee will approve a periodic plan of communication and training on security which will include the SMESEC framework use.

- Basic Training: A program for training and continuous awareness of issues related to security of information systems is defined. This training will include the use of new security framework which, due to its usability, is intended to be easy to use with little training.
- Advanced Training: Specific technical training will be given to the staff of the division of information systems and Security department that will integrate SMESEC framework in the online voting platform deployed for real elections

6 Conclusions

6.1 Final analysis and next steps

The second version of e-Voting has been almost successfully integrated and deployed, according to the requirements defined at the beginning of the project and based on the foreseen scenario. This prototype will be fully validated in task T5.3 "System Prototype Demonstration", were a demonstration will be done in the planned scenario.

The development has been made with some delays regarding the initial work plan, due to delays in the readiness of some tools to be integrated. But these delays have been recovered and the second prototype has been finished on time.

Once deployed, the second prototype with the framework and corresponding tools has provided us with application level firewall capabilities to filter the connections to the Voting Portal. Thus, malformed requests can be redirected away from the server decreasing the probability to compromise the service. It also provided intruders detection during actions taken by them when exploring the network. It also allowed the gathering and display of events occurred in the system. It also enabled the detection of malicious requests that may be exploiting zero-day attacks. And, finally, it also provided security awareness by including online quizzes to self-evaluate the security of SMEs.

Since the framework was not fully working during the time reported in this document, all tools have been installed one by one, with the necessary communications between them. This has made necessary a good technical background in order to integrate and deploy them. This is expected to be mitigated through the use of SMESEC framework as a whole, with all the tools integrated on it.
Next steps are to finalize the deployment of some of the tools and to fully test them as well as the framework, within the scope of WP5.

6.2 Fulfillment of objectives

Overall, the results of the integration activities conducted to obtain the second prototype of the e-Voting use case are satisfactory and meet the initial expectations, having obtained the expected results. All the core tools that were planned to be integrated in this second iteration have been successfully integrated, and only some deployment work is remaining.

Furthermore, SMESEC provides, through CYSEC tool, a way to improve cybersecurity through internal employees, as well as through clients.

But, as commented in previous subsection, integration process has not been all the straightforward that was supposed to be. This is expected to be improved when the process is done through the framework and not through the different tools

6.3 Future outcomes and business development

The main goal to be achieved using SMESEC security framework is to increase the security at the infrastructure level, as it currently is at application level only in no other tools are used. However, with SMESEC, Scytl will be able to offer its e-Voting service combined with a robust security framework that will allow SMEs and public authorities to be aware of their security by themselves and to add security measures in their election processes with a budget adapted to each case. This will enhance not only the level of security of its platform, with an additional security layer, but also its credibility with clients. Such approach will help these entities to carry out secure consultation processes even with limited budgets.

SMESEC will provide the security layer for hardening, monitoring, attack detection and prevention as well as a method to ensure the availability of the election process.

References

[1] **Deliverable:** SMESEC. *D.2.1 – SME security characteristics description, security and market analysis report.* Oikonomou, George. 2017

[2] **Deliverable:** SMESEC. *D.3.2.– SMESEC Unified Architecture*. Copty, Fady. 2018