



Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής  
Πρόγραμμα Μεταπτυχιακών Σπουδών  
«Πληροφορική»

**Μεταπτυχιακή Διατριβή**

Τίτλος Διατριβής	<b>Προστασία SCADA βιομηχανικών συστημάτων ελέγχου με μια ευέλικτη αρχιτεκτονική λύση ανοιχτού κώδικα</b> <b>Protection of SCADA Industrial Control Systems using a flexible open source architectural solution</b>
Όνοματεπώνυμο Φοιτητή	<b>Εμμανουήλ Σαμάνης</b>
Πατρώνυμο	<b>Κωνσταντίνος</b>
Αριθμός Μητρώου	<b>ΜΠΣΠ 16028</b>
Επιβλέπων	<b>Κωνσταντίνος Πατσάκης Επίκουρος Καθηγητής</b>

Ημερομηνία Παράδοσης **Οκτώβριος 2018**



**Τριμελής Εξεταστική Επιτροπή**

(υπογραφή)

(υπογραφή)

(υπογραφή)

Κωνσταντίνος Πατσάκης  
Επίκουρος Καθηγητής

Ευθύμιος Αλέπης  
Επίκουρος Καθηγητής

Ιωάννης Τασούλας  
Επίκουρος Καθηγητής

## Περίληψη

Σε αυτή την διπλωματική εργασία παρουσιάζεται ένας αυτόματος τρόπος εγκατάστασης και λειτουργίας ενός SCADA honeypot, το οποίο είναι ικανό να λειτουργεί σε ένα εσωτερικό δίκτυο και να εξομοιώνει την λειτουργία ενός πραγματικού SCADA/ICS συστήματος. Στην σημερινή εποχή, η διασύνδεση των SCADA συστημάτων με το internet είναι απαραίτητη, γιατί προσδίδει ακρίβεια και αποτελεσματικότητα. Αυτό μπορεί να γίνει είτε με σύνδεση απευθείας, μέσω των εσωτερικών δικτύων μιας υποδομής. Αυτό μπορεί να εξυπηρετήσει αρκετά γιατί μέσω Internet, είναι εφικτό να δημιουργηθούν διαδικασίες συντήρησης των SCADA συστημάτων, αλλά και ανάλυσης των δεδομένων που παράγουν. Για αυτό τον λόγο η έκθεση των κρίσιμων αυτών συστημάτων σε κινδύνους, είναι εξαιρετικά υψηλή και ακολούθως το ρίσκο για την διακοπή ή αλλοίωση της λειτουργίας τους. Οπότε οι λόγοι αυτοί καθιστούν επιτακτική την ασφάλεια αυτών των κρίσιμων υποδομών. Ένας τρόπος να γίνει αυτό είναι να «μάθουμε» από τους επιτιθέμενους. Οπότε η καινοτομία που παρουσιάζει αυτή η διπλωματική, είναι η εισαγωγή ενός τρόπου αυτοματοποίησης της διαδικασίας της εγκατάστασης και παραμετροποίησης ενός SCADA honeypot διαμέσου μια σελίδας που θα είναι προσβάσιμη σε ένα εσωτερικό δίκτυο από ένα διαχειριστή. Επιπλέον με την χρήση ανοιχτού τύπου κώδικα λογισμικών άριστα διασυνδεδεμένων όπως το OSSEC, μπορεί ένας διαχειριστής να παρακολουθεί και να ελέγχει σε ζωντανό χρόνο επιθέσεις στα εγκατεστημένα honeypots μέσω της διεπαφής του εργαλείου KIBANA, που αποτελεί κομμάτι του ELK stack. Το κεφάλαιο οχτώ ακολούθως, δείχνει τον αρχιτεκτονικό σχεδιασμό και την διασύνδεση όλων των ανοιχτού κώδικα τεχνολογιών. Η λύση που προτείνει η διπλωματική αυτή ξεκινά με την χρήση του εργαλείου Fabric, στο οποίο έχει γίνει η κατάλληλη υλοποίηση κώδικα και παραμετροποίηση ώστε να είναι ικανό να συνδεθεί με όλους τα μηχανήματα ώστε να μπορεί να εκτελεί με αυτόματο τρόπο εργασίες. Έπειτα δίνονται οι οδηγίες εγκατάστασης και παραμετροποίησης του OSSEC ώστε να μπορεί να δημιουργεί τους ελέγχους που είναι απαραίτητοι μαζί με την εξαγωγή εκθέσεων σχετικά με την λειτουργία και τις επιθέσεις. Στο τέλος δίνεται μια παρουσίαση των δυνατοτήτων την υλοποίησης που έχει γίνει με το εργαλείο Fabric και πως η αυτοματοποίηση εξυπηρετεί τις εγκαταστάσεις και τον έλεγχο. Επίσης ένα penetration test στο τέλος παρουσιάζει την συμπεριφορά του συστήματος όταν βρίσκεται υπό επίθεση. Σε όλη την διαδικασία το πρωτόκολλο ModBus χρησιμοποιήθηκε για την επικοινωνία με το PLC. Επίσης είναι ακόμα ανοιχτό το SNMP για την επικοινωνία όπως σε πραγματικά συστήματα. Η προτεινόμενη υλοποίηση και αρχιτεκτονική που αναπτύχθηκε μπορεί να προδώσει μεγάλη αξία σε εταιρίες που χρησιμοποιούν SCADA συστήματα γιατί θωρακίζει την σωστή λειτουργία των PLC και τον βελτιώνει τον έλεγχο των συστημάτων. Επιπλέον στην υλοποίηση μπορούν να προστεθούν αργότερα πολλές βελτιώσεις ώστε να εξελιχθεί το σύστημα με στόχο βελτίωση της ασφάλειας των SCADA συστημάτων.

**ABSTRACT**

The automated configuration of Conpot SCADA Honeypot architecture, presented in this thesis, is designed for operation inside an internal network and it is able to simulate a SCADA/ICS system. Today a part of business innovation which delivers efficiency and accuracy, is to connect SCADA systems to the internet, either directly or through internal networks. The internet connection for these systems is important because a process can be established for remote system maintenance, control and production data analysis. Hence considering the exposure of these systems to threats is enormous and accordingly the business risk for critical infrastructures. Hence for these reasons the need of securing the SCADA ICS devices is critical. One way is to learn from attackers directly by using SCADA honeypots. However as the concept of SCADA honeypot is well known in this thesis an innovation has been presented; an automation method to create and configure SCADA honeypots inside the internal network from a single web page. Thus an administrator can start and configure ad hoc honeypots. Moreover integration has been used with a well-known monitoring tool, OSSEC in order for the administrator to check and been informed of real time attacks in the honeypot systems with a variety of ways of through a modern Kibana console using ELK stack. Furthermore the chapter eight starts with the architecture design and information regarding all the open source programs that have been used and the workflow of the implementation. Firstly the solution starts with the steps used to implement the Fabric tool so as to be connected with all servers alongside with python code for the functions used for automation. Next instruction is given on the setup of OSSEC server and how the deployment created alongside with installation of agents and configuration of security policies and reports. Also in the end a wide demonstration of the solution is presented in two phases. First by showing the capabilities of Fabric and secondly how the architecture works when a honeypot is under attack. The proposed architecture is compatible with the majority of SCADA protocols, but in this thesis the most important is the Modbus protocol which is the key of this solution as this is the most popular in PLC implementation. Moreover, SNMP service on the honeypot is active as it is widely used in real SCADA environments. The solution this thesis proposes can add significant value to corporates because offers a whole system that can provide protection against PLC attacks. Besides that this system can hold many improvements for example can be trained in order to identify security patterns during an attack and provide the PLC administrators on time alerts.

## Contents

<b>TABLE OF TABLES .....</b>	<b>6</b>
CHAPTER 1 INTRODUCTION .....	8
<b>1.1 MOTIVATION .....</b>	<b>8</b>
1.2 ANALYZING THE NEED FOR INDUSTRIAL SCADA SECURITY.....	9
1.3 EXPLOITING SCADA SYSTEMS.....	10
1.4 SCADA ICS VULNERABILITIES .....	10
<b>CHAPTER 2 RELATIVE WORK .....</b>	<b>18</b>
2.1 RELATIVE PAPERS .....	18
<b>CHAPTER 3 SCADA SECURITY .....</b>	<b>19</b>
3.1 THE ROLE OF HONEYPOTS IN SCADA SYSTEMS.....	22
3.2 THE USAGE OF SNMP PROTOCOL IN SCADA SYSTEMS .....	23
3.3 HIDS FOR SCADA .....	24
3.4 COMPONENTS OF A SCADA SYSTEM.....	26
3.5 ATTACKING A SCADA SYSTEM .....	26
3.6 SCADA FORENSICS.....	27
<b>CHAPTER 4 CONPOT-0.5.1.....</b>	<b>28</b>
4.1 CONPOT INSTALLATION .....	28
4.2 UBUNTU 12.04 LTS / 14.04 LTS.....	28
4.3 CENTOS 7.3 .....	29
4.4 DEBIAN 7.2.0 64BIT & 6.0.7 64BT .....	29
4.5 HOW TO USE CONPOT.....	30
<b>CHAPTER 5 OSSEC .....</b>	<b>32</b>
5.1 OSSEC FEATURES .....	32
5.2 OSSEC ARCHITECTURE .....	32
5.3 INSTALLATION.....	33
5.4 USAGE .....	34
<b>CHAPTER 6 FABRIC AUTOMATION TOOL.....</b>	<b>34</b>
6.1 INSTALLATION.....	34
6.2 USAGE .....	35
6.3 FABRIC-WEB.....	35
<b>CHAPTER 7 ELK STACK.....</b>	<b>37</b>
7.1 ELASTICSEARCH.....	38
7.2 LOGSTASH.....	38
7.3 KIBANA .....	38
7.4 INSTALL ELASTIC.....	39

7.5 USAGE .....	41
<b>CHAPTER 8 IMPLEMENTATION.....</b>	<b>41</b>
8.1 ARCHITECTURE - DESIGN.....	42
8.2 SOFTWARE.....	42
8.3 VIRTUAL MACHINES OS .....	42
8.4 DEPENDENCIES .....	43
8.5 WORKFLOW - CUSTOMIZATION.....	43
8.6 PROOF OF CONCEPT DEPLOYING CONPOT .....	57
8.7 PROOF OF CONCEPT ATTACKING CONPOT .....	72
8.8 OSSEC REPORTING .....	75
<b>CHAPTER 9 FUTURE WORK – CONCLUSIONS .....</b>	<b>79</b>
9.1 FUTURE DEVELOPMENT .....	79
9.2 CONCLUSIONS - SUMMARY .....	79
<b>CHAPTER 10 REFERENCES .....</b>	<b>80</b>
<b>APPENDIX A .....</b>	<b>81</b>
<b>APPENDIX B .....</b>	<b>90</b>

## Table of tables

<b>Figure 1 Business System versus ICS Risk</b> .....	<b>9</b>
Figure 2 Policy and Procedure Vulnerabilities .....	<b>11</b>
Figure 3 Platform Configuration Vulnerabilities .....	<b>13</b>
Figure 4 Platform Hardware Vulnerabilities .....	<b>14</b>
Figure 5 Platform Software Vulnerabilities .....	<b>15</b>
Figure 6 Platform Malware Protection Vulnerabilities .....	<b>16</b>
Figure 7 Network Configuration Vulnerabilities.....	<b>16</b>
Figure 8 Network Hardware Vulnerabilities .....	<b>17</b>
Figure 9 Network Perimeter Vulnerabilities .....	<b>17</b>
Figure 10 Network Monitoring and Logging Vulnerabilities.....	<b>18</b>
Figure 11 Communication Vulnerabilities .....	<b>18</b>
Figure 12 Wireless Connection Vulnerabilities .....	<b>18</b>
Figure 13 Modern/Common Diagram .....	<b>20</b>
Figure 14 Modern/Proprietary Diagram .....	<b>21</b>
Figure 15 Legacy/Proprietary Diagram .....	<b>22</b>
Figure 16 Modbus honeypot software architecture.....	<b>23</b>
Figure 17 Wingpath ModSnmp diagram .....	<b>24</b>
Figure 18 Sample of a simple ICS.....	<b>25</b>
Figure 19 List of activities for pen testing .....	<b>27</b>
Figure 20 Running Conpot .....	<b>31</b>
Figure 21 Default index page .....	<b>36</b>
Figure 22 Example of command execution .....	<b>37</b>
Figure 23 OSSEC Integrated with ELK stack.....	<b>38</b>
Figure 24 OSSEC Log Management with Elasticsearch.....	<b>41</b>
Figure 25 Solution architecture diagram .....	<b>42</b>
Figure 26 Fabric basic Navigation menu.....	<b>44</b>
Figure 27 template files location .....	<b>46</b>
Figure 28 OSSEC rules and policies.....	<b>47</b>
Figure 29 OSSEC alerts from MySQL.....	<b>51</b>
Figure 30 OSSEC event viewer .....	<b>52</b>
Figure 31 OSSEC event history.....	<b>53</b>
Figure 32 OSSEC file integrity .....	<b>54</b>
Figure 33 OSSEC statistics .....	<b>55</b>
Figure 34 Kibana dashboard.....	<b>56</b>
Figure 35 Kibana Json format of event.....	<b>57</b>

Figure 36 Dependencies installation 1 .....	<b>58</b>
Figure 37 Dependencies installation output .....	<b>59</b>
Figure 38 Conpot installation .....	<b>60</b>
Figure 39 Conpot installation output .....	<b>61</b>
Figure 40 open firewall .....	<b>62</b>
Figure 41 open firewall output .....	<b>63</b>
Figure 42 move ossec agent.....	<b>64</b>
Figure 43 move ossec agent output.....	<b>65</b>
Figure 44 deploy ossec agent .....	<b>66</b>
Figure 45 deploy ossec agent output .....	<b>67</b>
Figure 46 ossec-server agent installation.....	<b>68</b>
Figure 47 ossec agent installation .....	<b>69</b>
Figure 48 start conpot by template .....	<b>69</b>
Figure 49 start conpot by template output .....	<b>70</b>
Figure 50 check conpot .....	<b>71</b>
Figure 51 check conpot running output .....	<b>72</b>
Figure 52 Nmap on target .....	<b>73</b>
Figure 53 OSSEC alert on Nmap.....	<b>73</b>
Figure 54 Nmap on 161 port.....	<b>73</b>
Figure 55 findunitid and conpot logs .....	<b>74</b>
Figure 56 Modbus protocol.....	<b>75</b>
Figure 57 successful exploitation output .....	<b>75</b>
Figure 58 unauthorized login and kibana event .....	<b>75</b>
Figure 59 unauthorized login in email alert.....	<b>76</b>
Figure 60 permission denied and kibana event .....	<b>76</b>
Figure 61 permission denied in email alert .....	<b>76</b>
Figure 62 privilege escalation and kibana event .....	<b>77</b>
Figure 63 add user and kibana event.....	<b>77</b>
Figure 64 syslog integrity and ossec gui event.....	<b>77</b>
Figure 65 Host is down and ossec gui event .....	<b>77</b>
Figure 66 Host is down email alert .....	<b>78</b>
Figure 67 OSSEC report from Fabric UI for level 10 alerts.....	<b>78</b>



## Chapter 1 Introduction

In this chapter, we present what inspired and motivated us to embark on this project and we analyze the challenges and problems we dealt with, while implementing architecture in order to develop an integrated stack of technologies in various servers. This architecture can be deployed easily to any infrastructure and includes these major features:

- ✓ Deployment of SCADA honeypots (CONPOT)
- ✓ Procedure automation from a web based user interface (FABRIC)
- ✓ Host intrusion detection system integrated with all systems (OSSEC)
- ✓ ELK stack integrated with HID in order to properly monitor the infrastructure and identify security patterns through policies (ELK)

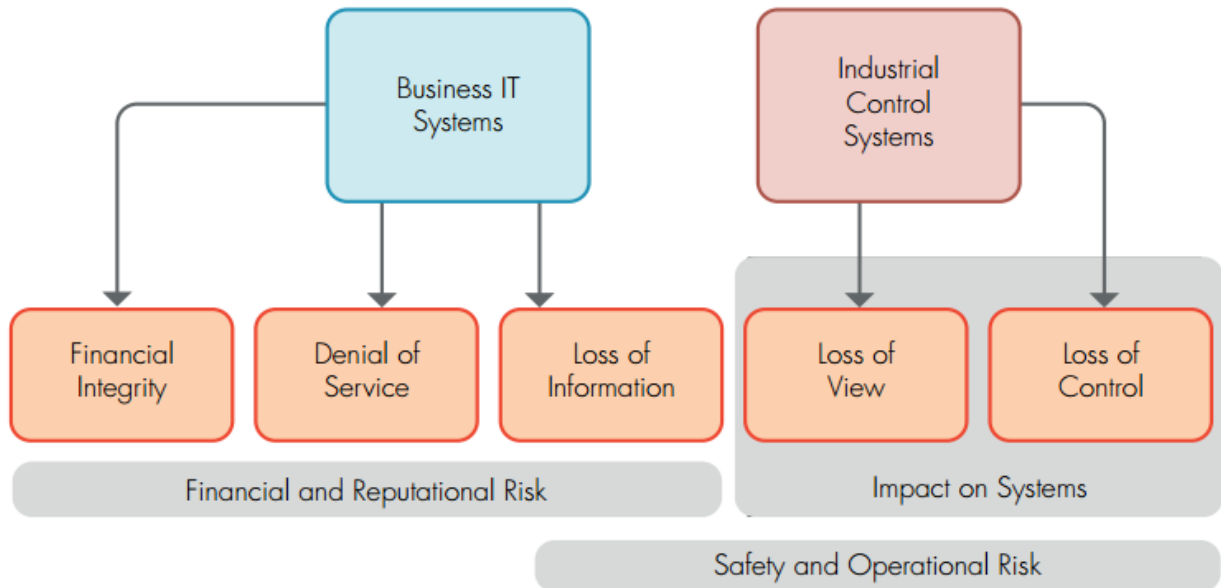
### 1.1 Motivation – Problem description

SCADA is a mainly industrious technology which allows a user to collect data and control a variety of infrastructure and facilities. With this feature it is no necessary for the operator to supervise the components while the operations runs smoothly. SCADA or else Supervisory Control and Data Acquisition is mainly industrial control system. Thus, these systems are computer controlled systems with the ability to monitor and control an industrial process or assembly line or any other facility based process. (Boyer, 2004)

Based on the ("Cyber Security for SCADA Systems,"2013), "Over 1 million SCADA / ICS systems are connected to the internet with unique IPs", "General state of real time system security seen as poor by hackers", and "Honeypot deployment of virtual SCADA proves attackers change settings". Hence the security of SCADA systems in real time represents a real challenge in today's world.

As ("Cyber Security for SCADA Systems,"2013) says. "High profile cyber security threats are a recent phenomenon – think of the Stuxnet or Night Dragon attacks – yet the systems running critical industrial processes are typically a generation older. Consequently, there are many legacy systems that may be vulnerable to cyber-attack because cyber security was simply not a consideration at the time of initial design and installation. The security of even recently deployed systems may also be an issue, and often there are media reports of instances where systems are connected to the internet with inadequate protection, or the manufacturers of the equipment have used hardcoded usernames and passwords, thereby gifting cyber intruders with inside knowledge with the ability to manipulate the system settings."

From the scope of security there are many differences between a real time IT system and an Industrial control system. An organization is more concerned for intellectual property theft, access to financial information or systems integrity. Although these reasons are very serious, Industrial IT systems are not vulnerable, in contrast a compromised SCADA system may lead to loss of control of a plant or a massive destruction that may to disastrous accidents. The following figure gives the whole picture.



**Figure 1 Business System versus ICS Risk**

(Cyber Security for SCADA Systems, 2013)

Our goal in this thesis is to create a centralized solution based on various open source technologies integrated in order to offer an automated way for the deployment of SCADA honeypots. Additionally all these honeypots will be monitored through a web based console, based on security policies. Also all security alarms triggered from agents deployed in honeypot systems will be stored in a database for further analysis and threat pattern recognition. Hence this solution can be used as is from customers offering a unique security solution and threat information gathering stack in order to protect critical IT-Infrastructure SCADA systems against modern cyber-attacks.

## 1.2 Analyzing the need for industrial SCADA security

"Critical Infrastructure' means an asset, system or part thereof located in Member States which is essential for the maintenance of vital societal functions, health, safety, security, economic or social well-being of people, and the disruption or destruction of which would have a significant impact in a Member State as a result of the failure to maintain those functions<sup>11</sup>." (Mattioli & Moulinos, 2015)

SCADA industrial control systems compose the most critical systems infrastructure. These systems are responsible for process control and continuity of national critical functions. The business sectors been controlled by SCADA systems include energy, oil & gas, water or chemical. Thus it is more than obvious that as industries seek more and more process automation and maintenance-free operations the role of a SCADA in the business continuity is very important and essential. (Mattioli & Moulinos, 2015)

Lately the SCADA systems transformed from isolated systems to open architectures and standard technologies. This transformation enhanced these systems with quality and real time information in order operators to make more accurate and better business decisions. As the latest

trend is the computerization of critical manufacturing the result is to connect SCADA and IT environments. Hence as a result could be the increased attack from exposing critical functions to high cyber security risks. The security of SCADA systems is top priority because of the results from the great impact on national critical functions. (Mattioli & Moulinos, 2015)

SCADA cyber-attacks it is known are becoming more and more dangerous and aimed specific control system technologies. Some examples of well-prepared sophisticated attacks dedicated to exploit vulnerabilities of control system are the Aurora vulnerability and Stuxnet attack. (Mattioli & Moulinos, 2015)

### **1.3 Exploiting SCADA systems**

Two main categories of infection exist. A) Access inside SCADA network via a work station computer, B) infection of PLC in order to execute a malicious code. These two cases can be prevented and monitored from the case study of this thesis. Furthermore for SCADA ICS systems stability, assurance of device, longevity and accessibility are essential. Many legacy devices are involved in ICS operation, so knowledge of SCADA vulnerabilities and unpatched operating systems of the IT machines are necessary to understand and manage situations and possible scenarios.

Hence when is possible patches and fixes must be deployed frequently to all elements involves in ICS process otherwise the infrastructure must be monitored with a variety of tools. Moreover it is quite unusual to protect from a zero-day exploit; knowledge is required regarding how ICS interconnects with itself and the rest network infrastructure inside organization alongside with system behavior analysis in order to increase perception. Typically a system like IDS can identify threats but not new ones but a human security analyst may be able to understand unusual traffic from sensors and monitors integrated with cyber and physical elements of ICS. (Colbert & Kott, 2016)

### **1.4 SCADA ICS vulnerabilities**

Here we will list vulnerabilities that may be found in a typical SCADA ICS. Only the main category will be referred alongside with a basic explanation in order to provide an overview of the issues may arise for SCADA systems.

#### ➤ Policy and Procedure Vulnerabilities

"Vulnerabilities are often introduced into ICSs because of incomplete, inappropriate, or nonexistent security documentation, including policy and implementation guides (procedures)." (Stouffe, Falco & Kent, 2006)

Vulnerability	Description
Inadequate security policy for the ICS	Vulnerabilities are often introduced into ICSs due to inadequate policies or the lack of policies specifically for control system security.
No formal ICS security training and awareness program	A documented formal security training and awareness program is designed to keep staff up to date on organizational security policies and procedures as well as industry cyber security standards and best practices. Without training on specific ICS policies and procedures, staff cannot be expected to maintain a secure ICS environment.
Inadequate security architecture and design	Control engineers have historically had no training in security and until relatively recently vendors have not included security features in their products
No specific or documented security procedures were developed from the security policy for the ICS	Specific security procedures should be developed for the ICS. They are the roots of a sound security program.
Absent or deficient ICS equipment implementation guidelines	Equipment implementation guidelines should be kept up to date and readily available. These guidelines are an integral part of security procedures in the event of an ICS malfunction.
Lack of administrative mechanisms for security enforcement	Staff should be held accountable for administering documented security policies and procedures.
Few or no security audits on the ICS	Independent security audits should review and examine a system's records and activities to determine the adequacy of system controls and ensure compliance with established ICS security policy and procedures. Audits should also be used to detect breaches in ICS security services and recommend changes as countermeasures which may include making existing security controls more robust and/or adding new security controls.
No ICS specific continuity of operations or disaster recovery plan (DRP)	A DRP is needed in the event of a major hardware or software failure or destruction of facilities. Lack of a specific DRP for the ICS could lead to extended downtimes.
Lack of ICS specific configuration change management	A process for controlling modifications to hardware, firmware, software, and documentation should be implemented to ensure an ICS is protected against inadequate or improper modifications before, during, and after system implementation. A lack of configuration change management procedures can lead to security oversights, exposures, and risks.

### Figure 2 Policy and Procedure Vulnerabilities

(Stouffe, Falco & Kent, 2006)

➤ Platform Vulnerabilities

"Vulnerabilities in ICSs can occur due to flaws, misconfigurations, or poor maintenance of their platforms, including hardware, operating systems, and ICS applications." (Stouffe, Falco & Kent, 2006)

Vulnerability	Description
OS and vendor software patches may not be developed until significantly after security vulnerabilities are found	Because of the complexity of ICS software and possible modifications to the underlying OS, changes must undergo comprehensive regression testing. The elapsed time for such testing and subsequent distribution of updated software provides a long window of vulnerability
OS and application security patches are not maintained	Out-of-date OSs and applications may contain newly discovered vulnerabilities that could be exploited. Documented procedures should be developed for how security patches will be maintained.
OS and application security patches are implemented without exhaustive testing	OS and application security patches deployed without testing could compromise normal operation of the ICS. Documented procedures should be developed for testing new security patches.
Default configurations are used	Using default configurations often leads to insecure and unnecessary open ports and exploitable services and applications running on hosts.
Critical configurations are not stored or backed up	Procedures should be available for restoring ICS configuration settings in the event of accidental or adversary-initiated configuration changes to maintain system availability and prevent loss of data. Documented procedures should be developed for maintaining ICS configuration settings.
Data unprotected on portable device	If sensitive data (e.g., passwords, dial-up numbers) is stored in the clear on portable devices such as laptops and PDAs and these devices are lost or stolen, system security could be compromised. Policy, procedures, and mechanisms are required for protection.
Lack of adequate password policy	Password policies are needed to define when passwords must be used, how strong they must be, and how they must be maintained. Without a password policy, systems might not have appropriate password controls, making unauthorized access to systems more likely. Password policies should be developed as part of an overall ICS security program taking into account the capabilities of the ICS to handle more complex passwords.
No password used	<p>Passwords should be implemented on ICS components to prevent unauthorized access. Password-related vulnerabilities include having no password for:</p> <ul style="list-style-type: none"> <li>• System login (if the system has user accounts)</li> <li>• System power-on (if the system has no user accounts)</li> <li>• System screen saver (if an ICS component is unattended over time)</li> </ul>

Vulnerability	Description
Password disclosure	<p>Passwords should be kept confidential to prevent unauthorized access. Examples of password disclosures include:</p> <ul style="list-style-type: none"> <li>• Posting passwords in plain sight, local to a system</li> <li>• Sharing passwords to individual user accounts with associates</li> <li>• Communicating passwords to adversaries through social engineering</li> <li>• Sending passwords that are not encrypted through unprotected communications</li> </ul>
Password guessing	<p>Poorly chosen passwords can easily be guessed by humans or computer algorithms to gain unauthorized access. Examples include:</p> <ul style="list-style-type: none"> <li>• Passwords that are short, simple (e.g., all lower-case letters), or otherwise do not meet typical strength requirements. Password strength also depends on the specific ICS capability to handle more stringent passwords</li> <li>• Passwords that are set to the default vendor supplied value</li> <li>• Passwords that are not changed on a specified interval</li> </ul>
Inadequate access controls applied	<p>Poorly specified access controls can result in giving an ICS user too many or too few privileges. The following exemplify each case:</p> <ul style="list-style-type: none"> <li>• System configured with default access control settings gives an operator administrative privileges</li> <li>• System improperly configured results in an operator being unable to take corrective actions in an emergency situation</li> </ul> <p>Access control policies should be developed as part of an ICS security program.</p>

**Figure 3 Platform Configuration Vulnerabilities**

(Stouffe, Falco & Kent, 2006)

Vulnerability	Description
Inadequate testing of security changes	Many ICS facilities, especially smaller facilities, have no test facilities, so security changes must be implemented using the live operational systems
Inadequate physical protection for critical systems	Access to the control center, field devices, portable devices, media, and other ICS components needs to be controlled. Many remote sites are often unstaffed and may not be physically monitored.
Unauthorized personnel have physical access to equipment	Physical access to ICS equipment should be restricted to only the necessary personnel, taking into account safety requirements, such as emergency shutdown or restarts. Improper access to ICS equipment can lead to any of the following: <ul style="list-style-type: none"> <li>Physical theft of data and hardware</li> <li>Physical damage or destruction of data and hardware</li> <li>Unauthorized changes to the functional environment (e.g., data connections, unauthorized use of removable media, adding/removing resources)</li> <li>Disconnection of physical data links</li> <li>Undetectable interception of data (keystroke and other input logging)</li> </ul>
Insecure remote access on ICS components	Modems and other remote access capabilities that enable control engineers and vendors to gain remote access to systems should be deployed with security controls to prevent unauthorized individuals from gaining access to the ICS.
Dual network interface cards (NIC) to connect networks	Machines with dual NICs connected to different networks could allow unauthorized access and passing of data from one network to another.

Vulnerability	Description
Undocumented assets	To properly secure an ICS, there should be an accurate listing of the assets in the system. An inaccurate representation of the control system and its components could leave an unauthorized access point or backdoor into the ICS.
Radio frequency and electro-magnetic pulse (EMP)	The hardware used for control systems is vulnerable to radio frequency electro-magnetic pulses (EMP). The impact can range from temporary disruption of command and control to permanent damage to circuit boards.
Lack of backup power	Without backup power to critical assets, a general loss of power will shut down the ICS and could create an unsafe situation. Loss of power could also lead to insecure default settings.
Loss of environmental control	Loss of environmental control could lead to processors overheating. Some processors will shut down to protect themselves, and some just melt if they overheat.
Lack of redundancy for critical components	Lack of redundancy in critical components could provide single point of failure possibilities

**Figure 4 Platform Hardware Vulnerabilities**

(Stouffe, Falco & Kent, 2006)

Vulnerability	Description
Buffer overflow	Software used to implement an ICS could be vulnerable to buffer overflows; adversaries could exploit these to perform various attacks.
Installed security capabilities not enabled by default	Security capabilities that were installed with the product are useless if they are not enabled or at least identified as being disabled.
Denial of service (DoS)	ICS software could be vulnerable to DoS attacks, resulting in the prevention of authorized access to a system resource or delaying system operations and functions.
Mishandling of undefined, poorly defined, or "illegal" conditions	Some ICS implementations are vulnerable to packets that are malformed or contain illegal or otherwise unexpected field values.
OLE for Process Control (OPC) relies on Remote Procedure Call (RPC) and Distributed Component Object Model (DCOM)	Without updated patches, OPC is vulnerable to the known RPC/DCOM vulnerabilities.
Use of insecure industry-wide ICS protocols	Distributed Network Protocol (DNP) 3.0, Modbus, Profibus, and other protocols are common across several industries and protocol information is freely available. These protocols often have little or no security capabilities.
Use of clear text	Many ICS protocols transmit messages in clear text across the transmission media, making them susceptible to eavesdropping by adversaries.
Unneeded services running	Many platforms have a wide variety of processor and network services defined to operate as a default. Unneeded services are seldom disabled and could be exploited.
Use of proprietary software that has been discussed at conferences and in periodicals	Proprietary software issues are discussed at international ICS conferences (including "Black Hat" conferences) and available through technical papers and periodicals. Also, control system maintenance manuals are available from the vendors. This information can help adversaries to create successful attacks against ICSs.
Inadequate authentication and access control for configuration and programming software	Unauthorized access to configuration and programming software could provide the ability to corrupt a device.

Vulnerability	Description
Intrusion detection/prevention software not installed	Incidents can result in loss of system availability; the capture, modification, and deletion of data; and incorrect execution of control commands. IDS/IPS software may stop or prevent various types of attacks, including DoS attacks, and also identify attacked internal hosts, such as those infected with worms. IDS/IPS software must be tested prior to deployment to determine that it does not compromise normal operation of the ICS.
Logs not maintained	Without proper and accurate logs, it might be impossible to determine what caused a security event to occur.
Incidents are not detected	Where logs and other security sensors are installed, they may not be monitored on a real-time basis and so security incidents may not be rapidly detected and countered.

**Figure 5 Platform Software Vulnerabilities**

(Stouffe, Falco & Kent, 2006)



Vulnerability	Description
Malware protection software not installed	Malicious software can result in performance degradation, loss of system availability, and the capture, modification, or deletion of data. Malware protection software, such as antivirus software, is needed to prevent systems from being infected by malicious software.
Malware protection software or definitions not current	Outdated malware protection software and definitions leave the system open to new malware threats.
Malware protection software implemented without exhaustive testing	Malware protection software deployed without testing could impact normal operation of the ICS.

**Figure 6 Platform Malware Protection Vulnerabilities**

(Stouffe,Falco & Kent, 2006)

➤ Network Vulnerabilities

"Vulnerabilities in ICSs may occur from flaws, misconfigurations, or poor administration of ICS networks and their connections with other networks." (Stouffe,Falco & Kent, 2006)

Vulnerability	Description
Weak network security architecture	The network infrastructure environment within the ICS has often been developed and modified based on business and operational requirements, with little consideration for the potential security impacts of the changes. Over time, security gaps may have been inadvertently introduced within particular portions of the infrastructure. Without remediation, these gaps may represent backdoors into the ICS.
Data flow controls not employed	Data flow controls, such as access control lists (ACL), are needed to restrict which systems can directly access network devices. Generally, only network administrators should be able to access such devices directly. Data flow controls should ensure that other systems cannot directly access the devices.
Poorly configured IT security equipment	Using default configurations often leads to insecure and unnecessary open ports and exploitable network services running on hosts. Improperly configured firewall rules and router ACLs can allow unnecessary traffic.
Network device configurations not stored or backed up	Procedures should be available for restoring network device configuration settings in the event of accidental or adversary-initiated configuration changes to maintain system availability and prevent loss of data. Documented procedures should be developed for maintaining network device configuration settings.
Passwords are not encrypted in transit	Passwords transmitted in clear text across transmission media are susceptible to eavesdropping by adversaries, who could reuse them to gain unauthorized access to a network device. Such access could allow an adversary to disrupt ICS operations or to monitor ICS network activity.
Passwords exist indefinitely on network devices	Passwords should be changed regularly so that if one becomes known by an unauthorized party, the party has unauthorized access to the network device only for a short time. Such access could allow an adversary to disrupt ICS operations or monitor ICS network activity.
Inadequate access controls applied	Unauthorized access to network devices and administrative functions could allow a user to disrupt ICS operations or monitor ICS network activity.

**Figure 7 Network Configuration Vulnerabilities**

(Stouffe,Falco & Kent, 2006)

Vulnerability	Description
Inadequate physical protection of network equipment	Access to network equipment should be controlled to prevent damage or destruction.
Unsecured physical ports	Unsecured universal serial bus (USB) and PS/2 ports could allow unauthorized connection of thumb drives, keystroke loggers, etc.
Loss of environmental control	Loss of environmental control could lead to processors overheating. Some processors will shut down to protect themselves, and some just melt if they overheat.
Non-critical personnel have access to equipment and network connections	Physical access to network equipment should be restricted to only the necessary personnel. Improper access to network equipment can lead to any of the following: <ul style="list-style-type: none"> <li>Physical theft of data and hardware</li> <li>Physical damage or destruction of data and hardware</li> <li>Unauthorized changes to the security environment (e.g., altering ACLs to permit attacks to enter a network)</li> <li>Unauthorized interception and manipulation of network activity</li> <li>Disconnection of physical data links.</li> </ul>

Vulnerability	Description
Control network services not within the control network	Where IT services such as DNS, DHCP are used by control networks, they are often implemented in the IT network, causing the ICS network to become dependent on the IT network that may not have the reliability and availability requirements needed by the ICS
Lack of redundancy for critical networks	Lack of redundancy in critical networks could provide single point of failure possibilities

**Figure 8 Network Hardware Vulnerabilities**

(Stouffe, Falco & Kent, 2006)

Vulnerability	Description
No security perimeter defined	If the control network does not have a perimeter clearly defined, then it is not possible to ensure that the necessary security controls are deployed and configured properly. This can lead to unauthorized access to systems and data, as well as other problems.
Firewalls nonexistent or improperly configured	A lack of properly configured firewalls could permit unnecessary data to pass between networks, such as control and corporate networks. This could cause several problems, including allowing attacks and malware to spread between networks, making sensitive data susceptible to monitoring/eavesdropping on the other network, and providing individuals with unauthorized access to systems.
Control networks used for non-control traffic	Control and non-control traffic have different requirements, such as determinism and reliability, so having both types of traffic on a single network makes it more difficult to configure the network so that it meets the requirements of the control traffic. For example, non-control traffic could inadvertently consume resources that control traffic needs, causing disruptions in ICS functions.

**Figure 9 Network Perimeter Vulnerabilities**

(Stouffe, Falco & Kent, 2006)

Vulnerability	Description
Inadequate firewall and router logs	Without proper and accurate logs, it might be impossible to determine what caused a security incident to occur.
No security monitoring on the ICS network	Without regular security monitoring, incidents might go unnoticed, leading to additional damage and/or disruption. Regular security monitoring is also needed to identify problems with security controls, such as misconfigurations and failures.

**Figure 10 Network Monitoring and Logging Vulnerabilities**

(Stouffe,Falco & Kent, 2006)

Vulnerability	Description
Critical monitoring and control paths are not identified	Rogue and/or unknown connections into the ICS can leave a backdoor for attacks.
Standard, well-documented communication protocols are used in plain text	Adversaries that can monitor the ICS network activity can use a protocol analyzer or other utilities to decode the data transferred by protocols such as telnet, FTP, and NFS. The use of such protocols also makes it easier for adversaries to perform attacks against the ICS and manipulate ICS network activity.
Authentication of users, data or devices is substandard or nonexistent	Many ICS protocols have no authentication at any level. Without authentication, there is the potential to replay, modify, or spoof data or to spoof devices such as sensors and user identities.
Lack of integrity checking for communications	There are no integrity checks built into most industrial protocols; adversaries could manipulate communications undetected. To ensure integrity, the ICS can use lower-layer protocols (e.g., IPsec) that offer data integrity protection.

**Figure 11 Communication Vulnerabilities**

(Stouffe,Falco & Kent, 2006)

Vulnerability	Description
Inadequate authentication between clients and access points	Strong mutual authentication between wireless clients and access points is needed to ensure that clients do not connect to a rogue access point deployed by an adversary, and also to ensure that adversaries do not connect to any of the ICS's wireless networks.
Inadequate data protection between clients and access points	Sensitive data between wireless clients and access points should be protected using strong encryption to ensure that adversaries cannot gain unauthorized access to the unencrypted data.

**Figure 12 Wireless Connection Vulnerabilities**

(Stouffe,Falco & Kent, 2006)

## Chapter 2 Relative Work

### 2.1 Relative papers

Relative projects that have been implemented are few in the SCADA security sector; fewer are those with the greatest resemblance. As such we compared this thesis project with other quite similar technological projects.

From the work of Athar Mahboob and Junaid Ahmed Zubairi (Mahboob,Zubairi, 2013) we can study how we can configure SCADA security via open source tools, very similar to thesis implementation. Hence in this paper we can study SCADA security vulnerabilities alongside with the use of open source tools for security monitoring. They choose the use of OSSIM and honeynet in comparison to this thesis where OSSEC and conpot was the basic open source tools involved. Hence OSSIM is an event processing tool able to perform correlation of events produced amongst other features. This feature could for example provide a triggered event in case multiple password authentication failures within a short period of time. Additionally correlation feature can check whether an event could be correlated with a vulnerability database or vulnerabilities in the assets of the organization. On the contrary OSSEC use a variety of ruled documented in xml files highly customizable in order to manipulate triggered events. In conclusion in this paper of Athar Mahboob and Junaid Ahmed Zubairi, security vulnerabilities and security monitoring tools presented for SCADA systems. However in contrast with this thesis there was lack of automation in the procedures and none of custom implementation alongside with penetration test in order to prove the concept. (Mahboob,Zubairi, 2013)

Furthermore the next paper from Charlie Scott, is trying to improve the security monitoring of SCADA systems. The author chooses to install the conpot honeypot on a facilities network. This is the same decision as in this thesis because conpot use HTTPS, SNMP and Modbus protocol. Also conpot has huge support from community witch is important for maintenance of the honeypot. Moreover in this paper a modification took place in default.xml in order conpot to simulate ION6200. On the contrary in this thesis through Fabric implementation all templates are available for deployment. Also the author used Splunk in order to monitor alerts from conpot log. On the other hand for this thesis an HIDS system like OSSEC has been used in order to monitor through policies the honeypot providing information from events triggered from not only conpot log but from the system itself sending this valuable data for analysis in database. All these data can be processed in order to extract security patterns and train the HIDS system.

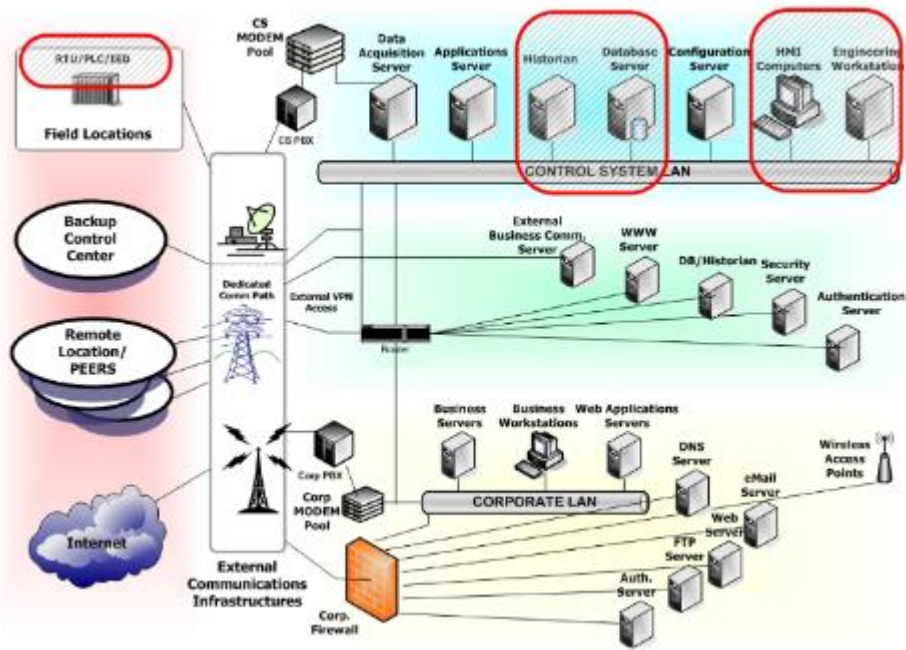
### **Chapter 3 SCADA Security**

There is no simple solution regarding the security of information and critical infrastructure from a variety of cyber-attacks. Defending cyber-attacks is not an easy job and things like risks, probabilities, costs, and mitigation requirements must be a priority. (Miller & Clark , 2015)

There are three main categories of SCADA Systems:

- Modern/Common

“This category of technologies inside the control systems domain will be those that would be most susceptible to modern cyber threats and vulnerabilities, at the same time being mature enough to allow some contemporary forensic methods to be successfully performed on them. Most common technologies that fall into this category include Microsoft Windows, UNIX platform, or an- other vendor specific solution that has functionality that can be investigated using standard forensics methodologies” (15)

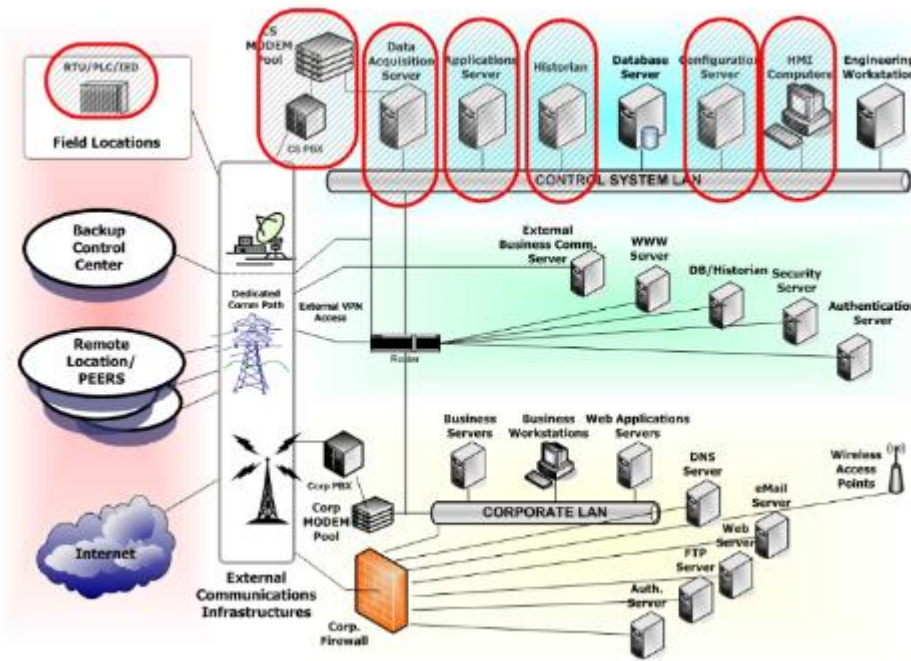


**Figure 13 Modern/Common Diagram**

(Miller & Clark , 2015)

· Modern/Proprietary

“Modern/Proprietary Modern/Proprietary technologies are those that are critical to a control systems operation, have been created within the last 10 years, are still fully supported and understood primarily by the vendor (or systems integrator)” (Miller & Clark , 2015)

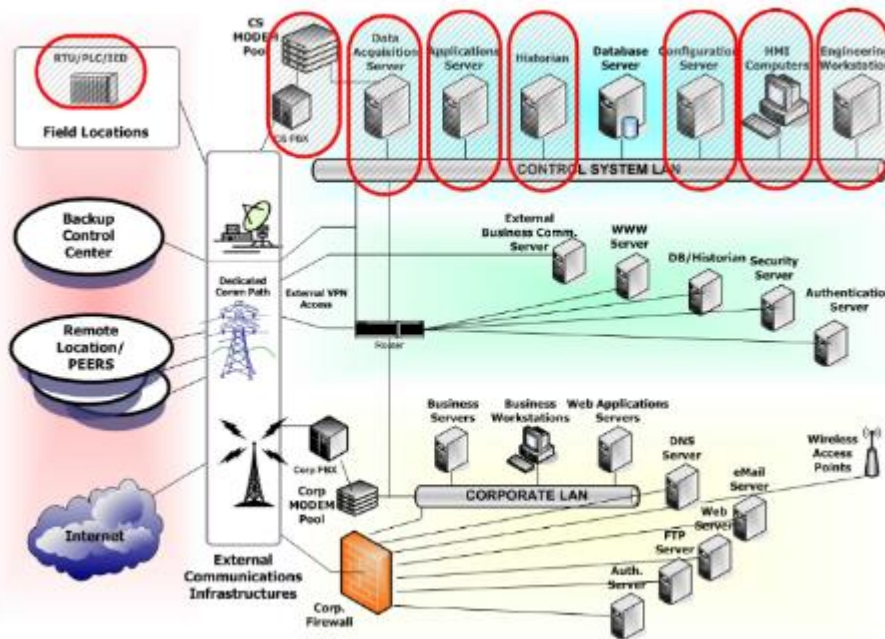


**Figure 14 Modern/Proprietary Diagram**

(Miller & Clark , 2015)

Legacy/Proprietary

"Legacy/Proprietary technologies (Figure 15) are those that are critical to a control systems operation, may have been deployed more than 10 years ago, have moderate computing capabilities (compared to modern systems), may or may not be supported by the vendor (if still around), in most cases only understood (in-depth) by the vendor." (Miller & Clark , 2015)



**Figure 15 Legacy/Proprietary Diagram**

(Miller & Clark , 2015)

SCADA communication protocols responsible for the communication between field and the interface. The basic communication protocol which is been used in this thesis as well in the implementation sector; is the Modbus protocol. Modbus developed by Modicon in 1979 and is the most popular communication protocol for SCADA applications because of its simplicity and ease of use. On the other hand Modbus has many vulnerabilities well known to attackers, like lack of encryption and any other security measures Modbus exposes this protocol to different vulnerabilities. Even though the known issues SCADA protocols like Modbus have a long lifespan and are still being massively deployed and used. (Lehto & Neittaanmäki, 2015)

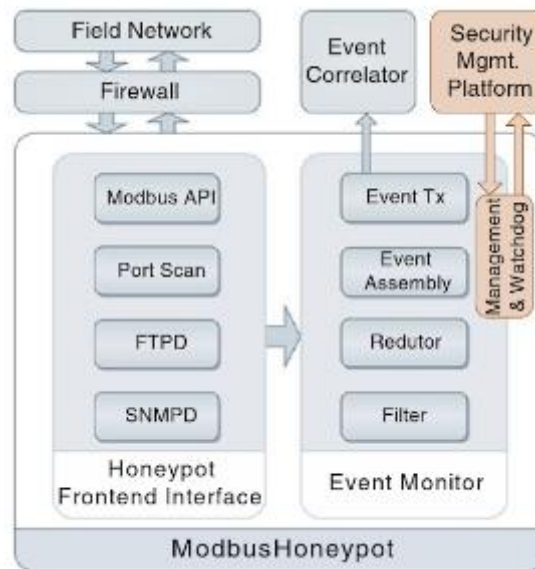
### 3.1 The role of Honeypots in SCADA systems

Honey pot consist of a very important tool in a defense-in-depth architecture. The technology tend to rely upon virtualization for many reasons and modern malware is designed to understand the architecture behind honeypot and either self-destruct or with some built in routines and upon detection will trigger a harmless behavioral signature in order to be undetected from sensors. (Radvanovsky, Brodsky, 2013)

Honeypots can be groups into two categories, research honeypots and production. The former used to obtain information about attack methods while the latter are used to protect ICT infrastructure by providing alarms to operators against critical infrastructure components. (Lehto & Neittaanmäki, 2015)

High interaction honeypots can easily probed, attacked and compromised. Usually the honeypot let the attacker interact with the system so as to capture as much information possible for the intrusion and exploitation techniques. On the other hand low interaction honeypots emulate the vulnerabilities than exposing the real weaknesses not allowing the attacker to interact with the system. Here is an example of a proposed honeypot for monitoring using the Modbus protocol running both

simulated and complete implementation of services running on PLC devices. (Lehto & Neittaanmäki, 2015)



**Figure 16 Modbus honeypot software architecture**

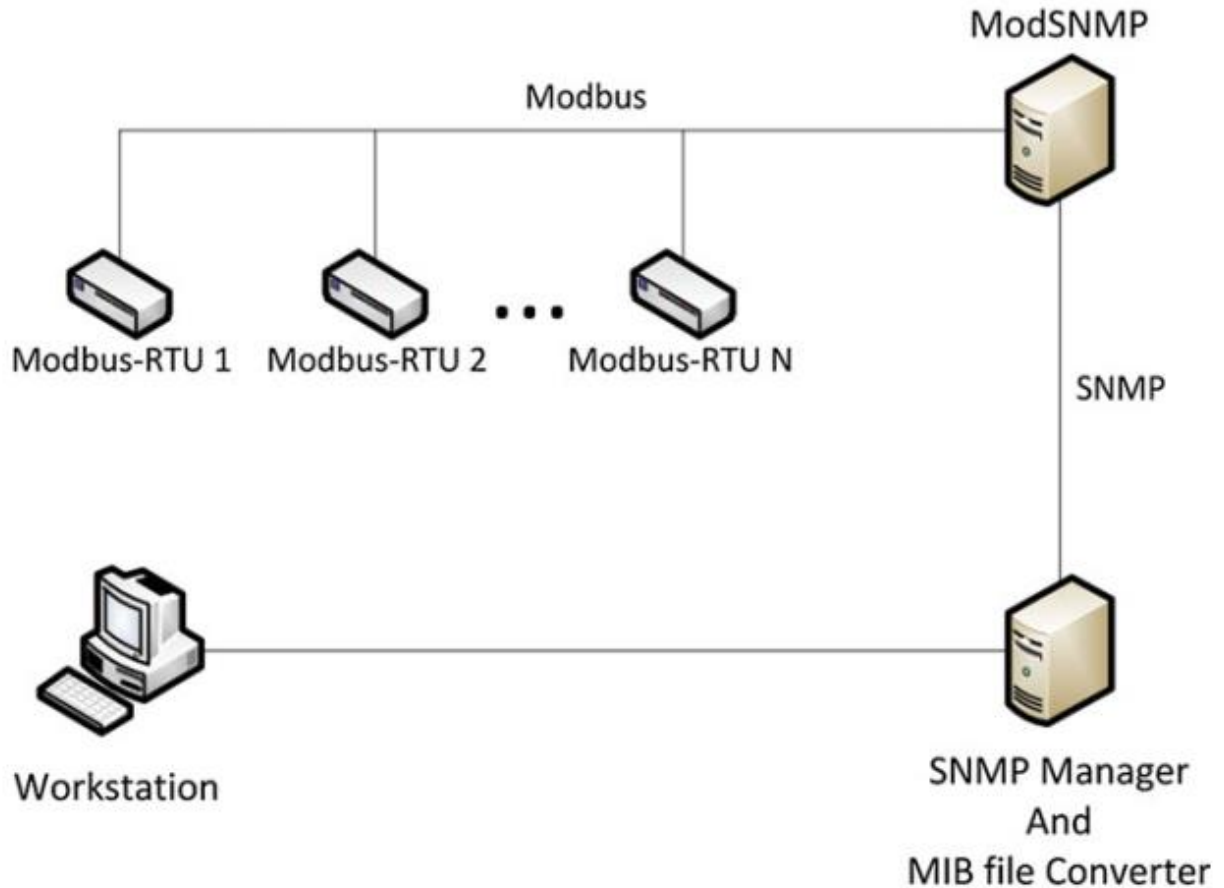
(Lehto & Neittaanmäki, 2015)

### 3.2 The usage of SNMP protocol in SCADA systems

SNMP is wide used in SCADA systems despite the fact that many authors recommend not using SNMP traffic in networks where SCADA systems are deployed in order to secure them. Hence SNMP can manage RTU and PLC devices and also can poll process information from control devices in order to implement SCADA operations. Some examples can be:

“Wingpath, for instance, uses a Modbus to SNMP converter (ModSnm 2013) that acts as a gateway between Modbus slaves and an SNMP manager (see Fig. 4). To poll a value from a given PLC the SNMP manager sends a SNMP GET command to the ModSnm server. This server then sends a Modbus request to the PLC, receives the corresponding response and sends a GETRESPONSE message to the SNMP manager. Other approaches use SNMP traps to poll data from PLCs (Tsubakimoto 2011) Moxa also uses SNMP to gather information from control devices, but instead of the gateway approach of Wingpath and (Tsubakimoto 2011), its control devices are SNMP capable (Moxa 2009). This solution aims at small tasks, such as monitoring Specialized Honeypots for SCADA Systems 263 environmental sensors to measure temperature and humidity, power regulators, or video cameras in server rooms.” (Lehto & Neittaanmäki, 2015)





**Figure 17 Wingpath ModSnm diagram**

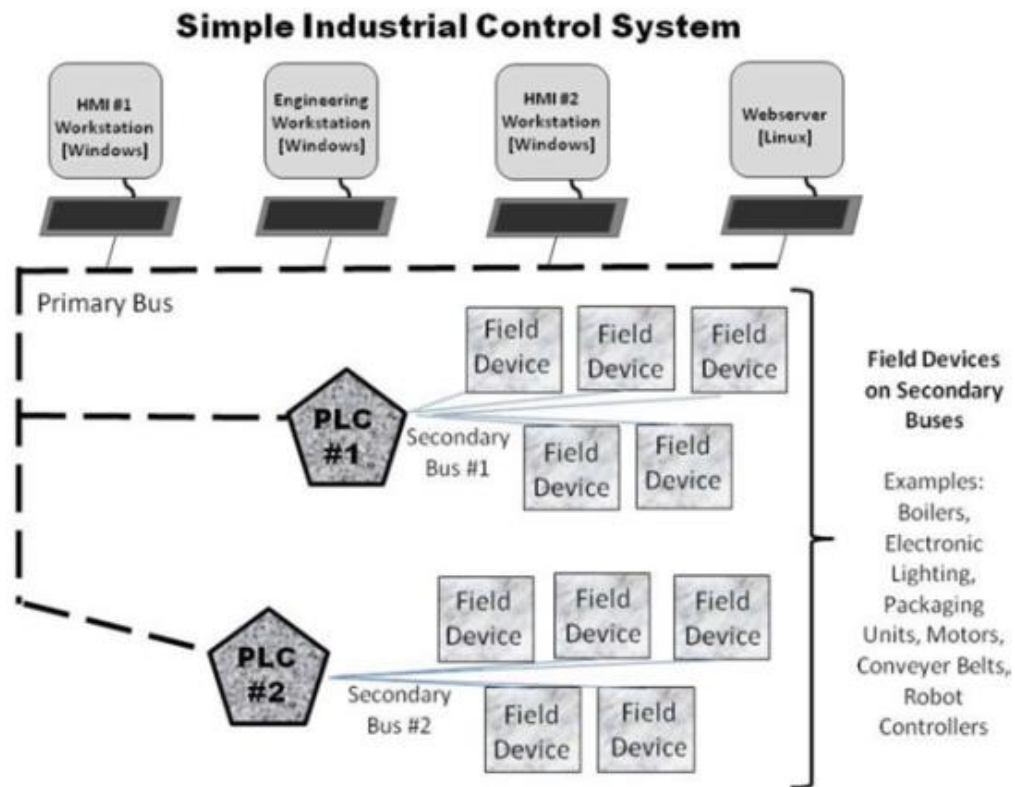
(Lehto & Neittaanmäki, 2015)

Thus in Conpot SNMP agent is included and used to get device, service and health statistics. This protocol runs with UDP on 161 port.

Hence SNMP agent is able to simulate the expected device management functionalities and can be configured to provide full access to industrial PLC information in order to attract potential attackers. (Lehto & Neittaanmäki, 2015)

### 3.3 HIDS for SCADA

A basic description of a SCADA ICS network will be given in order to understand better the architecture.



**Figure 18 Sample of a simple ICS**

(Colbert & Kott, 2016)

The system has two programmable logic controllers, each connected to a standard ICT device network with some workstations. The workstations are typical systems running whether Microsoft or Unix OS. The network is called “Primary Bus” and the network is Packet based. The network also includes Secondary Buses that control field devices. The buses are IP packet based but usually are hard-wired cables with specific voltage or current control needed to run the field devices, this means that are not meant to use protocol such as TCP/IP for the communication. The components of this network are not meant to connect to internet. So in many cases no antivirus is installed because the systems are not supposed to be accessed outside of the network. (Colbert & Kott, 2016)

HIDS were developed for standard ICT computer systems but can be used as well workstations too and other systems. The most important thing is to not interrupt the availability of the systems on the process of collecting HIDS data over the Primary Bus. The HIDS cannot be run directly on PLC or field devices. The basic reason is that the firmware was not manufacture to sun such software. Additionally the devices CPU, memory or bus was not designed to have any other process except PLC processes. (Colbert & Kott, 2016)

“Digital Bond’s Quickdraw adaptation for signature-based methods using Snort was used on ICSs beginning approximately 2008. ICS (and ICT) security researchers however often favor a combination of signature-based ID methods and no signature based ID methods. Starting approximately 2010, ICS ID techniques began to diverge from their ICT Enterprise “parents” and developed unique and useful methodologies that focused more on characterizing the actual ICS

process values, even to the point of creating and monitoring a distinct copy of the ICS plant PLC registers.” (Colbert & Kott, 2016)

### **3.4 Components of a SCADA system**

The most SCADA components are the following:

- ❖ Human–machine interface (HMI), keypad or touch screen.
- ❖ RTUs (remote terminal units).
- ❖ Supervisory systems to process signals and send commands to the units.
- ❖ PLC (programmable logic controllers).
- ❖ Networking systems.
- ❖ Databases and reporting systems.

(Radvanovsky, Brodsky, 2013)

### **3.5 Attacking a SCADA system**

There are some general steps in order to perform a penetration test to an IT system, depending the tools someone use these steps may be integrated together.

(Sandia National Laboratories, 2005)

- 1) Identify the hosts, nodes and networks
- 2) Identify services available on items from 1)
- 3) Identify possible vulnerabilities from items in 2)

For each step specific action came along for the pen tester. Here is a table with the usual list of actions:

Activity	Usual Actions for IT	Preferred Actions for SCADA
Identification of hosts, nodes, and networks	Ping Sweep (e.g. nmap)	<ol style="list-style-type: none"> <li>1. Examine CAM tables on switches.</li> <li>2. Examine router config files or route tables.</li> <li>3. Physical verification (chasing wires).</li> <li>4. Passive listening or IDS (e.g. snort) on network.</li> </ol>
Identification of services	Port Scan (e.g. nmap)	<ol style="list-style-type: none"> <li>1. Local port verification (e.g. netstat).</li> <li>2. Port scan of a duplicate, development, or test system.</li> </ol>
Identification of vulnerabilities within a service	Vulnerability Scan (e.g. nessus, ISS, etc...)	<ol style="list-style-type: none"> <li>1. Local banner grabbing with version lookup in CVE.</li> <li>2. Scan of duplicate, development, or test system.</li> </ol>

**Figure 19 List of activities for pen testing**

(Sandia National Laboratories, 2005)

The SCADA systems are low in resources and unable to defend against attacks. Thus a separation from IT systems and networks is the best way to protect them. Although this seems to be a good security approach is totally unpractical because the data from SCADA systems are very important to business decision on a daily basis. So there must be deployed a secure channel so as to communicate IT systems with SCADA infrastructure. (Sandia National Laboratories, 2005)

### 3.6 SCADA forensics

When SCADA investigation is on for a system for evidence and facts the volatile data are the most important for the case. The volatile data are the most important source of information when network logging is enabled. Usually contains current information about the system, registry, cache and memory. If the attacker changes the password or the organization has forgotten it, it may be usefully to gather as much information is possible from network scanning methods. Usually after an incident the system is powered down, this causes valuable information to be lost. On the contrary SCADA systems cannot go down even if it is known that are compromised. Furthermore, the most common places to look for evidence within volatile data are registers, cache, physical and virtual memory, network connections, processes and disk. Also, external devices must be examined such as floppy, CD-ROM and printers. All data extracted must be removed and analyzed in a different location offline. (Radvanovsky, Brodsky,2013)

Volatile list:

- Registers, cache

- Routing table, address resolution protocol (ARP) cache, process table, kernel statistics
- Memory
- Temporary file systems
- Disk
- Remote logging and monitoring data that is relevant to the system in question
- Physical configuration, network topology
- Archival media

The order must be kept in SCADA systems and all evidence must be captured prior to an event as a routine function. (Radvanovsky, Brodsky,2013)

## Chapter 4 Conpot-0.5.1

“Conpot is a low interactive server side Industrial Control Systems honeypot designed to be easy to deploy, modify and extend. By providing a range of common industrial control protocols we created the basics to build your own system, capable to emulate complex infrastructures to convince an adversary that he just found a huge industrial complex. To improve the deceptive capabilities, we also provided the possibility to server a custom human machine interface to increase the honeypots attack surface. The response times of the services can be artificially delayed to mimic the behavior of a system under constant load. Because we are providing complete stacks of the protocols, Conpot can be accessed with productive HMI's or extended with real hardware. Conpot is developed under the umbrella of the HoneyNet Project and on the shoulders of a couple of very big giants.”

(CONPOT ICS/SCADA Honeypot 2018)

### 4.1 Conpot Installation

For this thesis the following installation process was used from the official Conpot Documentation for release 0.5.1. All these commands are embedded in the Fabric under automation workflow. For the implementation three different operation systems tested and included in the code. For each operating system, Ubuntu 12.04/14.04, Debian and Centos 7.3 details of the installation is below exactly as provided from the official vendor:

### 4.2 Ubuntu 12.04 LTS / 14.04 LTS

#### 1) You need to add multiverse to the source, like:

```
$ sudo vim /etc/apt/sources.list
```

#### 2) Install dependencies:

```
sudo apt-get install libmysqlclient-dev libsmi2ldbl snmp-mibs-downloader python-dev
→libevent-dev \
libxslt1-dev libxml2-dev python-pip python-mysqldb pkg-config libvirt-dev
```

#### 3) The stable version of ConPot can be downloaded from PyPI:

```
pip install conpot
```

The development version can be cloned from github:

```
cd /opt
git clone git@github.com:mushorg/conpot.git
cd conpot
```

```
python setup.py install
(CONPOT ICS/SCADA Honeypot, 2018)
```

### 4.3 CentOS 7.3

**1) Login via ssh with an account with sufficient system privileges (e.g root)**

**2) Upgrade the system**

```
$ sudo yum -y update
```

3) Install needed packages and libs

```
$ sudo yum -y install libxslt-devel libxml2-devel python-pip python
```

```
$ sudo yum -y install mariadb-server mysql-connector-python.noarch mariadb-devel
```

```
$ sudo yum -y install git python-lxml.x86_64 python-devel
```

```
$ sudo yum -y groupinstall "Development tools"
```

```
$ wget https://bootstrap.pypa.io/get-pip.py && sudo python ./get-pip.py
```

```
$ sudo pip install -U lxml
```

3) Start mysql server

```
$ sudo chkconfig mariadb on
```

```
$ sudo service mariadb start
```

```
$ sudo mysql_secure_installation
```

(CONPOT ICS/SCADA Honeypot, 2018)

**4) CONPOT installation**

```
$ git clone https://github.com/mushorg/conpot
```

```
$ cd conpot/
```

```
$ sudo python setup.py install
```

**5) Open ports in firewalld : 80 , 102, 161 and 502**

```
$ firewall-cmd --permanent --add-port=80/tcp
```

```
$ firewall-cmd --permanent --add-port=102/tcp
```

```
$ firewall-cmd --permanent --add-port=161/tcp
```

```
$ firewall-cmd --permanent --add-port=502/tcp
```

```
$ firewall-cmd --reload
```

**6) Start the Conpot honeypot**

```
$ conpot --template default
```

(CONPOT ICS/SCADA Honeypot, 2018)

### 4.4 Debian 7.2.0 64bit & 6.0.7 64bit

**1) Install dependencies:**

```
apt-get install git libsmi2ldbl smistrip libxslt1-dev python-dev libevent-dev
```

Debian 6 specific

```
apt-get install python-pip
```

```
pip install argparse
```

```
wget $package_url
```

```
dpkg -i $package_name or deb http://ftp.nl.debian.org/debian squeeze main non-free
```

```
apt-get update
```

```
apt-get install snmp-mibs-downloader
```

## 2) Conpot Installation

```
pip install conpot
```

OR

The development version can be cloned from github - but we need a modified modbus-tk first.

```
cd /opt
```

```
git clone https://github.com/mushorg/modbus-tk.git
```

```
cd modbus-tk
```

```
python setup.py install
```

```
cd ..
```

```
git clone https://github.com/mushorg/conpot.git
```

```
cd conpot
```

```
python setup.py install
```

(CONPOT ICS/SCADA Honeypot, 2018)

## 4.5 How to use Conpot

After the installation Conpot can be started by the command line as follows:

```

osboxes@osboxes:~$ sudo conpot
[sudo] password for osboxes:

  ____          _ _   _
  |  _ \        | | | | | | | |
  | |_) |       | | | | |
  |  _ <        | | | | |
  | |_| |       | | | | |
  |  ./.        | | | | |
  |_____|       |_| |_| |

Version 0.5.1
MushMush Foundation

-----
Available templates:
-----

--template ipmi
  Unit:          IPMI - 371
  Desc:          Creates a simple IPMI device
  Protocols:    IPMI
  Created by:   Lukas Rist

--template default
  Unit:          Siemens - S7-200
  Desc:          Rough simulation of a basic Siemens S7-200 CPU with 2 slave
  Protocols:    HTTP, MODBUS, s7comm, SNMP
  Created by:   the conpot team

--template proxy
  Unit:          None - Proxy
  Desc:          Sample template that demonstrates the proxy feature.
  Protocols:    Proxy
  Created by:   the conpot team

--template guardian_ast
  Unit:          Guardian - Guardian AST tank-monitoring system
  Desc:          Guardian AST tank-monitoring system
  Protocols:    guardian_ast
  Created by:   the conpot team

--template kamstrup_382
  Unit:          Kamstrup - 382
  Desc:          Register clone of an existing Kamstrup 382 smart meter
  Protocols:    Kamstrup
  Created by:   Johnny Vestergaard

osboxes@osboxes:~$ █

```

**Figure 20 Running Conpot**

(CONPOT ICS/SCADA HoneyPot, 2018)



Depends on the SCADA PLC needs to simulate from the available templates each command should be executed as follows:

- conpot --template ipmi
- conpot --template default
- conpot --template proxy
- conpot --template guardian\_ast
- conpot --template kamstrup\_382

(CONPOT ICS/SCADA Honeypot, 2018)

## Chapter 5 OSSEC

OSSEC is an open source fully integrated platform able to monitor and control systems with various operating systems. Additionally OSSEC works as a host intrusion and detection systems and as a log monitor SIM/SIEM. Furthermore, OSSEC can meet specific compliance requirement like PCI, HIPAA. Also have the ability to detect and alert in case of unauthorized file system alteration and the ability to detect malicious behavior in the log files of other products as well installed in the remote system. From security aspect OSSEC covers the section of file integrity monitoring, log parsing and policy enforcement or control.

### 5.1 OSSEC Features

OSSEC Key benefits:

- ❖ Multi-platform support, i.e Linux, solaris, AIX, HPUX, Mac OS, ESX, Windows
- ❖ Real time alerts and integration with SMTP or SMS and syslog produce alerts
- ❖ Active response options also available
- ❖ Integration with other technologies like SIM/SIEM for centralized reporting and event correlation
- ❖ Centralized management in order to configure policies for all systems.
- ❖ Agent based and agent less monitoring for various OS
- ❖ File integrity checks
- ❖ Log file checking (parsing)
- ❖ Rootkit detection
- ❖ Integration with VMware ESX and a variety of Firewall, switches and routers

(Daniel B. Cid, n.d)

### 5.2 OSSEC Architecture

OSSEC has a central manager to monitor and receives information from agents or agentless for syslog, databases or other custom log files. (OSSEC Project, 2014)

#### 1) Manager

The manager is the centralized deployment of OSSEC. It performs the integrity checking, connectivity with databases, log parsing, and event monitoring. All the xml based rules, decoders and configuration files are stored in the manager in order admin to manage the agents easily from GUI or terminal. (OSSEC Project, 2014)

#### 2) Agent based monitor

The agent is a program installed in the target machine for monitoring purposes. The agent can collect information in real time and send them over to manager for correlation and analysis. Because of small memory and CPU usage it cannot affect remote system performance. (3)

Regarding security can use a non-elevated user during the installation progress and inside chroot jail isolated from system. The agent configuration is pushed from the manager and some configuration is stored in target machine side. (OSSEC Project, 2014)

### 3) Agentless monitor

When an agent installation is not an option for the target machine file integrity be done agentless with a user creation. (OSSEC Project, 2014)

## 5.3 Installation

The OSSEC needs some requirements before installation, so first we install the gcc, libc, apache and php5.

- apt-get install build-essential
- apt-get install apache2 apache2-utils libapache2-mod-php5

(Fadyushin & Popov, 2016)

After that download the OSSEC:

- git clone https://github.com/ossec/ossec-hids
- git clone https://github.com/ossec/ossec-wui

Install the package:

- cd ossec-hids/
- ./install.sh

When the prompt ask about the installation type the option here must be for server! (Fadyushin & Popov, 2016)

Setting up the web server:

- mv ossec-wui\* /var/www/html/ossec-wui
- cd /var/www/html/ossec-wui
- ./setup.sh

And start the OSSEC service:

- service ossec-hids start

Agent installation to target:

1. Select A to add an agent (A).
2. Write the name of our agent.
3. Specify the IP address of our agent.
4. Choose an agent ID. We can leave the ID that OSSEC suggests.
5. "Confirm adding it? (Y / n)" Answer with y.
6. Then, select the E key to extract an agent.
7. Specify the ID of our new agent.
8. Copy the base64 string and press Enter.
9. Select the Q output from the manager to work with agents.

(Fadyushin & Popov, 2016)

- /etc/init.d/ossec restart
- /var/ossec/bin/manage\_agents

In the interactive mode, we need to perform the following steps:

1. Select the I key to import from the server to add a key that we copied.
2. Insert the key, add the agent, and exit.

Then, we can run our agent:

- `/etc/init.d/ossec start`

(Fadyushin & Popov, 2016)

## 5.4 Usage

### Check the status:

- `/var/ossec/bin/ossec-control status`

### Start OSSEC:

- `/var/ossec/bin/ossec-control start`

### Restart OSSEC:

- `/var/ossec/bin/ossec-control restart`

### Backup settings incase you break anything during changes:

- `cp /var/ossec/etc/ossec.conf /var/ossec/etc/ossec.conf_ba`

### Changing settings:

- `nano /var/ossec/etc/ossec.conf`

### Checking the rules for further options:

- `/var/ossec/rules`

### Checking the logs:

- `tail -f /var/ossec/logs/ossec.log`

(Daniel B. Cid, n.d)

## Chapter 6 Fabric automation tool

Fabric is a pythonic tool using 2.5-2.7 python library and basically command line tool for the purpose of streamlining in order to automate via SSH deployment and administration tasks. Specifically Fabric is a tool to execute arbitrary Python functions with the command line prompt. Furthermore Fabric uses a library of subroutines in order to execute Python functions that include tasks or automated jobs or anything that can be included like package installation and copy of files or even ad hoc commands to remote systems. (Fabric Pythonic remote execution, 2014)

### 6.1 Installation

Dependencies:

In order for Fabric's installation to succeed, you will need the following:

- the Python programming language, versions 2.7 or 3.4+;
- the Invoke command-running and task-execution library;
- and the Paramiko SSH library (as well as its own dependencies; see its install docs.)

Fabric tool is very simple to install, this instructions used also for this thesis.

Fabric is best installed via pip:

```
$ pip install fabric
```

Or

```
$ pip install -e git+https://github.com/fabric/fabric
```

Or cloning the Git repository and running:

```
$ pip install -e .
```

(Fabric Pythonic remote execution, 2014)

## 6.2 Usage

Fabric tool uses the fabfile in order to execute automated tasks. Hence all automated tasks must be implemented inside the fabfile.py using an editor; the fabfile must be inside the same directory where fabric tool is deployed.

For example:

For example:

```
# leafpad fabfile.py
```

Edit as follows:

```
#!/usr/bin/env python
```

```
from fabric.api import run, env
```

```
env.hosts = ['host1', 'host2']
```

```
def taskA():
```

```
run('ls')
```

```
def taskB():
```

```
run('whoami')
```

and from the command line execute: `# fab taskA taskB`, in order to execute to remote host the “ls” and “whoami” command.

The fabric will execute the commands as follows:

- ✓ taskA executed on host1
- ✓ taskA executed on host2
- ✓ taskB executed on host1
- ✓ taskB executed on host2

(Fabric Pythonic remote execution, 2014)

## 6.3 Fabric-web

The Fabric-web is tool from <https://github.com/daniellawrence/fabric-web>, which uses the fabric tool inside an html site. However in this thesis this tool has been used widely, but the Fabric-web tool has been customized, in order the user interface to be more friendly than the current.

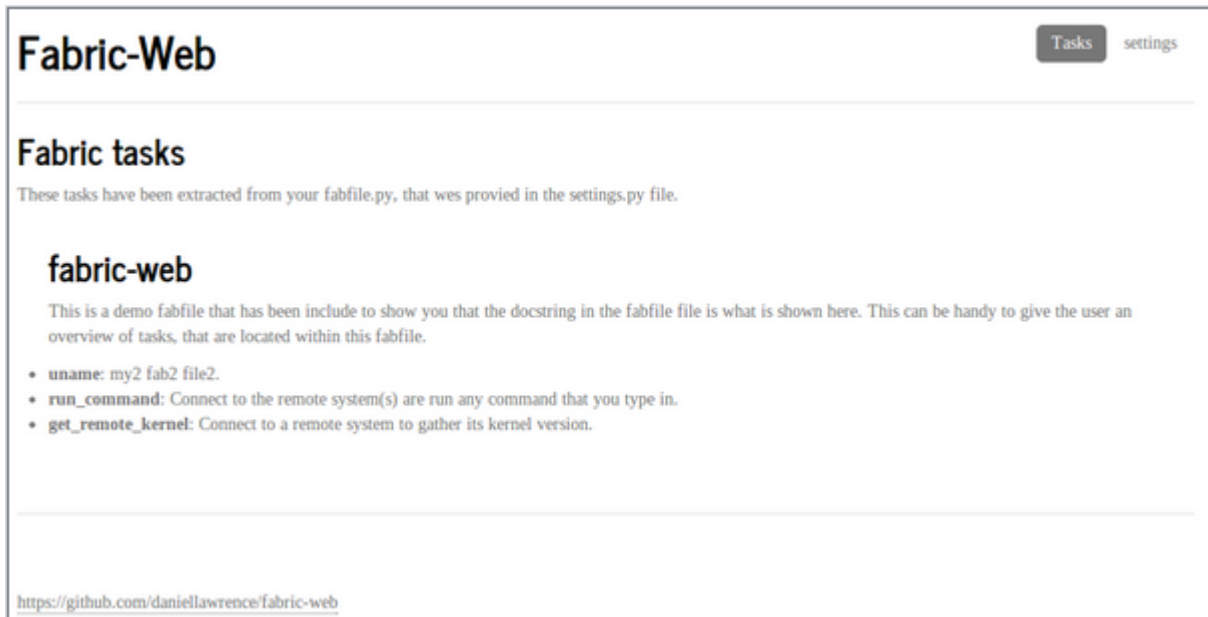
The installation is very easy, can be done via git:

```
$ git clone https://github.com/daniellawrence/fabric-web
```

```
$ pip install -r requirements.txt
```

After the installation the fabric-web is available on <http://localhost:5000/> after executing `./main.py`

The Fabric web use the fabfile, every function inside fabfile compose the Fabric tasks in the browser page as follows:



**Fabric-Web** Tasks settings

---

## Fabric tasks

These tasks have been extracted from your fabfile.py, that was provided in the settings.py file.

### **fabric-web**

This is a demo fabfile that has been include to show you that the docstring in the fabfile file is what is shown here. This can be handy to give the user an overview of tasks, that are located within this fabfile.

- **uname:** my2 fab2 file2.
- **run\_command:** Connect to the remote system(s) are run any command that you type in.
- **get\_remote\_kernel:** Connect to a remote system to gather its kernel version.

---

<https://github.com/daniellawrence/fabric-web>

**Figure 21 Default index page**

(fabric-web, 2018)

The screenshot displays the Fabric-Web interface. At the top left, the title 'Fabric-Web' is shown. To the right, there are two buttons: 'Tasks' and 'settings'. Below the title, a large heading reads 'executed run\_command on dansysadm.com, localhost, nas .'. Underneath, there are three sections, each with a heading and a code block:

- output for nas**:

```
Executing task 'run_command'
run: uptime
out: 19:45:13 up 26 days, 3:01, 1 user, load average: 0.62, 0.45, 0.34
out:
```
- output for dansysadm.com**:

```
Executing task 'run_command'
run: uptime
out: 19:47:28 up 85 days, 21:37, 1 user, load average: 0.65, 0.48, 0.42
out:
```
- output for localhost**:

```
Executing task 'run_command'
run: uptime
out: 19:44:58 up 7:22, 5 users, load average: 0.25, 0.26, 0.27
out:
```

At the bottom, there is a section titled 'CMD' with the text 'Here is the command that we just run, if you want to add it into a script.' and a code block showing the command: `$ fab run_command -H dansysadm.com,localhost,nas`.

**Figure 22 Example of command execution**

(fabric-web, 2018)

## Chapter 7 ELK Stack

For the needs of this thesis ELK stack will be used alongside with OSSEC. ELK stack is consisted of open source technologies, specifically Elasticsearch, Logstash, and Kibana (ELK/Elastic Stack). Elastic has created an end to end stack in order to provide insights in real time from both structured and unstructured data. Also Elastic stack is responsible for searching and analyzing data; in our case integrated with OSSEC as it shows in the following figure. (Elastic Stack, 2018)



**Figure 23 OSSEC Integrated with ELK stack**

(Vic Hargrave, n.d)

## 7.1 Elasticsearch

- “Open source, distributed, full text search engine
- Based on Apache Lucene
- Stores data as structured JSON documents
- Supports single system or multi-node clusters
- Easy to set up and scale – just add more nodes
- Provides a RESTful API
- Installs with RPM or DEB packages and is controlled with a service script”

(Vic Hargrave, n.d)

## 7.2 Logstash

- “Log aggregator and parser
- Supports transferring parsed data directly to Elasticsearch
- Controlled by a configuration file that specifies input, filtering (parsing) and output
- Key to adapting Elasticsearch to other log formats
- Run logstash in logstash home directory as follows: bin/logstash —conf <logstash config file>”

(Vic Hargrave, n.d)

## 7.3 Kibana

- “General purpose query UI
- Javascript implementation
- Query Elasticsearch without coding
- Includes many widgets
- Run Kibana in browser as follows:
- `http://<web server ip>:<port>/<kibana path>`”

(Vic Hargrave, n.d)

## 7.4 Install Elastic

### Prerequisites:

- OS: CentOS 7
- RAM: 4GB
- CPU: 2
- Java 8

### Install Elastic

- 1) import the Elasticsearch public GPG key into rpm:  
`sudo rpm --import http://packages.elastic.co/GPG-KEY-elasticsearch`
- 2) Creation of yum repository:  
`echo '[elasticsearch-2.x]  
name=Elasticsearch repository for 2.x packages  
baseurl=http://packages.elastic.co/elasticsearch/2.x/centos  
gpgcheck=1  
gpgkey=http://packages.elastic.co/GPG-KEY-elasticsearch  
enabled=1  
' | sudo tee /etc/yum.repos.d/elasticsearch.repo`
- 3) Installation:  
`sudo yum -y install elasticsearch`
- 4) Edit configuration file:  
`sudo vi /etc/elasticsearch/elasticsearch.yml`

### Install Kibana

- 1) Create yum repository:  
`sudo vi /etc/yum.repos.d/kibana.repo  
[kibana-4.4]  
name=Kibana repository for 4.4.x packages  
baseurl=http://packages.elastic.co/kibana/4.4/centos  
gpgcheck=1  
gpgkey=http://packages.elastic.co/GPG-KEY-elasticsearch  
enabled=1`
- 2) Install Kibana:  
`sudo yum -y install kibana`
- 3) Edit configuration file:  
`sudo vi /opt/kibana/config/kibana.yml  
server.host: "localhost"`
- 4) Install Nginx:  
`sudo yum -y install epel-release  
sudo yum -y install nginx httpd-tools  
sudo htpasswd -c /etc/nginx/htpasswd.users kibanaadmin  
sudo vi /etc/nginx/nginx.conf  
include /etc/nginx/conf.d/*.conf;  
sudo vi /etc/nginx/conf.d/kibana.conf  
server {  
listen 80;  
  
server_name example.com;  
  
auth_basic "Restricted Access";  
auth_basic_user_file /etc/nginx/htpasswd.users;`



```

        location / {
            proxy_pass http://localhost:5601;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection 'upgrade';
            proxy_set_header Host $host;
            proxy_cache_bypass $http_upgrade;
        }
    }
    sudo systemctl start nginx
    sudo systemctl enable nginx

```

#### Install Logstash

- 1) Add yum repository for Logstash:
 

```

sudo vi /etc/yum.repos.d/logstash.repo
[logstash-2.2]
name=logstash repository for 2.2 packages
baseurl=http://packages.elasticsearch.org/logstash/2.2/centos
gpgcheck=1
gpgkey=http://packages.elasticsearch.org/GPG-KEY-elasticsearch
enabled=1

```
- 2) Install logstash:
 

```

sudo yum -y install logstash

```
- 3) Configure logstash:
 

```

sudo vi /etc/logstash/conf.d/02-beats-input.conf
input {
    beats {
        port => 5044
        ssl => true
        ssl_certificate => "/etc/pki/tls/certs/logstash-forwarder.crt"
        ssl_key => "/etc/pki/tls/private/logstash-forwarder.key"
    }
}

sudo vi /etc/logstash/conf.d/10-syslog-filter.conf
filter {
    if [type] == "syslog" {
        grok {
            match => { "message" => "%{SYSLOGTIMESTAMP:syslog_timestamp}
%{SYSLOGHOST:syslog_hostname}
%{DATA:syslog_program}(?:[%%{POSINT:syslog_pid}])?:"
%{GREEDYDATA:syslog_message}" }
            add_field => [ "received_at", "%{@timestamp}" ]
            add_field => [ "received_from", "%{host}" ]
        }
        syslog_pri { }
        date {
            match => [ "syslog_timestamp", "MMM d HH:mm:ss", "MMM dd HH:mm:ss" ]
        }
    }
}

sudo vi /etc/logstash/conf.d/30-elasticsearch-output.conf
output {
    elasticsearch {

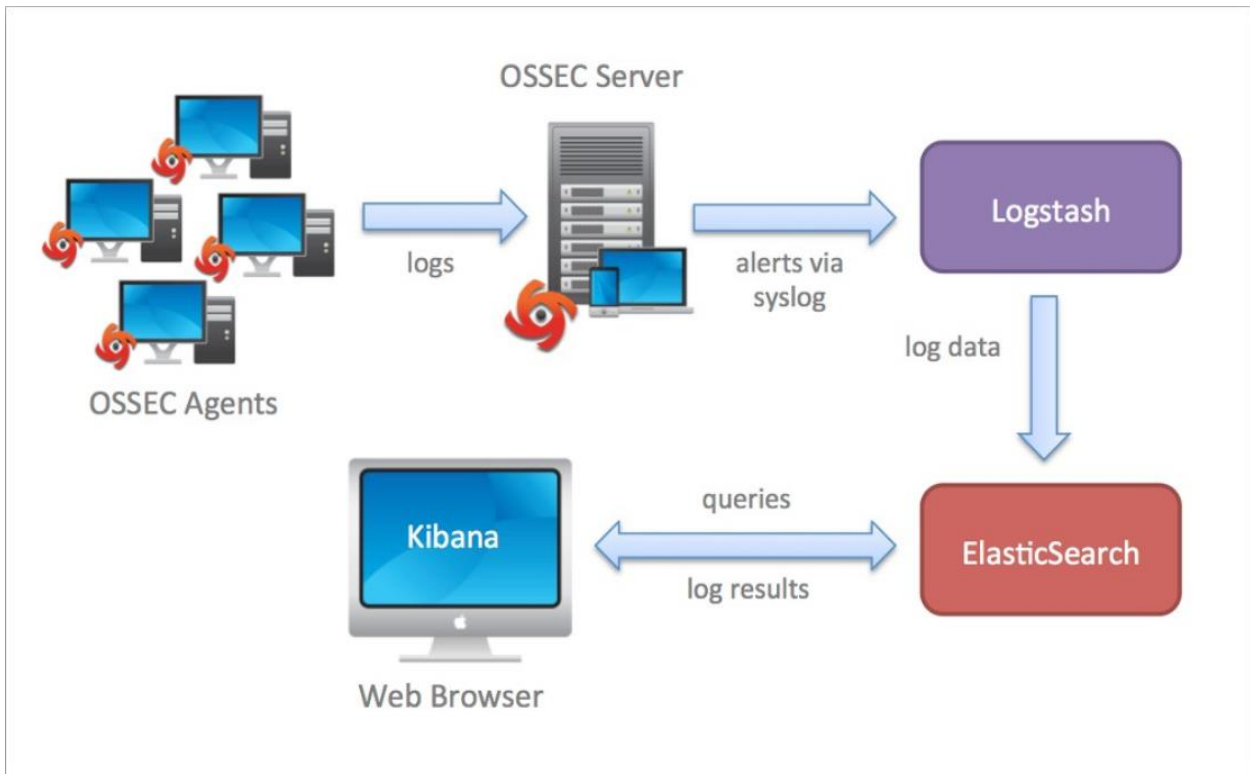
```

```

hosts => ["localhost:9200"]
sniffing => true
manage_template => false
index => "%{[@metadata][beat]}-%{+YYYY.MM.dd}"
document_type => "%{[@metadata][type]}"
}
}
sudo service logstash configtest

```

(How To Install Elasticsearch, Logstash, and Kibana, 2018)



**Figure 24 OSSEC Log Management with Elasticsearch**

(Vic Hargrave, n.d)

## 7.5 Usage

With this commands we control the services for the Elastic stack:

```
service elasticsearch start
```

```
service logstash start
```

```
service kibana start
```

## Chapter 8 Implementation

The implementation of the architecture below describes a system, able to create through automation the installation deployment and monitoring of network consisted of one or more SCADA honeypots.

## 8.1 Architecture - Design

The basic architecture diagram for the implementation of this thesis is the following:

Conpot SCADA Honeypot Deployment and Monitoring

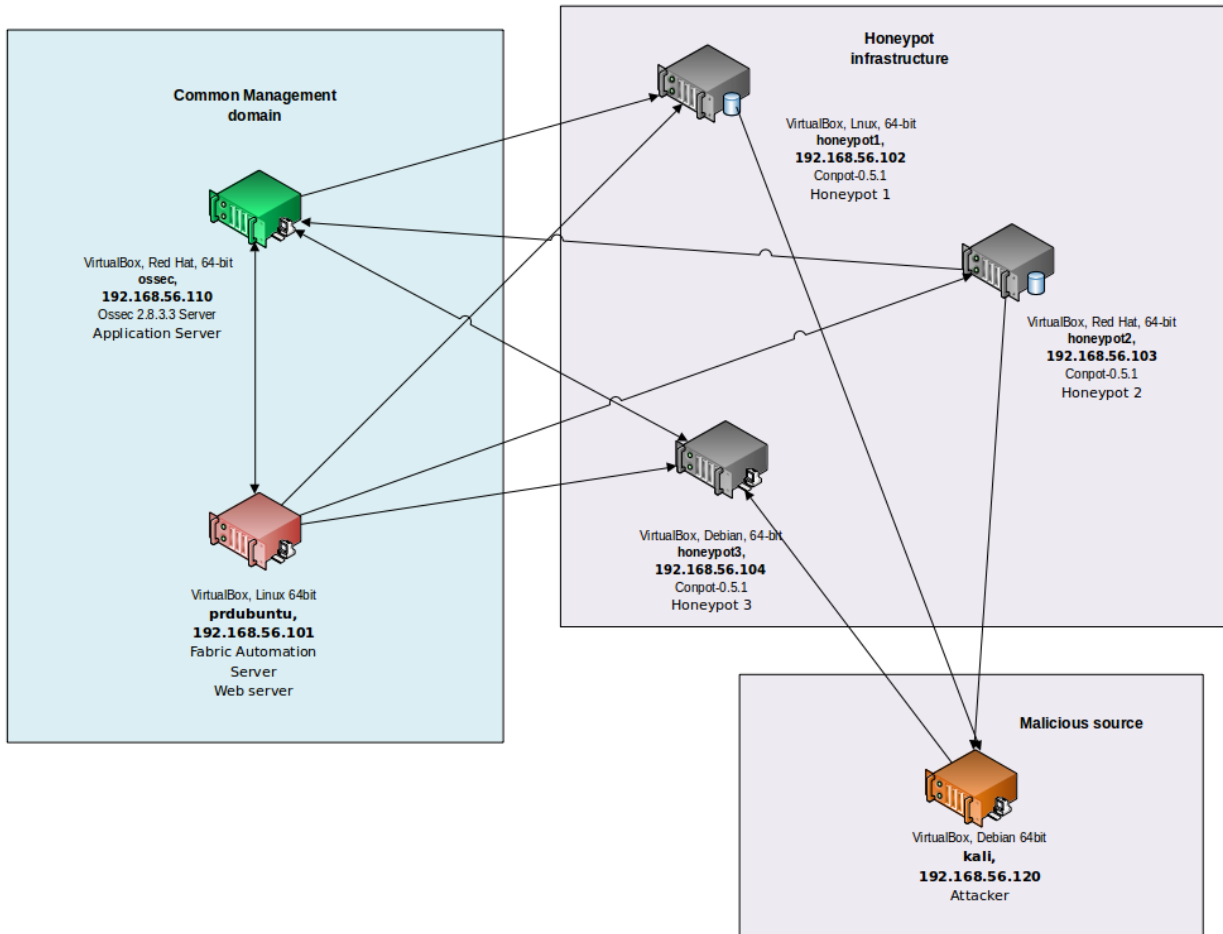


Figure 25 Solution architecture diagram

## 8.2 Software

- 1) Fabric: A Python (2.5-2.7) library and command-line tool
- 2) Ossec 2.9: OSSEC is a scalable, multi-platform, open source Host-based Intrusion Detection System (HIDS)
- 3) CONPOT ICS/SCADA Honeypot: Conpot is a low interactive server side Industrial Control Systems honeypot designed to be easy to deploy, modify and extend.

## 8.3 Virtual machines OS

### 1) VirtualBox, Red Hat, 64-bit

ossec, 192.168.56.110

Ossec 2.8.3.3 Server

Application Server

### 2) VirtualBox, Linux 64bit

prubuntu, 192.168.56.102 Fabric Automation Server

Web server

### 3) VirtualBox, Lnx, 64-bit

honeypot1, 192.168.56.101

Conpot-0.5.1

Honeypot

### 4) VirtualBox, Red Hat, 64-bit

honeypot2, 192.168.56.103

Conpot-0.5.1

Honeypot

### 5) VirtualBox, Debian, 64-bit

honeypot3, 192.168.56.104

Conpot-0.5.1

Honeypot

### 6) VirtualBox, Debian 64bit

kali, 192.168.56.120 Attacker

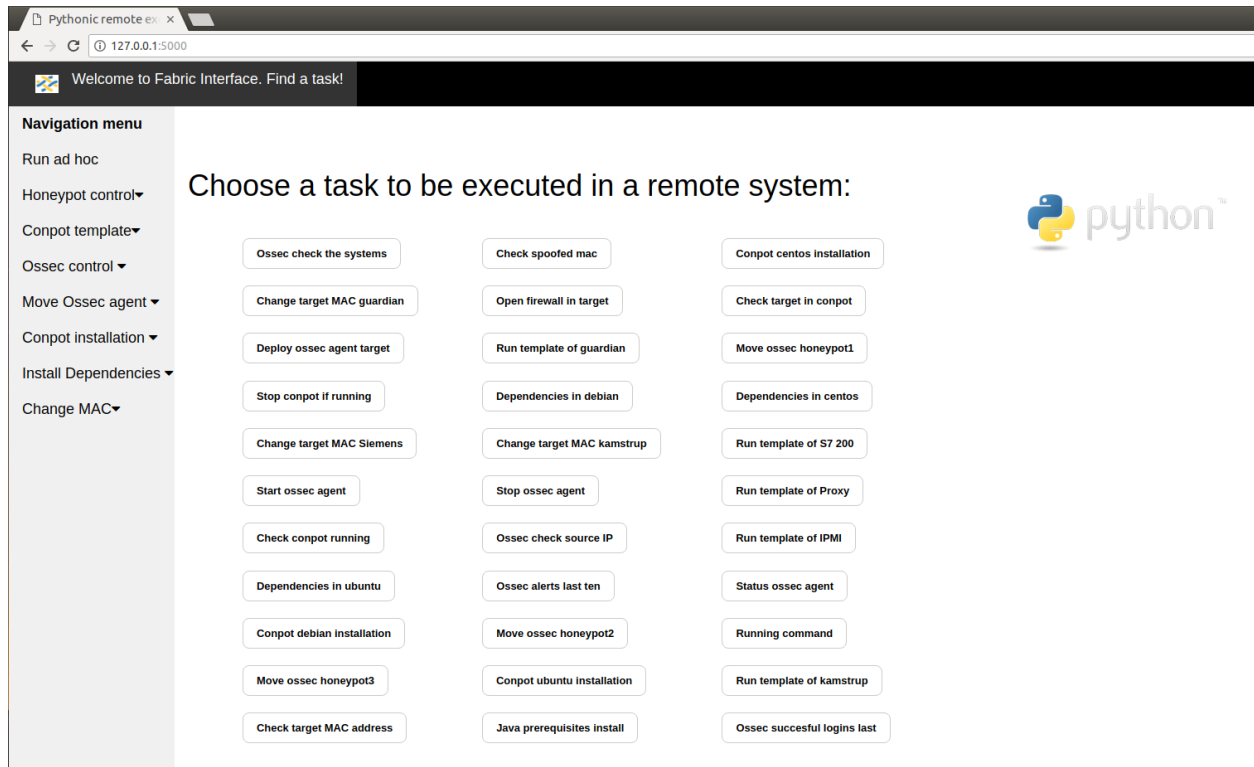
## 8.4 Dependencies

- ❖ All servers belong to an internal network without firewall or iptable rules inside machines
- ❖ We need two interfaces for each node, one for NAT connection with internet in order to update packages necessary for installation and deployment and one interface for LAN network connection. For each Virtual machine that host honeypot another LAN interface has been assigned for the simulation of a real network, in order to have another internal connection with FABRIC server when MAC address spoofing is enabled through the CONPOT software. Although a specific MAC address of a PLC used inside virtualbox configuration of network interfaces at the beginning. This setting can be used in a template before cloning VMs inside a network.
- ❖ OS versions: honeypots: CentOS installation 7.3, Debian 7.2.0 64bit/6.0.7 64bit, Ubuntu 12.04 LTS / 14.04 LTS, application server (ossec): CentOS centos-release-6-7.el6.centos.12.3.x86\_64, webserver (fabric): Ubuntu server 16.04

## 8.5 Workflow - Customization

To begin with an internal network created without firewall between nodes as explained in the architecture diagram in the beginning of this chapter based on the dependencies mentioned. Nevertheless it is absolutely necessary to create two interfaces, one with access to internet so as packages can be downloaded. The implementation of the virtual machines is done with virtual box with the use of NAT and internal network option to adapters. However, it is important to mention for this thesis, only open source tools have been used in all phases of the implementation. Moreover password-less login enabled for all virtual machines with root privileged user.

After the creation of virtual machines deployment of tools started on the web server with fabric and additionally with fabric – web gui. After that the creation of many python functions inside fabfile.py took place. Thus inside fabfile.py credentials set alongside with IPs and hostnames for the targets. In our case there no available DNS or local hostnames in hosts file so use of ip address is the option. Here is the explanation of functions set inside fabfile.py, the whole code of fabfile.py is available in Apendix-A.



**Figure 26 Fabric basic Navigation menu**

- ❖ Run\_template\_of\_S7\_200():  
 """ Rough simulation of a basic Siemens S7-200 CPU with 2 slaves. Protocols: HTTP, MODBUS, s7comm, SNMP """
- ❖ Stop\_conpot\_if\_running():  
 """ With this method conpot honeypot can be terminated manually by user """
- ❖ Run\_template\_of\_kamstrup():  
 """ Register clone of an existing Kamstrup 382 smart meter. Protocols: Kamstrup """
- ❖ Run\_template\_of\_guardian():  
 """ Guardian AST tank-monitoring system. Protocols: guardian\_ast """
- ❖ Run\_template\_of\_Proxy():  
 """ Sample template that demonstrates the proxy feature. Protocols: Proxy """
- ❖ Run\_template\_of\_IPMI():  
 """ Creates a simple IPMI device. Protocols: IPMI """
- ❖ Dependencies\_in\_centos():  
 """ Install the required dependencies for CentOS version 7.3-1611 & Conpot 0.5.1 """
- ❖ Conpot\_centos\_installation():  
 """ Install Conpot 0.5.1 to CentOS """
- ❖ Dependencies\_in\_ubuntu():

```

""" Install the required dependencies for Ubuntu 12.04 LTS / 14.04 LTS / 16.04 LTS """
❖ Conpot_ubuntu_installation():
""" Install the stable version of ConPot """
❖ Dependencies_in_debian():
""" Install the required dependencies for Debian 6.0.7 64bit & 7.2.0 64bit """
❖ Conpot_debian_installation():
""" Install the development version of ConPot """
❖ Running_command(executable_command):
"""Connect to the remote system(s) are run any command that you type in. You can also choose
conpot template manually:
conpot --template ault
conpot --template kamstrup_382
conpot --template guardian_ast
conpot --template proxy
conpot --template ipmi """
❖ Move_ossec_honeypot1():
""" Put ossec agent software in target """
❖ Move_ossec_honeypot2():
""" Put ossec agent software in target """
❖ Move_ossec_honeypot3():
""" Put ossec agent software in target """
❖ Deploy_ossec_agent_target():
""" Deploy agent conpot software in target """
❖ Ossec_succesful_logins_last():
""" This is high level ad-hoc reporting ossec utility. Show Successful Logins. 'This is a high
level report for further details look at: /var/ossec/logs/alerts/alerts.log or contact ossec
administrator!' """
❖ Ossec_alerts_last_ten():
""" This is high level ad-hoc reporting ossec utility. Show Alerts Level 10 and Greater. 'This
is a high level report for further details look at: /var/ossec/logs/alerts/alerts.log or contact
ossec administrator!' """
❖ Ossec_check_source_IP():
""" This is high level ad-hoc reporting ossec utility. Show the srcip for all users. 'This is a
high level report for further details look at: /var/ossec/logs/alerts/alerts.log or contact ossec
administrator!' """
❖ Ossec_check_the_systems():
""" This is high level ad-hoc reporting ossec utility. Show Changed files as reported by
Syscheck. 'This is a high level report for further details look at:
/var/ossec/logs/alerts/alerts.log or contact ossec administrator!' """
❖ Java_prerequisites_install():
""" This is method installs JRE and JDK """
❖ Change_target_MAC_Siemens():
""" This is method change the MAC address from the target node to imitate Siemens PLC
"""
❖ Change_target_MAC_kamstrup():
""" This is method change the MAC address from the target node to imitate kamstrup PLC
"""
❖ Change_target_MAC_guardian():
""" This is method change the MAC address from the target node to imitate guardian PLC
"""

```

- ❖ Check\_target\_MAC\_address():  
""" This is method returns target MAC address """
- ❖ Start\_ossec\_agent():  
""" This is method starts ossec agent """
- ❖ Stop\_ossec\_agent():  
""" This is method stops ossec agent """
- ❖ Status\_ossec\_agent():  
""" This is the status of ossec agent """
- ❖ Check\_spoofed\_mac():  
""" This is the status of ossec agent """

At this moment with the instructions of the installation process of Fabric tool alongside with the Fabric-Web the webserver who is responsible for the automation process of the whole workflow is ready. Some changes implemented on the Fabric-Web in order the user interface to be friendlier. The changes implemented in the templates files:

```
opcadm@prdubuntu:~/fabric-web/templates$ ls
base.html execute.html index.html task_form.html
opcadm@prdubuntu:~/fabric-web/templates$
```

### Figure 27 template files location

Specifically modifications made inside base.html and index.html. Both files code is available in the Appendix B.

After the webserver installation, configuration and dependencies installation; some necessary modifications were made inside Conpot package in order to fix some issues. The Conpot package has been copied inside webserver in order Fabric to copy from there and deploy it in every target.

Conpot modifications:

1)

Inside /usr/local/lib/python2.7/dist-packages/conpot/conpot.cfg, these rules modified:

```
[fetch_public_ip]
```

```
enabled = False
```

```
urls = ["http://www.telize.com/ip", "http://queryip.net/ip/", "http://ifconfig.me/ip"]
```

These urls are no longer available.

```
[change_mac_addr]
```

```
enabled = False
```

```
iface = eth0
```

```
addr = 00:1c:06:03:61:68
```

This feature enables MAC spoofing; in our case we change the MAC address from the VirtualBox configuration.

2)

Because Conpot Modbus server by default works on serial mode the current Conpot version cannot handle the UID in a proper way. So when there is no modbus device (PLC) behind a UID the result should be something like no response and not device failure. This bug found in the section of penetration testing to Conpot with Metasploit while the test returns SLAVE DEVICE FAILURE giving error when these requests should be ignored when there is no slave behind a UID.

(Return no response when there is no slave behind a UID, 2018)

This specific issue solved with the use of this code inside /usr/local/lib/python2.7/dist-packages/conpot/protocols/modbus/slave\_db.py. The whole code is available in the Appendix A.

Moreover installation of OSSEC v2.8.3 took part in a centos machine as described in a previous chapter. This machine is responsible to monitor target logs alongside with Conpot log file and for host intrusion detection.

The agent needed to communicate with OSSEC server is deployed through Fabric automation tool. Although for the agent installation to be complete keys must be exchanged manually between server and agent as follows:

**server side**

```
/var/ossec/bin/manage_agents copy key
```

**client side – agent**

```
/var/ossec/bin/manage_agents
```

```
import key
```

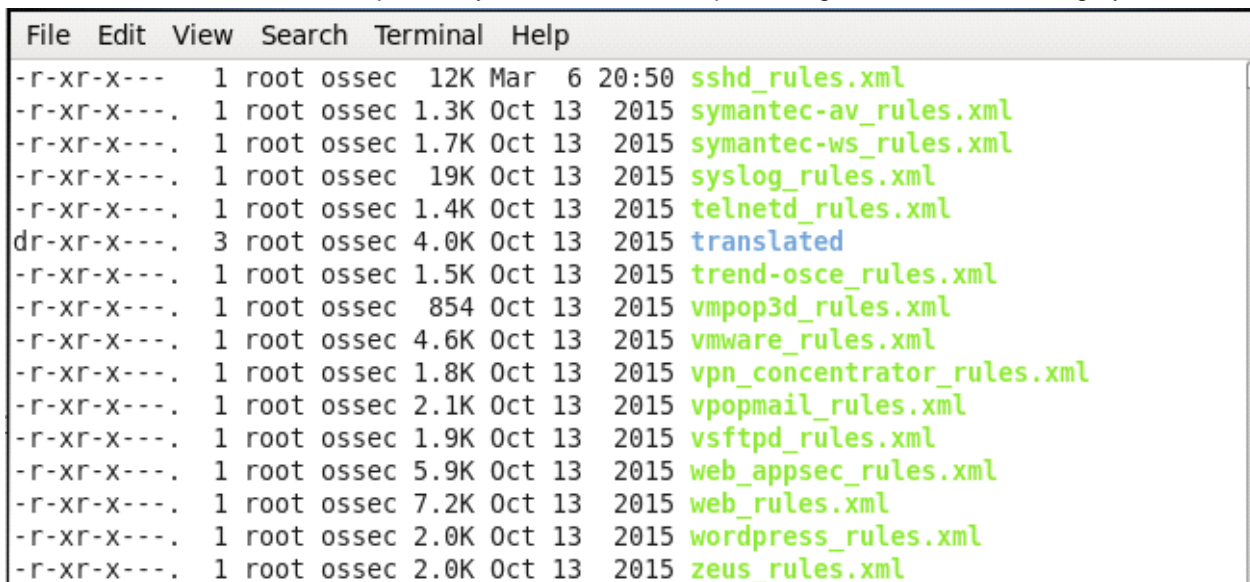
**restart both client and server**

```
/var/ossec/bin/ossec-control restart
```

OSSEC has two different methods to send notifications. Former is to create an alert based on an event generating in OSSEC server. Every event is defined in an xml rule, there we can set alert level and how and when an email alert will be generated or the alert will come only in the console, In our case KIBANA. For this implementation a postfix mail server installed in order to send alerts with email notification. Also daily reports can be executed daily and sent to a specific email address, these are high level aggregated reports for the number and categories of events to the console.

Latter we can generate ad hoc reports anytime for the target in our case the honeypot. In our case these ad hoc execution of reports can be done from Fabric web based user interface.

Furthermore OSSEC has a great variety of policies and rules for host based attacks not to mention also the variety of rules for the generation of alerts. However the OSSEC is highly customizable and the creation of custom rules is quite easy in order to check specific logs or to check file integrity.



File	Edit	View	Search	Terminal	Help
-r-xr-x---	1	root	ossec	12K Mar 6 20:50	sshd_rules.xml
-r-xr-x---	1	root	ossec	1.3K Oct 13 2015	symantec-av_rules.xml
-r-xr-x---	1	root	ossec	1.7K Oct 13 2015	symantec-ws_rules.xml
-r-xr-x---	1	root	ossec	19K Oct 13 2015	syslog_rules.xml
-r-xr-x---	1	root	ossec	1.4K Oct 13 2015	telnetd_rules.xml
dr-xr-x---	3	root	ossec	4.0K Oct 13 2015	translated
-r-xr-x---	1	root	ossec	1.5K Oct 13 2015	trend-osce_rules.xml
-r-xr-x---	1	root	ossec	854 Oct 13 2015	vmpop3d_rules.xml
-r-xr-x---	1	root	ossec	4.6K Oct 13 2015	vmware_rules.xml
-r-xr-x---	1	root	ossec	1.8K Oct 13 2015	vpn_concentrator_rules.xml
-r-xr-x---	1	root	ossec	2.1K Oct 13 2015	vpopmail_rules.xml
-r-xr-x---	1	root	ossec	1.9K Oct 13 2015	vsftpd_rules.xml
-r-xr-x---	1	root	ossec	5.9K Oct 13 2015	web_appsec_rules.xml
-r-xr-x---	1	root	ossec	7.2K Oct 13 2015	web_rules.xml
-r-xr-x---	1	root	ossec	2.0K Oct 13 2015	wordpress_rules.xml
-r-xr-x---	1	root	ossec	2.0K Oct 13 2015	zeus_rules.xml

**Figure 28 OSSEC rules and policies**

These rules have been modified in order alerts and emails to be triggered:



## 1. unauthorized login (Attempting to login as default users)

```
<rule id="5710"
level="5"> <if_sid>5700</if_sid> <options>alert_by_email</options> <match>illegal
user|invalid user</match> <description>Attempt to login using a non-existent user.Contact ossec
admin!</description> <group>invalid_login,authentication_failed,</group> </rule>
```

## 2. privilege escalation (su to root)

```
<rule id="5303"
level="3"> <if_sid>5300</if_sid> <options>alert_by_email</options> <regex>session opened
for user root|^su root|^</regex> <regex>^+ \S+ \S+|proot$|^S+ to root on|^SU \S+ \S+ + \S+ \S+
root$</regex> <description>User successfully changed UID to root.Contact ossec
administrator</description> <group>authentication_success,</group> </rule>
```

## 3. Add user

```
<rule id="5901" level="8"> <options>alert_by_email</options> <match>^new
group</match> <description>New group added to the system.Contact ossec
administrator</description> </rule>
```

## 4. syslog integrity

```
<rule id="1001" level="2"> <options>alert_by_email</options> <match>^Couldn't open
/etc/securetty</match> <description>File missing. Root access unrestricted.Contact ossec
administrator</description> </rule>
```

## 5. Host is down (network error or attack)

```
<rule id="5725"
level="0"> <if_sid>5700</if_sid> <options>alert_by_email</options> <match>fatal: Write
failed: Host is down.Contact ossec administrator</match> <description>Host ungracefully
disconnected.</description> </rule>
```

And this file `/var/ossec/etc/ossec.conf`, has been modified as well in order to set the email sender and receiver:

```
<global> <email_notification>yes</email_notification> <email_to>alerts.ossec@gmail.com</em
ail_to> <smtp_server>localhost</smtp_server> <email_from>user@ossec.com</email_from> <
/global>
```

So the local mail server can send email in the address we desire. Although we can set different email recipients for a variety of categories, in our case we do not have a valid domain name and we need to whitelist the domain inside email provider.

However in order a rule to send an email alert the rule xml must be configured as well like the following:

```
<rule id="5710"
level="5"> <if_sid>5700</if_sid> <options>alert_by_email</options> <match>illegal
user|invalid user</match> <description>Attempt to login using a non-existent user.Contact
ossecadmin!</description> <group>invalid_login,authentication_failed,</group> </rule>in <desc
ription></description> you can add text custom
```

The last customization made in the OSSEC machine was to prepare the environment in order to install a MySQL database in order to forward all events there for future processing. These actions implemented in order to achieve MySQL integration with OSSEC.

- Updating OSSEC to include MySQL capability

### Pre configuration

```
1 # cd ossec-hids-*
2 # cd src; make setdb; cd ..
3 # ./install.sh
```

```
4 /var/ossec/bin/ossec-control enable database
```

### Configuration

```
netstat -lp | grep 3306
lsof -i TCP:3306
whereis mysql
mysql: /usr/lib64/mysql /usr/share/mysql /opt/lampp/bin/mysql /opt/lampp/bin/mysql.server
su user
```

RedHat / CentOS

```
# yum install mysql-devel
#connect
1 /opt/lampp/bin/mysql -u root -p
#show users
SELECT User, Host, authentication_string FROM mysql.user;
#show users
select host, user, password from mysql.user;
2 create database ossec;
grant INSERT,SELECT,UPDATE,CREATE,DELETE,EXECUTE on ossec.* to
ossecuser@localhost;
3 grant INSERT,SELECT,UPDATE,CREATE,DELETE,EXECUTE on ossec.* to
ossecuser@127.0.0.1;
set password for ossecuser@localhost=PASSWORD('ossecpass');
4 set password for ossecuser@127.0.0.1=PASSWORD('ossecpass');
5 mysql> flush privileges;
6 mysql> quit
7 /opt/lampp/bin/mysql -u root -p ossec < mysql.schema
/opt/lampp/bin/mysql -u root -p ossec < mysql.schema /home/user/Desktop/ossec-hids-
2.8.3/src/os_dbd/mysql.schema
mysql -u root -p ossec < mysql.schema /home/user/Desktop/ossec-hids-
2.8.3/src/os_dbd/mysql.schema
```

### Post configuration

```
/var/ossec/etc/ossec.conf
<ossec_config>
<database_output>
<hostname>192.168.2.30</hostname>
<username>ossecuser</username>
<password>ossecpass</password>
<database>ossec</database>
<type>mysql</type>
</database_output></ossec_config>
# /var/ossec/bin/ossec-control enable database
# /var/ossec/bin/ossec-control restart
```

**After the configuration checking whether the mysql is active:**

```
[root@ossec user]# grep ossec-dbd /var/ossec/logs/ossec.log
2018/07/06 13:41:13 ossec-dbd: Connected to database 'ossec' at '127.0.0.1'.
2018/07/06 13:42:49 ossec-dbd: INFO: Started (pid: 18907).
[root@ossec user]# /opt/lampp/bin/mysql -u root -p
```

**These are the tables of the database ossec:**

```
mysql> connect ossec;
Connection id: 5
Current database: ossec
```

```
mysql> show tables;
```

```
+-----+
| Tables_in_ossec      |
+-----+
| agent                |
| alert                |
| category             |
| data                 |
| location             |
| server               |
| signature            |
| signature_category_mapping |
+-----+
```

```
8 rows in set (0.00 sec)
```

**These are the columns of the alert table, where all ossec alerts are stored here:**

These are the columns witch shows info for every alert. These columns show id, server\_id, rule id, timestamp of the alert, location id, source ip, destination ip and ports. All these are critical info for a security analyst team.

```
mysql> show columns from alert;
```

```
+-----+-----+-----+-----+-----+-----+
| Field  | Type                | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id     | int(10) unsigned   | NO   | PRI | NULL    |      |
| server_id | smallint(5) unsigned | NO   | PRI | NULL    |      |
| rule_id | mediumint(8) unsigned | NO   | MUL | NULL    |      |
| timestamp | int(10) unsigned   | NO   | MUL | NULL    |      |
| location_id | smallint(5) unsigned | NO   |    | NULL    |      |
| src_ip  | int(10) unsigned   | YES  | MUL | NULL    |      |
| dst_ip  | int(10) unsigned   | YES  |    | NULL    |      |
| src_port | smallint(5) unsigned | YES  |    | NULL    |      |
| dst_port | smallint(5) unsigned | YES  |    | NULL    |      |
| alertid | tinytext           | YES  |    | NULL    |      |
```

```

+-----+-----+-----+-----+-----+-----+
10 rows in set (0.00 sec)

Below is a screenshot of the alert table where all alerts of the OSSEC are stored, easily with an
export command:
(SELECT 'ID number','alert policy')UNION(SELECT id,rule_id FROM alertINTO OUTFILE
'analysis.csv'FIELDS ENCLOSED BY '"' TERMINATED BY ',' ESCAPED BY '"'LINES
TERMINATED BY '\r\n');
| 1425 |      1 |      516 | 1538593123 |      25 |      0 |      0 |      0 |      0 | 1538593122.445
9 |
| 1426 |      1 |      516 | 1538593123 |      25 |      0 |      0 |      0 |      0 | 1538593122.479
1 |
| 1427 |      1 |      516 | 1538593123 |      25 |      0 |      0 |      0 |      0 | 1538593122.511
3 |
| 1428 |      1 |      501 | 1538593333 |      26 |      0 |      0 |      0 |      0 | 1538593330.544
8 |
| 1429 |      1 |      550 | 1538593483 |      4 |      0 |      0 |      0 |      0 | 1538593483.564
8 |
| 1430 |      1 |      550 | 1538593493 |      4 |      0 |      0 |      0 |      0 | 1538593493.609
3 |
| 1431 |      1 |     10100 | 1538593533 |      24 | 3232249957 |      0 |      0 |      0 | 1538593532.653
9 |
| 1432 |      1 |      5501 | 1538593533 |      24 |      0 |      0 |      0 |      0 | 1538593532.686
2 |
| 1433 |      1 |      550 | 1538593583 |      4 |      0 |      0 |      0 |      0 | 1538593580.714
1 |
| 1434 |      1 |      5502 | 1538593598 |      24 |      0 |      0 |      0 |      0 | 1538593596.758
6 |
| 1435 |      1 |      5502 | 1538593598 |      24 |      0 |      0 |      0 |      0 | 1538593596.782
7 |
| 1436 |      1 |      5503 | 1538594139 |      24 | 3232249957 |      0 |      0 |      0 | 1538594134.806
6 |
| 1437 |      1 |      5716 | 1538594139 |      24 | 3232249957 |      0 |      0 |      0 | 1538594136.842
5 |
| 1438 |      1 |      516 | 1538594144 |      27 |      0 |      0 |      0 |      0 | 1538594143.874
1 |
| 1439 |      1 |      516 | 1538594144 |      27 |      0 |      0 |      0 |      0 | 1538594143.905
6 |
| 1440 |      1 |      516 | 1538594144 |      27 |      0 |      0 |      0 |      0 | 1538594143.938
8 |
| 1441 |      1 |      516 | 1538594144 |      27 |      0 |      0 |      0 |      0 | 1538594143.971
4 |
| 1442 |      1 |     1002 | 1538594149 |      24 |      0 |      0 |      0 |      0 | 1538594146.100
46 |
| 1443 |      1 |      502 | 1538594830 |      15 |      0 |      0 |      0 |      0 | 1538594828.103
57 |
| 1444 |      1 |      5716 | 1538595095 |      24 | 3232249957 |      0 |      0 |      0 | 1538595090.105
12 |
| 1445 |      1 |      5716 | 1538595305 |      24 | 3232249957 |      0 |      0 |      0 | 1538595301.108
29 |
| 1446 |      1 |      5716 | 1538595370 |      24 | 3232249957 |      0 |      0 |      0 | 1538595369.111
46 |
+-----+-----+-----+-----+-----+-----+
-----
1446 rows in set (0.01 sec)

mysql> █

```

### Figure 29 OSSEC alerts from MySQL

All events are displayed in the OSSEC GUI, where alerts containing information about the level, rule id location and file path of the related event. Also in the upper left corner information about the agent status is available.

Main	Search	Integrity checking	Stats	About
------	--------	--------------------	-------	-------

October 06th, 2018 11:40:38 PM

### Available agents:

- ossec-server (127.0.0.1)  
**Name:** ossec-server  
**IP:** 127.0.0.1  
**Last keep alive:** 2018 Oct 06 23:40:38  
**OS:** Linux ossec 2.6.32-573.12.1.el6.x86\_64 #1 SMP Tue Dec 15 21:19:08 UTC 2015  
 x86\_64 x86\_64 x86\_64 GNU/Linux
- honeypot1 (192.168.56.102)  
**Name:** honeypot1  
**IP:** 192.168.56.102  
**Last keep alive:** 2018 Oct 06 23:37:09  
**OS:** Linux osboxes 4.4.0-31-generic #50~14.04.1-Ubuntu SMP Wed Jul 13 01:07:32  
 UTC 2016 x86\_64 - OSSEC HIDS v2.8.3
- +webserver (192.168.56.101)

### Latest modified files:

- +/etc/cups/subscriptions.conf
- +/etc/cups/subscriptions.conf.O
- +/etc/cups/subscriptions.conf.O
- +/etc/cups/subscriptions.conf
- +/usr/bin/gnome-text-editor
- +/usr/bin/gedit
- +/etc/gconf/gconf.xml.defaults/%gconf-tree-ro...

---

### Latest events

<b>Level:</b> 3 - Login session closed.	2018 Oct 06 23:36:29
<b>Rule id:</b> 5502	
<b>Location:</b> localhost->/var/log/secure	
Oct 6 23:36:28 localhost su: pam_unix(su:session): session closed for user root	
<b>Level:</b> 2 - Unknown problem somewhere in the system.	2018 Oct 06 23:22:04
<b>Rule id:</b> 1002	
<b>Location:</b> (webserver) 192.168.56.101->/var/log/syslog	

**Figure 30 OSSEC event viewer**

Also searching old events is available:



[Main](#) [Search](#) [Integrity checking](#) [Stats](#) [About](#)

October 06th 2018 11:41:21 PM

## Alert search options:

From: 2018-10-06 19:41 To: 2018-10-06 23:41   
 Real time monitoring

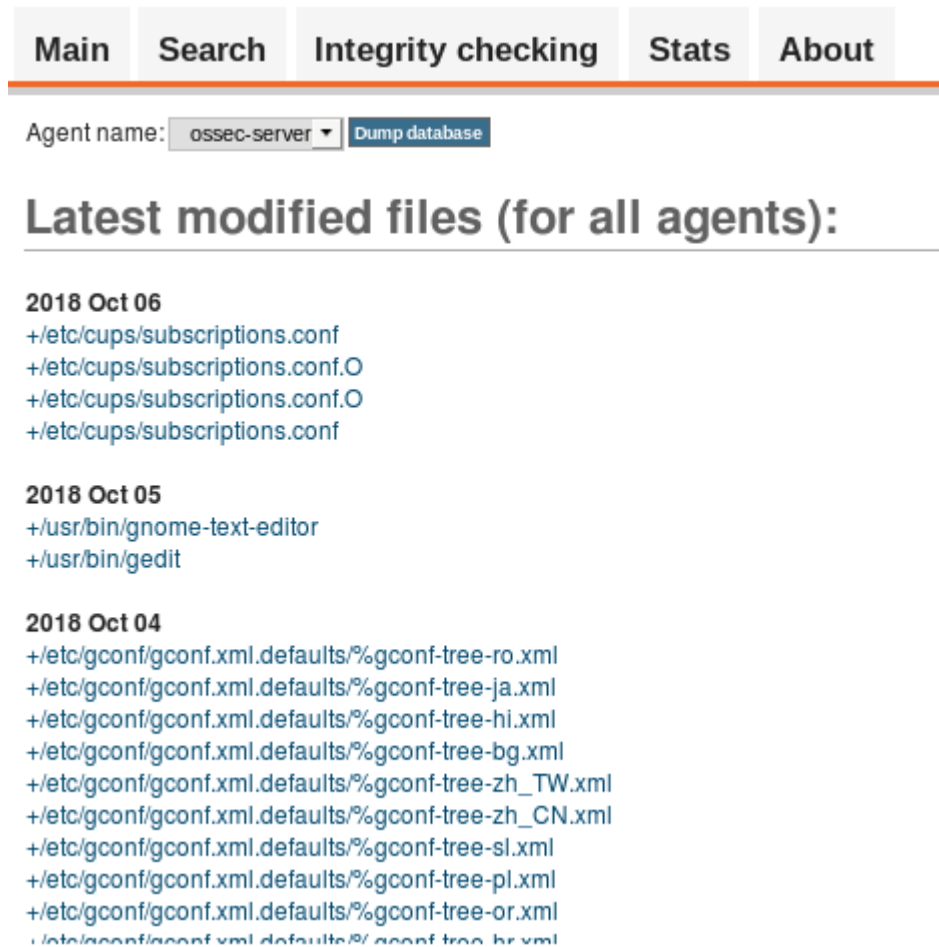
Minimum level: 7 Category: All categories   
Pattern:  Log formats: All log formats   
Srcip:  User:   
Location:  Rule id:   
Max Alerts: 1000

## Results:

No search performed.

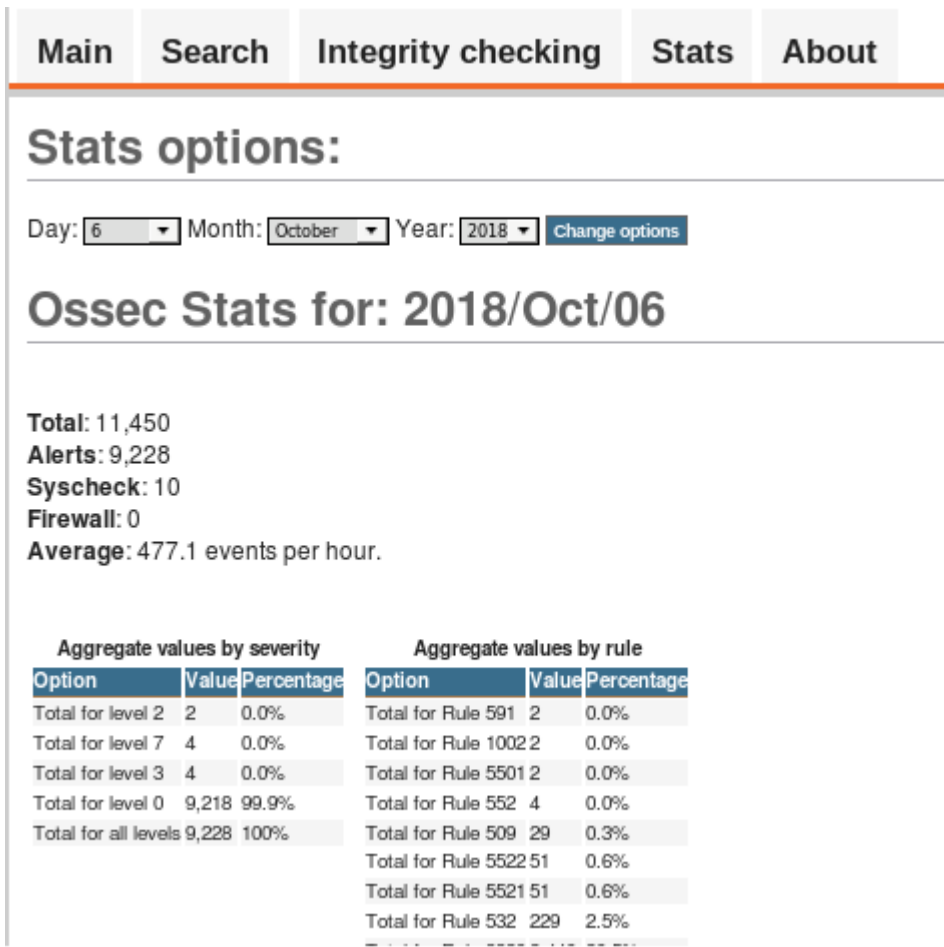
### Figure 31 OSSEC event history

Here an overview of the last modified files is available for all monitored systems:



**Figure 32 OSSEC file integrity**

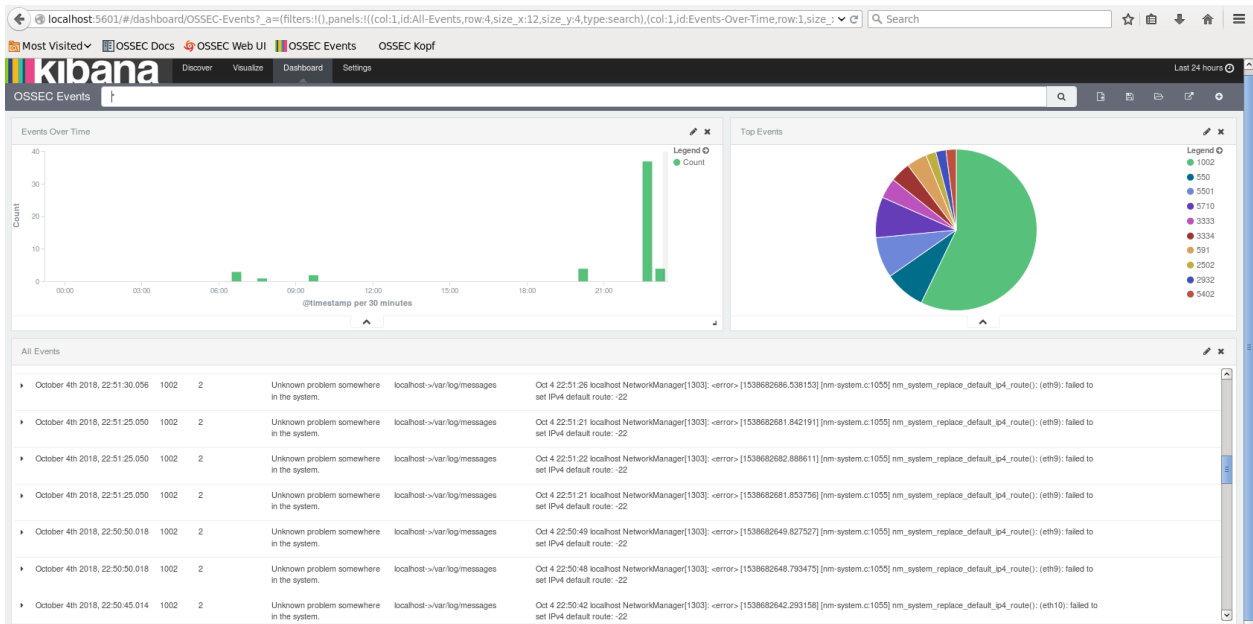
And general statistics over the whole monitoring of the systems:



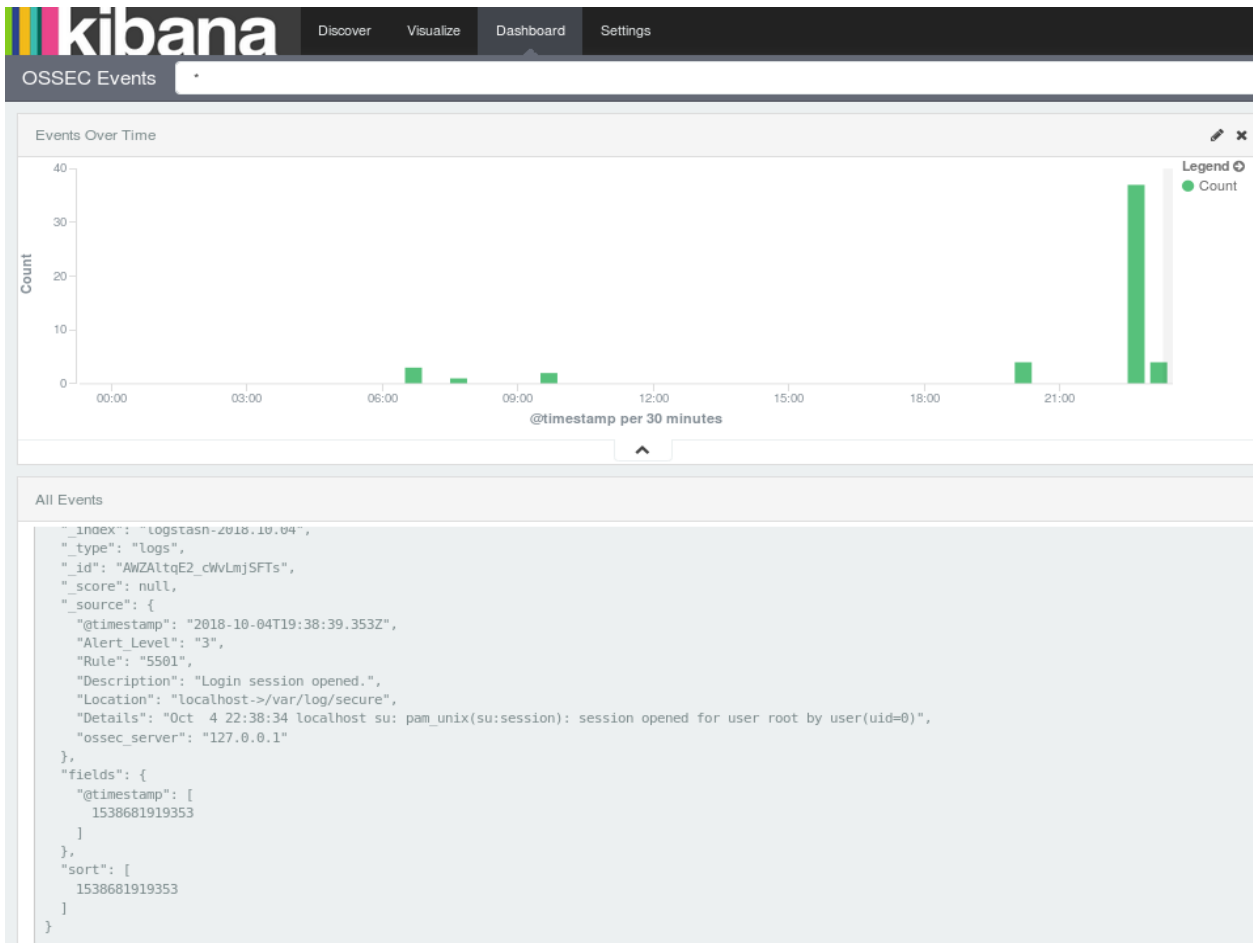
**Figure 33 OSSEC statistics**

Also Kibana dashboard is available for viewing alerts from all systems with ossec agent deployed in a faster way, including all relevant information attached to event. Furthermore each event has available json or table with all information.





**Figure 34 Kibana dashboard**



**Figure 35 Kibana Json format of event**

### 8.6 Proof of concept deploying conpot

In this section a demonstration of Fabric usage will be implemented. This includes the automation of installing prerequisites, move conpot software and deploying it alongside with the OSSEC agent. The implementation below is a proof of concept of the automation process of a system capable of installing, deploying and monitoring of a network of SCADA honeypots using a variety of PLC templates.

- 1) Dependencies installation

# Fabric-Web

## Dependencies Ubuntu!

Install the required dependencies for Ubuntu 12.04 LTS / 14.04 LTS / 16.04 LTS

### Execute the task on...

- Roles
- No Roles - I'm going to use simple hostnames.
  - ossec-server - 192.168.56.110
  - honeypot3 - 192.168.56.104
  - honeypot2 - 192.168.56.103
  - honeypot1 - 192.168.56.101
  - webservers - 192.168.56.102
  - intranet - honeypot1,honeypot2,honeypot3

hostname

**Figure 36 Dependencies installation 1**

## Fabric-Web

executed *dependencies\_ubuntu* on .  
output for 192.168.56.101

```
Executing task 'dependencies_ubuntu'  
run: apt-get --assume-yes install libmysqlclient-dev libsmi2ldb1  
out:  
out: Reading package lists... 0%  
out:  
out: Reading package lists... 100%  
out:  
out: Reading package lists... Done  
out:  
out:  
out: Building dependency tree... 0%  
out:  
out: Building dependency tree... 0%  
out:  
out: Building dependency tree... 50%  
out:
```

**Figure 37 Dependencies installation output**

- 2) Conpot installation

# Fabric-Web

## Conpot Ubuntu Installation!

Install the stable version of ConPot

### Execute the task on...

- Roles
- No Roles - I'm going to use simple hostnames.
  - ossec-server - 192.168.56.110
  - honeypot3 - 192.168.56.104
  - honeypot2 - 192.168.56.103
  - honeypot1 - 192.168.56.101
  - webservers - 192.168.56.102
  - intranet - honeypot1,honeypot2,honeypot3

hostname

**Figure 38 Conpot installation**

## Fabric-Web

### executed *conpot ubuntu installation* o output for 192.168.56.101

```
Executing task 'conpot_ubuntu_installation'  
run: pip install conpot  
out: Requirement already satisfied: conpot in /usr/local/lib/pyt  
out: Requirement already satisfied: beautifulsoup4 in /usr/local  
out: Requirement already satisfied: requests in /usr/lib/python2  
out: Requirement already satisfied: lxml in /usr/lib/python2.7/d  
out: Collecting pysnmp==4.2.5 (from conpot)  
out: Requirement already satisfied: MySQL-python in /usr/lib/pyt  
out: Requirement already satisfied: modbus-tk in /usr/local/lib/  
out: Requirement already satisfied: sphinx in /usr/local/lib/pyt  
out: Requirement already satisfied: mixbox in /usr/local/lib/pyt  
out: Requirement already satisfied: xlrd in /usr/local/lib/pytho  
out: Requirement already satisfied: pyghmi in /usr/local/lib/pyt  
out: Requirement already satisfied: stix-validator in /usr/local  
out: Requirement already satisfied: enum in /usr/local/lib/pytho  
out: Requirement already satisfied: bottle in /usr/local/lib/pyt
```

**Figure 39 Conpot installation output**

3) In order conpot to be open for attacks some ports must be open, so open firewall function automates this task

# Fabric-Web

---

## Open Firewall!

Open ports in firewalld : 80 , 102, 161 and 502

---

### Execute the task on..

- Roles
- No Roles** - I'm going to use simple hostnames.
  - ossec-server** - 192.168.56.110
  - honeypot3** - 192.168.56.104
  - honeypot2** - 192.168.56.103
  - honeypot1** - 192.168.56.101
  - webserver** - 192.168.56.102
  - intranet** - honeypot1,honeypot2,honeypot3

hostname

**Figure 40 open firewall**

## Fabric-Web

---

executed *open firewall* on .

output for 192.168.56.101

```
Executing task 'open_firewall'  
run: apt-get install firewalld -y  
out:  
out: Reading package lists... 0%  
out:  
out: Reading package lists... 100%  
out:  
out: Reading package lists... Done  
out:  
out:  
out: Building dependency tree... 0%  
out:  
out: Building dependency tree... 0%  
out:  
out: Building dependency tree... 50%  
out:
```

**Figure 41 open firewall output**

- 4) Move ossec agent binaries to honeypot and deploy



# Fabric-Web

---

## Move Ossec Agent Honeypot1!

Put ossec agent software in target

---

### Execute the task on...

- Roles
- No Roles - I'm going to use simple hostnames.
  - ossec-server - 192.168.56.110
  - honeypot3 - 192.168.56.104
  - honeypot2 - 192.168.56.103
  - honeypot1 - 192.168.56.101
  - webservice - 192.168.56.102
  - intranet - honeypot1,honeypot2,honeypot3

hostname

**Figure 42** move ossec agent

## Fabric-Web

**executed *move ossec agent honeypot***  
**output for 192.168.56.101**

```
Executing task 'move_ossec_agent_honeypot1'  
put: /home/opcadm/ossec-hids-2.8.3.zip -> /home/osboxes/ossec-hi
```

## Command

Here is the command that we just run, if you want to add it into a script.

```
$ fab move_ossec_agent_honeypot1 -H
```

**Figure 43 move ossec agent output**

# Fabric-Web

## Deploy Ossec Agent!

Deploy agent conpot software in target

### Execute the task on...

- Roles
- No Roles - I'm going to use simple hostnames.
  - ossec-server - 192.168.56.110
  - honeypot3 - 192.168.56.104
  - honeypot2 - 192.168.56.103
  - honeypot1 - 192.168.56.101
  - webservice - 192.168.56.102
  - intranet - honeypot1,honeypot2,honeypot3

hostname

**Figure 44** deploy ossec agent

## Fabric-Web

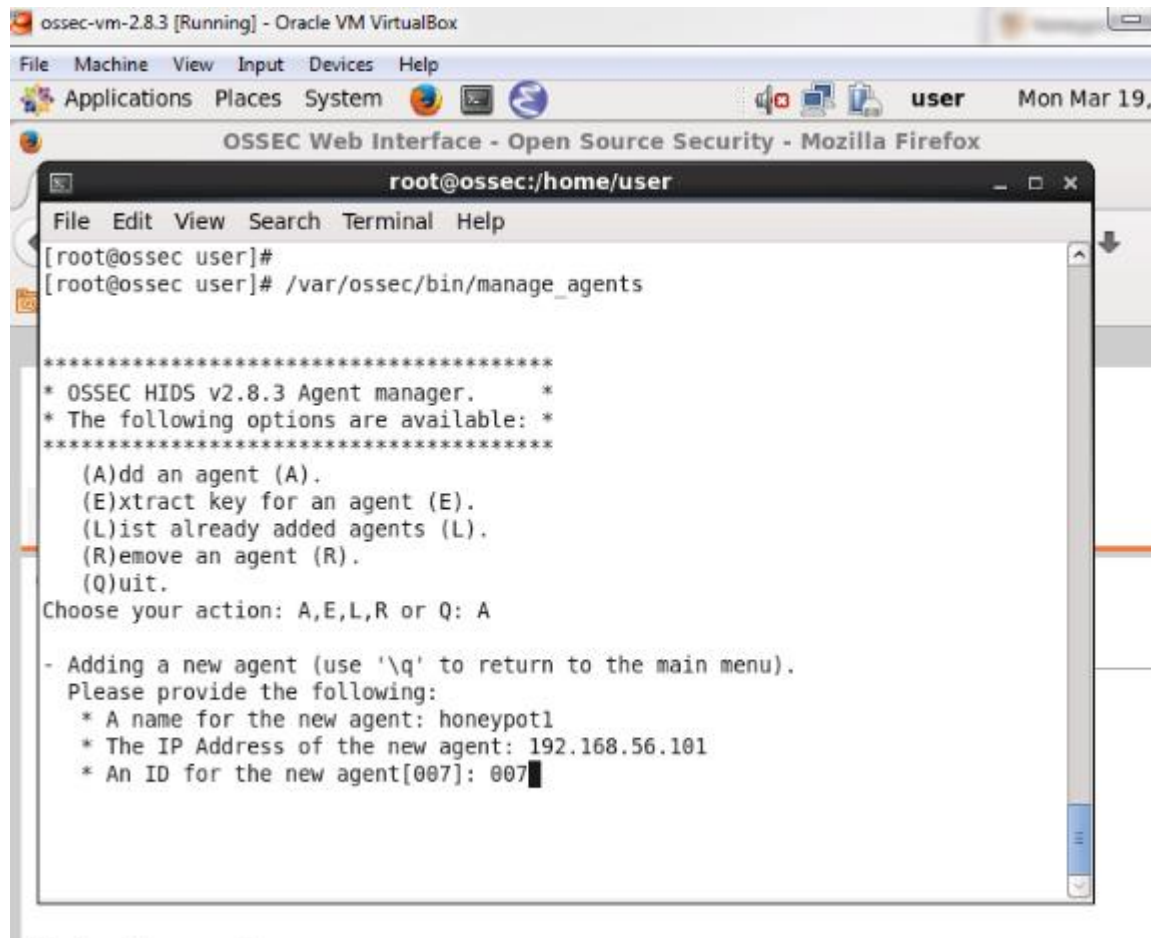
*executed `deploy ossec agent` on .*

**output for 192.168.56.101**

```
Executing task 'deploy_ossec_agent'  
run: apt-get install unzip  
out:  
out: Reading package lists... 0%  
out:  
out: Reading package lists... 100%  
out:  
out: Reading package lists... Done  
out:  
out:  
out: Building dependency tree... 0%  
out:  
out: Building dependency tree... 0%  
out:  
out: Building dependency tree... 50%  
out:
```

**Figure 45 deploy ossec agent output**

5) Insert ossec agent, this is a manual step in order to finalize the agent installation  
Ossec server side:

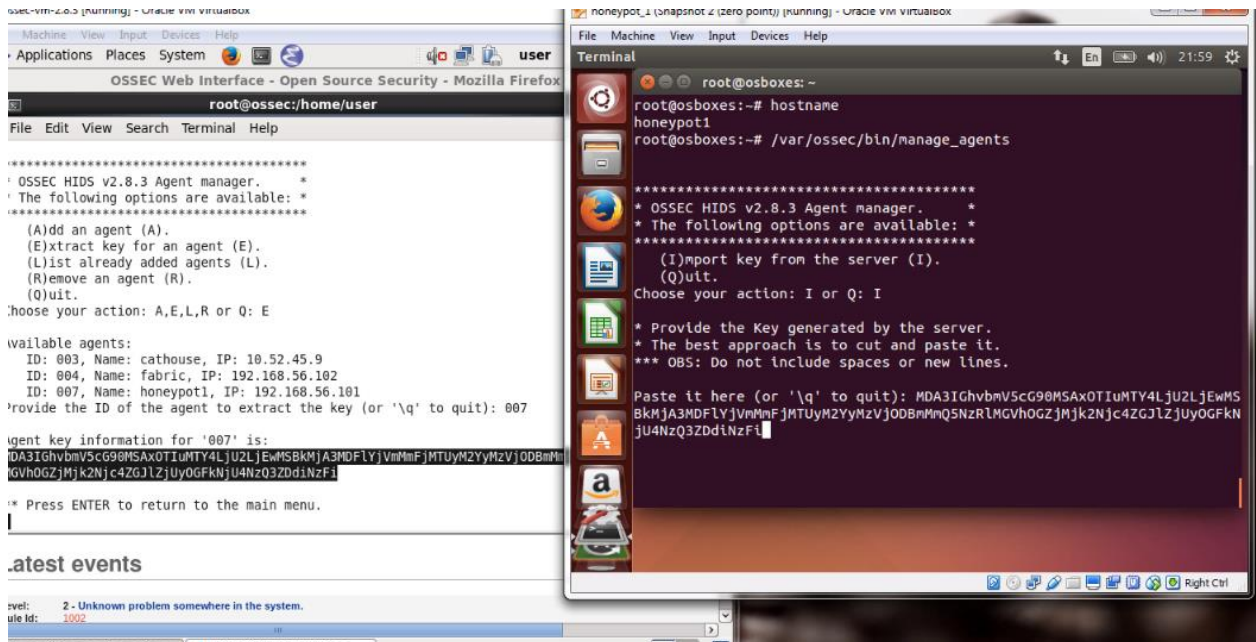


The screenshot shows a terminal window titled 'root@ossec:/home/user' running the command `/var/ossec/bin/manage_agents`. The terminal output displays the OSSEC agent manager menu, which lists options: (A)dd an agent (A), (E)xtract key for an agent (E), (L)ist already added agents (L), (R)emove an agent (R), and (Q)uit. The user has chosen option 'A'. The terminal then prompts for the following information:

- A name for the new agent: honeypot1
- The IP Address of the new agent: 192.168.56.101
- An ID for the new agent[007]: 007

**Figure 46 ossec-server agent installation**

Honeypot side:



**Figure 47 ossec agent installation**

6) Start conpot by choosing a template

## Fabric-Web

### Run Conpot Template S7 200!

Rough simulation of a basic Siemens S7-200 CPU with 2 slaves. Protocols: HTTP, MODBUS

Execute the task on...

- Roles
- No Roles - I'm going to use simple hostnames.
  - ossec-server - 192.168.56.110
  - honeypot3 - 192.168.56.104
  - honeypot2 - 192.168.56.103
  - honeypot1 - 192.168.56.101
  - webservice - 192.168.56.102
  - intranet - honeypot1,honeypot2,honeypot3

hostname

**Figure 48 start conpot by template**



**Fabric-Web**

**executed *run conpot template S7 200***

**output for 192.168.56.101**

```
Executing task 'run_conpot_template_S7_200'  
run: (nohup conpot --template default >& /dev/null < /dev/null &
```

**Command**

Here is the command that we just run, if you want to add it into a script.

```
$ fab run_conpot_template_S7_200 -H
```

**Figure 49 start conpot by template output**

- 7) Check whether the conpot software is running

# Fabric-Web

## Check Conpot Running!

Check if conpot is running

### Execute the task on...

Roles  **No Roles** - I'm going to use simple hostnames.

**ossec-server** - 192.168.56.110

**honeypot3** - 192.168.56.104

**honeypot2** - 192.168.56.103

**honeypot1** - 192.168.56.101

**webserver** - 192.168.56.102

**intranet** - honeypot1,honeypot2,honeypot3

hostname

**Figure 50 check conpot**



# Fabric-Web

executed *check conpot running on .*

output for 192.168.56.101

```
Executing task 'check_conpot_running'
run: ps -ef| grep template
out: nobody    20886      1 12 22:02 ?          00:00:03 /usr/bin/py
out: root      20932 17538   0 22:03 pts/12    00:00:00 /bin/bash -
out: root      20935 20932   0 22:03 pts/12    00:00:00 grep templa
out:
run: cat /root/conpot.log
out: 2018-03-19 22:02:52,809 Starting Conpot using template: /us
es/default
out: 2018-03-19 22:02:52,810 Starting Conpot using configuration
conpot/conpot.cfg
out: 2018-03-19 22:02:52,995 Starting new HTTP connection (1): w
out: 2018-03-19 22:02:55,395 Could not fetch public ip from http
out: 2018-03-19 22:02:55,400 Starting new HTTP connection (1): q
out: 2018-03-19 22:02:56,320 Could not fetch public ip from http
out: 2018-03-19 22:02:56,325 Starting new HTTP connection (1): i
```

**Figure 51 check conpot running output**

From this output we understand that conpot is running. Also a view in conpot logs follows.

## 8.7 Proof of concept attacking conpot

The following sector gives an example of a penetration test in the live SCADA honeypot where conpot is running with SIEMENS PLC template and OSSEC is monitored the whole procedure. From this chapter capabilities of the mechanism deployed earlier can be noticed. The honeypot was an Ubuntu system 64-bit with 192.168.56.101 local Ethernet IP address.

By using Nmap we have an overview of the network. After that from analysis we distinguish that behind port 102 is a Siemens control machine and the port 502 is open. Hence because port 502 belongs to protocol ModBus we understand that a PLC is behind that.

```
root@kali:~# nmap -A -Pn -p1-1000 192.168.56.101
```

```

root@kali:~# nmap -A -Pn -p1-1000 192.168.56.101

Starting Nmap 7.60 ( https://nmap.org ) at 2018-10-08 11:43 EDT
Nmap scan report for 192.168.56.101
Host is up (0.0037s latency).
Not shown: 995 filtered ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.10 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
| 1024 d7:92:f8:90:40:76:b2:8b:d8:40:bb:7b:f3:7c:a1:65 (DSA)
| 2048 18:81:6a:5d:18:03:bf:3b:9c:e2:8e:5a:4a:19:24:34 (RSA)
| 256 80:03:f4:5d:b2:1a:82:5e:6b:4a:24:03:3e:bb:33:d7 (ECDSA)
|_ 256 4a:ff:60:22:5b:62:d1:73:57:e4:15:7a:6f:9c:00:cf (EdDSA)
80/tcp    closed http
102/tcp   closed iso-tsap
161/tcp   closed snmp
502/tcp   closed mbap
MAC Address: 08:00:06:6C:66:85 (Siemens AG)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.10 - 4.8
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT     ADDRESS
1   3.72 ms 192.168.56.101

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 19.95 seconds

```

**Figure 52 Nmap on target**

MODBUS is an application-layer messaging protocol, positioned at level 7 of the OSI model. It provides client/server communication between devices connected on different types of buses or networks.

From OSSEC events we can see that scan was implemented on the monitored system of OSSEC.

```

Latest events
-----
Level:      6 - SSH insecure connection attempt (scan).
Rule Id:    5706
Location:   (honeypot1) 192.168.56.101->/var/log/auth.log
Src IP:     192.168.56.120
Mar 7 12:12:37 osboxes sshd[6227]: Did not receive identification string from 192.168.56.120

```

**Figure 53 OSSEC alert on Nmap**

In order to check whether UDP ports are open the use of Nmap as follows:

```
root@kali:~# nmap -sU -p161 192.168.56.101 --script snmp-sysdescr
```

```

root@kali:~# nmap -sU -p161 192.168.56.101 --script snmp-sysdescr

Starting Nmap 7.60 ( https://nmap.org ) at 2018-10-08 11:45 EDT
Nmap scan report for 192.168.56.101
Host is up (0.0029s latency).

PORT      STATE SERVICE
161/udp   filtered snmp
MAC Address: 08:00:06:6C:66:85 (Siemens AG)

Nmap done: 1 IP address (1 host up) scanned in 6.31 seconds

```

**Figure 54 Nmap on 161 port**

Thus now metasploit can be used in order to check conpot logs and OSSEC reaction. From all payloads we use modbusdetect.

use auxiliary/scanner/scada/modbusdetect

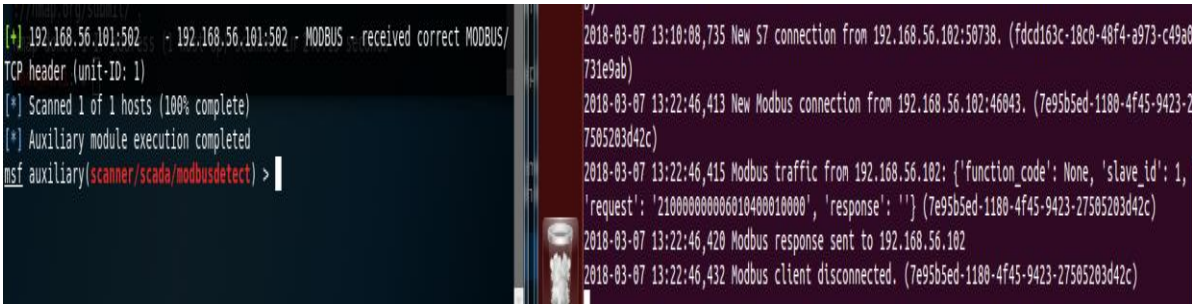


Figure modbusdetect use and conpot logs

We understand that modbus protocol is active from use auxiliary/scanner/scada/modbusdetect, so we believe that a SCADA PLC is behind that.

From auxiliary/scanner/scada/modbus\_findunitid we can see clients ID's:

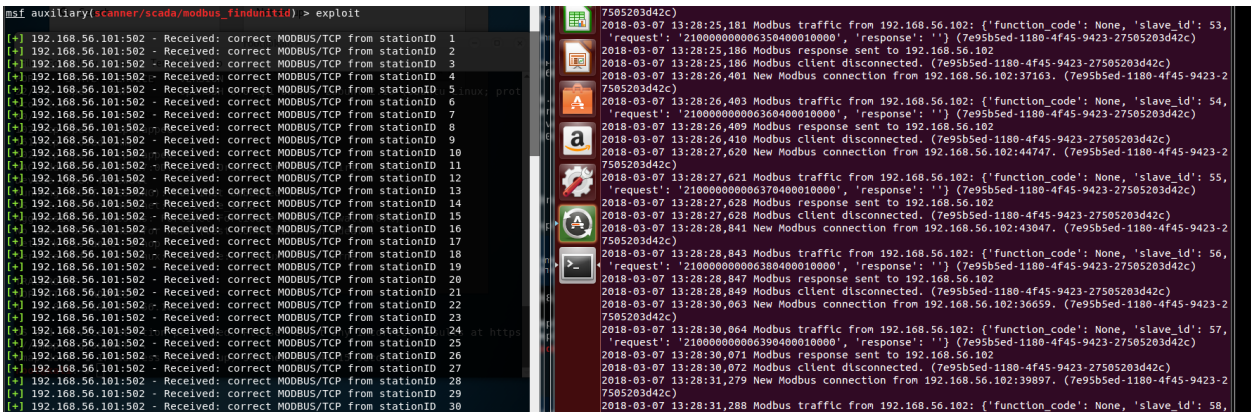
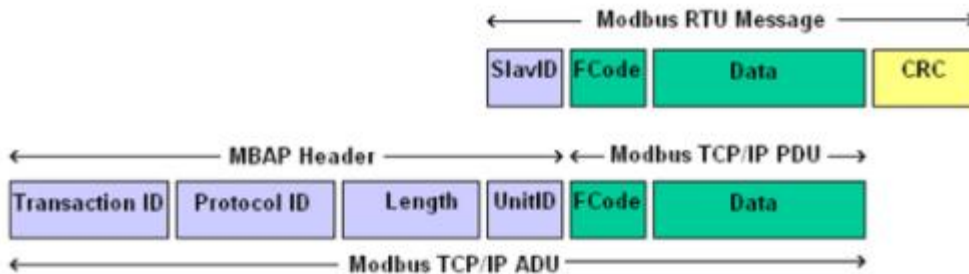


Figure 55 findunitid and conpot logs

From here all stationIDs are available from any system client.

Furthermore with payload auxiliary/scanner/scada/modbusclient, we can change and add binaries of modbus client coils. This can create a major disaster in the SCADA configuration because its can turnoff/on the PLC or change the motor speed.

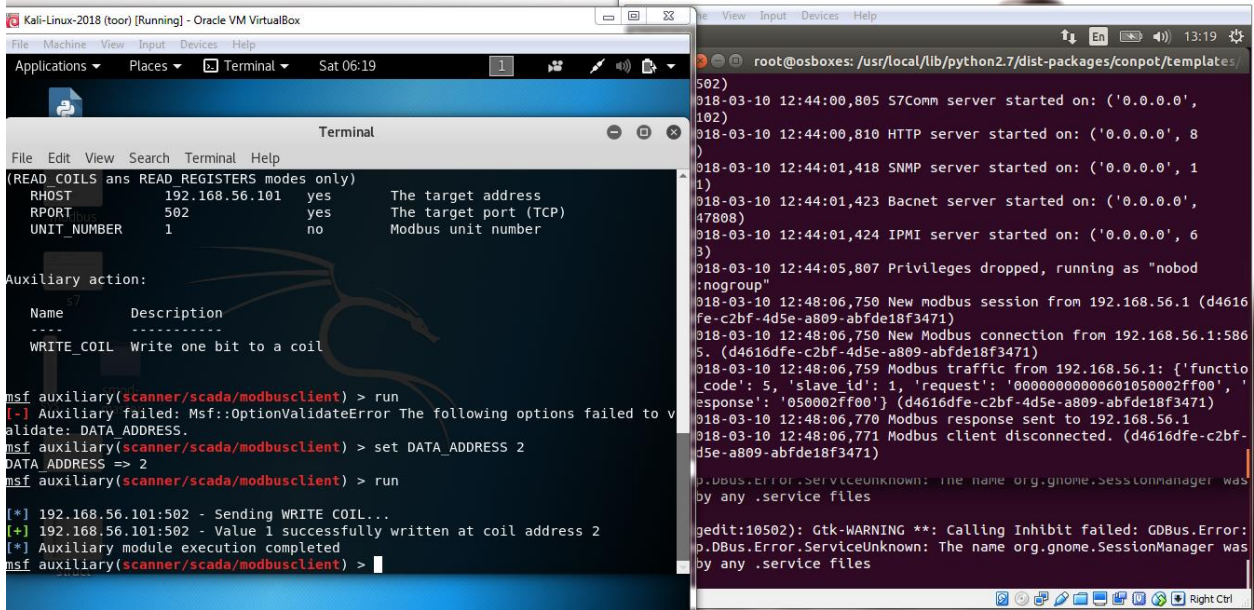
By default the server uses Modbus protocol on a serial mode. Thus in case a UID is missing, this means that there is no a slave device after a certain UID, the exploitation returns “fail”. All Modbus protocol packages contain slave ID or Unit ID function code, data section and error checker.



**Figure 56 Modbus protocol**

(Modbus TCP/IP, 2017)

In case there is no slave behind an ID because of the Siemens template we use for conpot the exploitation returns an error. But as explained in a previous chapter in conpot installation after a customization in conpot code inside /slave\_db.py this error is ignored and the attacker believes that the exploit is successful. Hence receives a positive output from read/write request in PLC and that the configuration was changed.

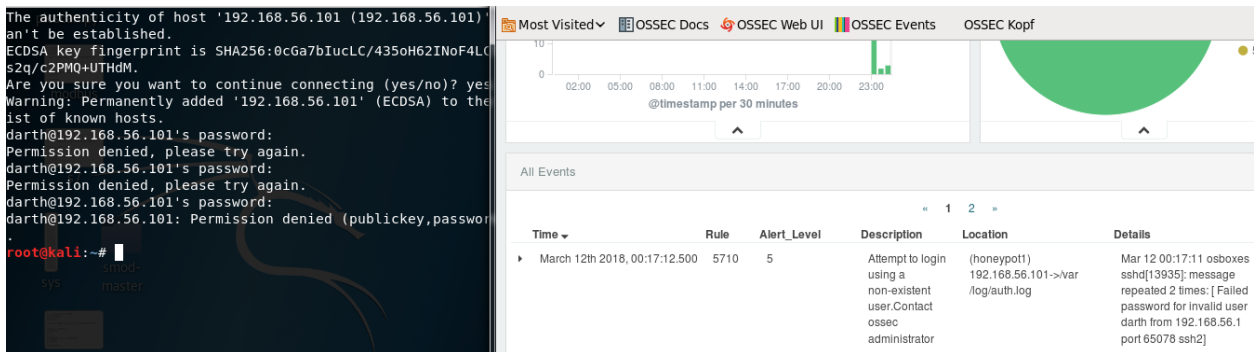


**Figure 57 successful exploitation output**

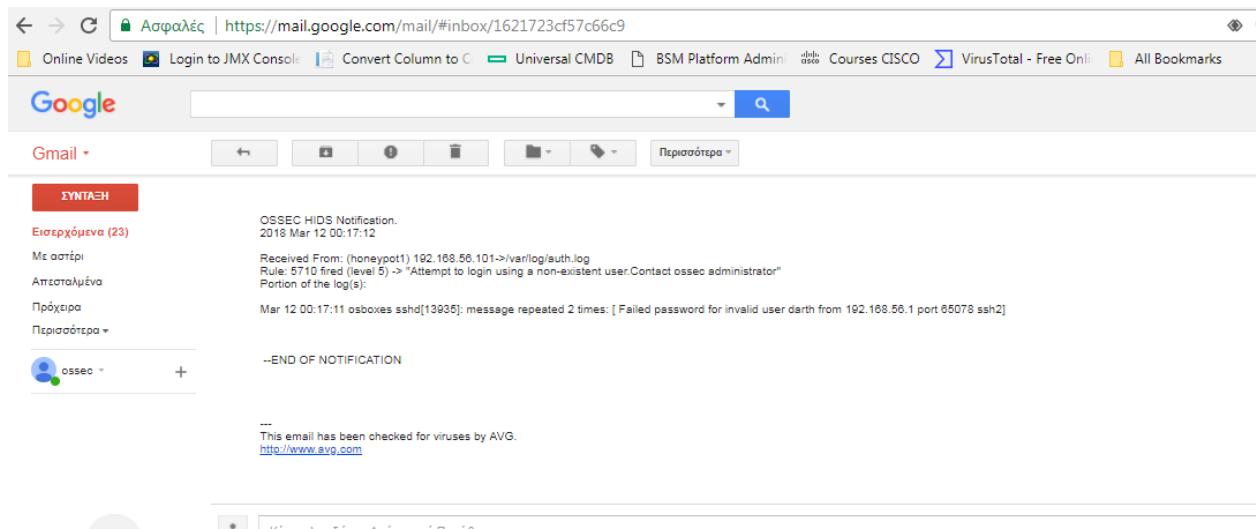
### 8.8 OSSEC reporting

A simple demonstration of OSSEC alerting mechanism will be demonstrated here:

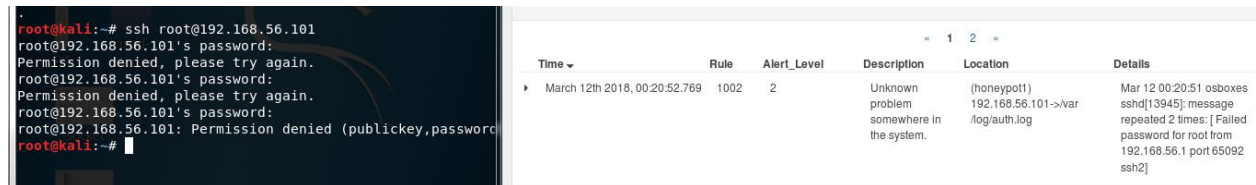
1. unauthorized login (Attempting to login as default users)



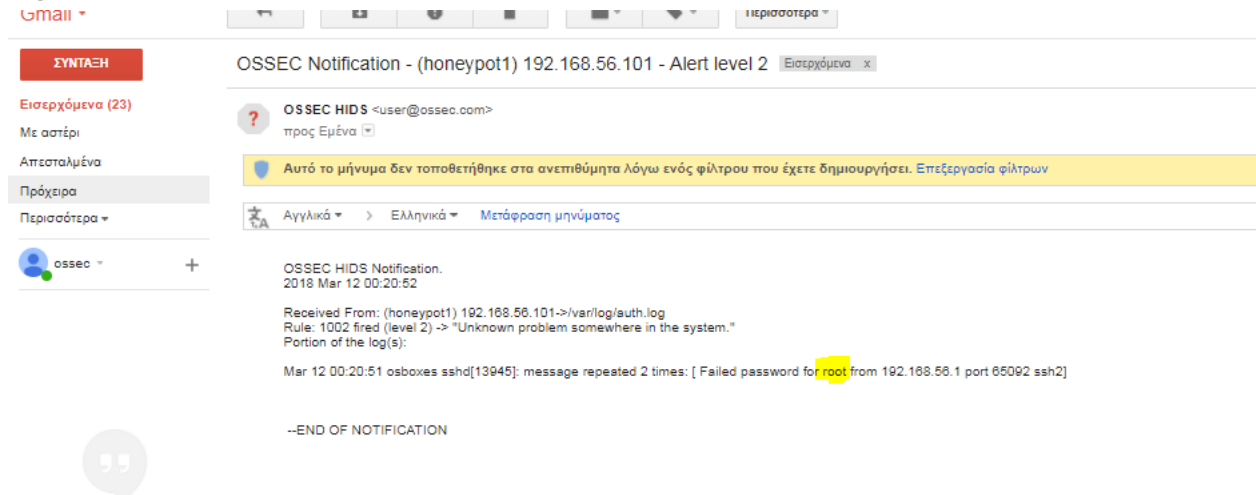
**Figure 58 unauthorized login and kibana event**



**Figure 59 unauthorized login in email alert**

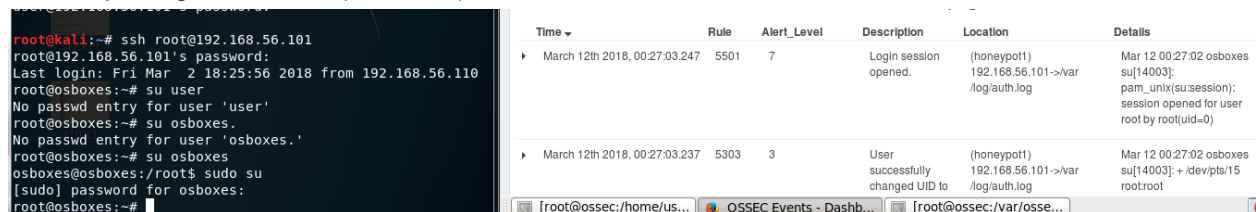


**Figure 60 permission denied and kibana event**



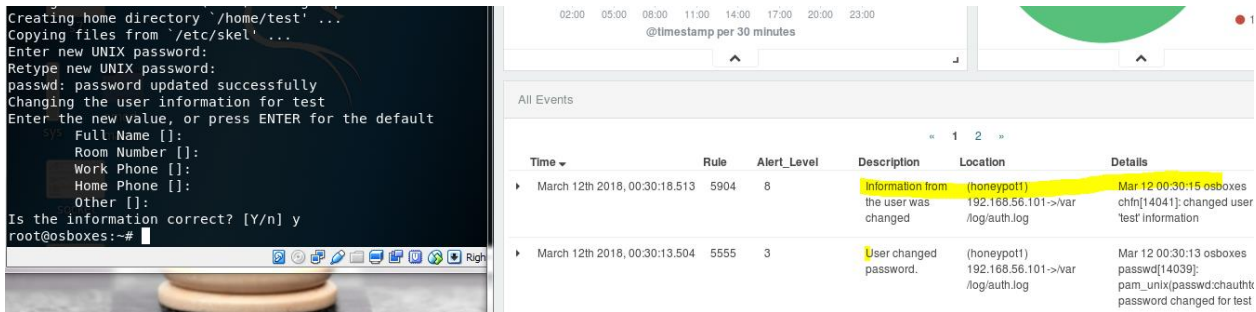
**Figure 61 permission denied in email alert**

2. privilege escalation (su to root)



**Figure 62 privilege escalation and kibana event**

3. Add user



**Figure 63 add user and kibana event**

4. syslog integrity

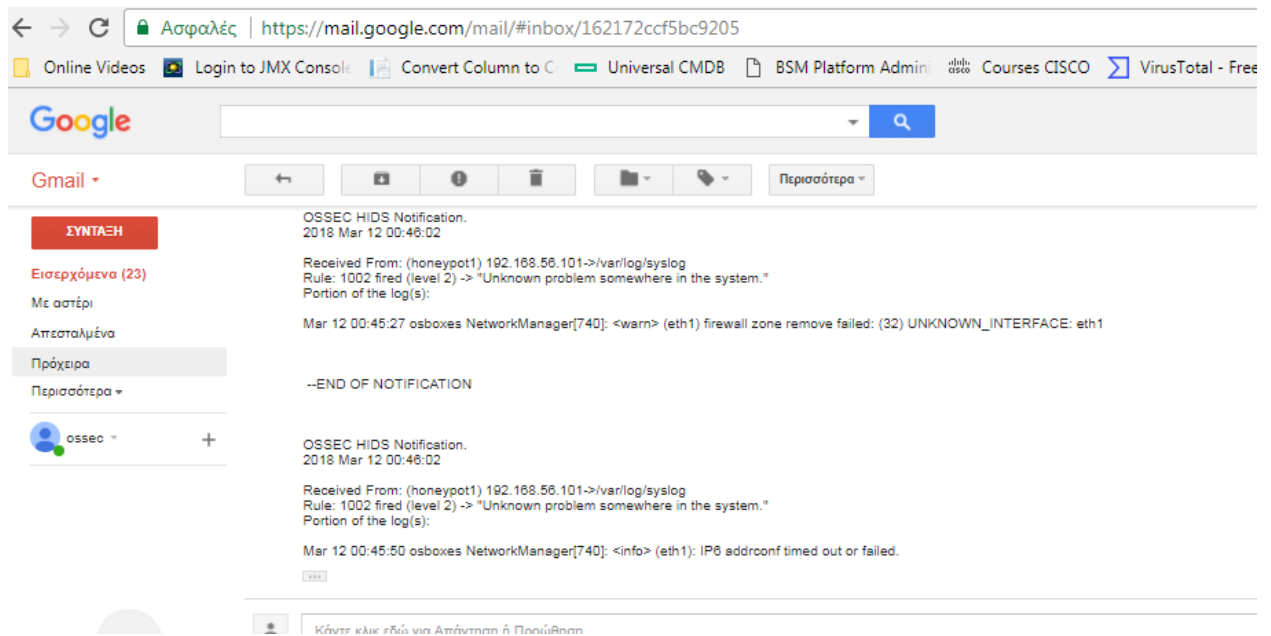


**Figure 64 syslog integrity and ossec gui event**

5. Host is down (network error or attack)

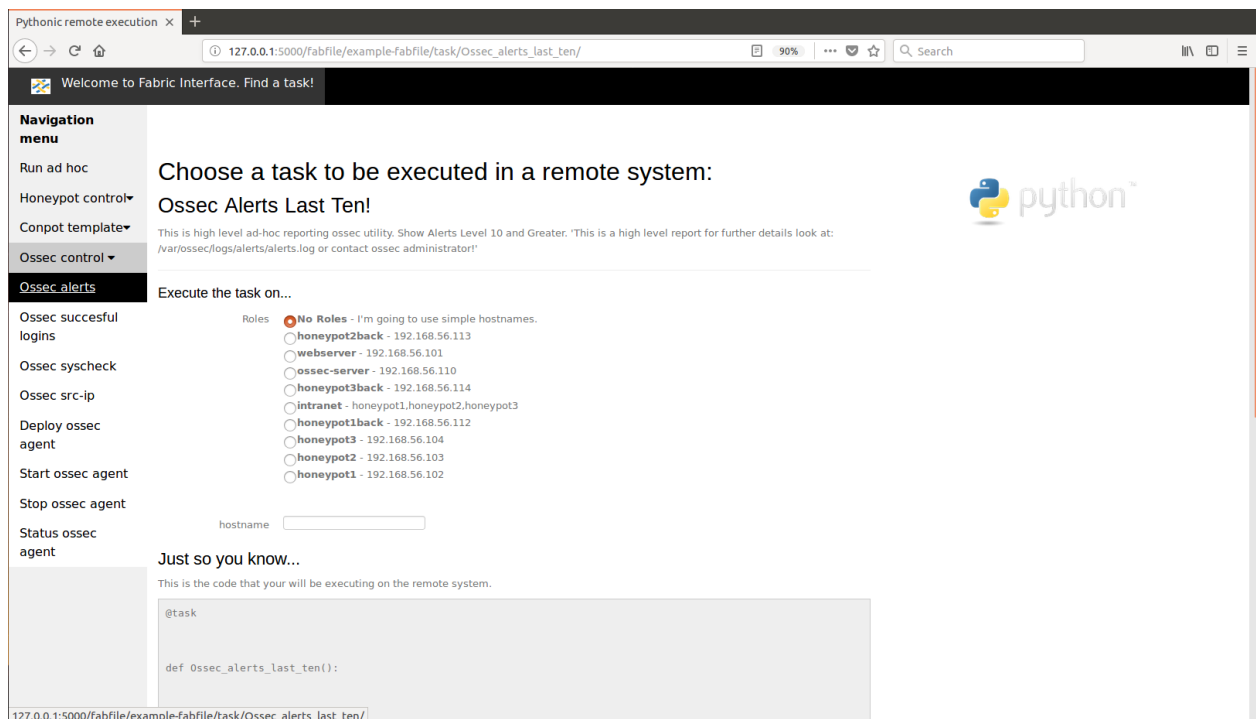


**Figure 65 Host is down and ossec gui event**



**Figure 66 Host is down email alert**

Moreover besides the configuration in ossec server where we can manage to send email alerts with reports via postfix server there are available in fabric configuration ad hoc high level reports. Thus from Fabric tool we can have any time reports over the systems that are monitored from OSSEC server.



**Figure 67 OSSEC report from Fabric UI for level 10 alerts**

## Chapter 9 Future work – Conclusions

This chapter includes the future development this system requires in order to enrich the processes and add more business value in order to be deployed in a large production infrastructure. Moreover conclusions of the whole process are here in order to give the perspective of the implementer.

### 9.1 Future development

This project aim is to give a holistic approach in the installation, deployment and monitoring of SCADA honeypots inside an internal network. Below are several future additions anyone can implement to this project in order to automate more the process and add value and precision.

Former to begin with a future plan could be to create a template and an automation mechanism of deploying a template which have already installed, deployed and configured all the steps now implanted via Fabric tool. This addition can enhance the whole procedure by giving a faster way to add inside the internal network a ready honeypot with whatever PLC configuration a user request every time. The procedure of the VM creation can be an automated task as well.

Moreover another interesting subject is to make the conpot software to be as much realistic as possible. The conpot software is open source and this gives the opportunity to edit and change parts of the source code in order to make the software for the attacker realistic. More variables can be added in order the software to be more like a real programming logic controller.

A very interesting addition could be a way to monitor the process live through a more business user interface with simple alerts indicating security incidents when these incidents apply to a specific pattern.

Another critical addition can be the integration with the inotify. The inotify is a linux kernel subsystem capable to notice changes in the file system. For example can it can understand when someone add a file in the file systems and report those changes to a console or an application.

The solution this thesis proposes can be used also for environments with an already deployed conpot. Further analysis is required in order to monitor a conpot installation and apply all the customization implemented here. Also for customers with a special custom configuration conpot can be easily configured to include all aspects and options.

Latter another very interesting topic needs further implementation would be to analyze some patters before and after a SCADA attack and determine as many as it is possible patterns. Thus after the analysis we can train the systems by applying machine learning algorithms in order the system to inform operators of an upcoming attack or false alarms. This can be done by email or/and event in the KIBANA user interface with further information on the triggered event. Big data algorithms also can be used for classification of a variety of attack patterns from the event database repository. Moreover future implementer should consider the upgrade of the tools that has been used to create the architecture of the implementation.

### 9.2 Conclusions - Summary

This thesis proposed a complex architecture of many integrated systems, each one with a different role. Firstly, the machine where Fabric tool is installed witch can easily automate tasks inside the network. The OSSEC system responsible for the monitoring of the infrastructure and event production based on several security policies. Lastly the honeypot machines where can be deployed with different OS or PLC template. This architecture can add value to an organization while using it because multiple SCADA honeypots can be deployed and configured easily with automated tasks. Moreover with the OSSEC and ELK operators can monitor suspicious attacks on the honeypots offering an insight over the malicious steps. All this movement from conpot logs can be valuable because attack patterns can be emerging by using big data algorithms while mining the database



integrated with OSSEC. Thus the capabilities of the conpot honeypot can be explored to its full by using the different PLC templates available or when a custom templates created based on customer needs and configuration. Hence this project can provide a security mechanism with broad capabilities using only open source software.

To begin with the automated configuration of a Conpot SCADA Honeypot architecture presented in this thesis is designed to operate inside an internal network, able to simulate a SCADA/ICS system. In the chapter one the analysis and some information about SCADA systems is discussed so as to give to the reader an insight to the motivation that lead to this diploma thesis. Furthermore, the second chapter focuses in the security policies and tools used for SCADA systems. In the next chapter a short presentation take place about the conpot honeypot and its abilities and usage. Similarly in the next chapter the fabric tool is presented alongside with installation instructions and usage. Subsequently the seventh chapter explains the usage of ELK stack and the reason that has been chosen. After that the chapter eight starts with the architecture design and information regarding all the open source programs has been used and the workflow of the implementation. A proof of concept follows with screenshots of the live system both when Fabric tool is used or what is happening and how OSSEC reacts when an attack is taking place in SCADA honeypot. At the end future work presented alongside with conclusions. In the appendixes all the code crated for the automation through fabric tool and the customized code from conpot is totally available.

## Chapter 10 References

- 1) Stuart A.Boyer, (2004). SCADA Supervisory Control and Data Acquisition 3rd edition.
- 2) Cyber Security for SCADA Systems.(2013, Autumn). Thalescyberassurance
- 3) Rossella Mattioli, Konstantinos Moulinos, (2015). Analysis of ICS-SCADA Cyber Security Maturity Levels in Critical Sectors.
- 4) Edward J. M. Colbert, Alexander Kott, (2016). Cyber-security of SCADA and Other Industrial Control Systems.
- 5) Keith Stouffe, Joe Falco, Karen Kent . (2006). Guide to Supervisory Control and Data Acquisition (SCADA) and Industrial Control Systems Security.
- 6) Athar Mahboob,Junaid Ahmed Zubairi, (2013). Securing SCADA Systems with Open Source Software, 978-1-4799-2569-8/13.
- 7) Stephen Miller, Richard H. Clark, (2015). Framework for SCADA Cybersecurity.
- 8) Martti Lehto, Pekka Neittaanmäki.(2015). Cyber Security: Analytics, Technology and Automation.
- 9) Robert Radvanovsky Jacob Brodsky.(2013) Handbook of SCADA / Control Systems Security. Boca Raton, FL Taylor & Francis Group
- 10) Sandia National Laboratories. (2005), Penetration Testing of Industrial Control Systems.
- 11) CONPOT ICS/SCADA Honeypot. (2018). Retrieved from <http://conpot.org/>
- 12) CONPOT ICS/SCADA Honeypot. (2018), Conpot Documentation Release 0.5.1.
- 13) Daniel B. Cid.(n.d). Log Analysis using OSSEC.
- 14) OSSEC Project. (2014), OSSEC Documentation Release 2.8.1.
- 15) Vyacheslav Fadyushin, Andrey Popov.(2016), Building a Pentesting Lab for Wireless Networks.
- 16) Fabric Pythonic remote execution. (2014), Fabric documentantion release.
- 17) fabric-web. (2018), Retrieved from <https://github.com/daniellawrence/fabric-web/>
- 18) Elastic Stack.(2018), Retrieved from <https://www.elastic.co/elk-stack>
- 19) Vic Hargrave.(nd). OSSEC Log Management with Elasticsearch.
- 20) How To Install Elasticsearch, Logstash, and Kibana. (2018), Retrieved from <https://www.digitalocean.com/community/tutorials/how-to-install-elasticsearch-logstash-and-kibana-elk-stack-on-centos-7>

- 21) Return no response when there is no slave behind a UID. (2018), Retrieved from <https://github.com/mushorg/conpot/pull/324>
- 22) slave\_db.py. (2018), Retrieved from [https://github.com/mushorg/conpot/blob/master/conpot/protocols/modbus/slave\\_db.py](https://github.com/mushorg/conpot/blob/master/conpot/protocols/modbus/slave_db.py)
- 23) Modbus TCP/IP. (2017), Retrieved from <http://www.simplymodbus.ca/TCP.htm>

## Appendix A

### Fabfile.py

```
#!/usr/bin/env python
"""This is the automation procedure of the deployment and configuration of conpot or other scada
honeypot program
"""

from fabric.api import task, run, settings, hide, put, cd
from fabric.api import env
from fabric.decorators import hosts, roles
from fabric.context_managers import cd
env.user = "root"
env.password = "osboxes.org"
env.roledefs = {
    'honeypot1': ['192.168.56.102'],
    'honeypot1back': ['192.168.56.112'],
    'honeypot2': ['192.168.56.103'],
    'honeypot2back': ['192.168.56.113'],
    'honeypot3': ['192.168.56.104'],
    'honeypot3back': ['192.168.56.114'],
    'webserver': ['192.168.56.101'],
    'ossec-server': ['192.168.56.110'],
    'intranet': ['honeypot1','honeypot2','honeypot3']
}
#@roles('intranet')
@task
def Open_firewall_in_target():
    """ Open ports in firewalld : 80 , 102, 161 and 502 """
    run('apt-get install firewalld -y')
    run('firewall-cmd --permanent --add-port=80/tcp')
    run('firewall-cmd --permanent --add-port=102/tcp')
    run('firewall-cmd --permanent --add-port=161/tcp')
    run('firewall-cmd --permanent --add-port=502/tcp')
    run('firewall-cmd --reload')
@task
def Check_target_in_conpot():
    """ Get info from target """
```

```

run('uptime')
run('hostname')
run('whoami')
@task
def Check_conpot_running():
    """ Check if conpot is running """
    run('ps -ef| grep template')
    run('cat /root/conpot.log')
@task
def Run_template_of_S7_200():
    """ Rough simulation of a basic Siemens S7-200 CPU with 2 slaves. Protocols: HTTP, MODBUS,
s7comm, SNMP """
    run("(nohup conpot --template default >& /dev/null < /dev/null &) && sleep 1|| exit 0")
@task

def Stop_conpot_if_running():
    """ With this method conpot honeypot can be terminated manually by user """
    run("pkill -9 -u `id -u nobody`")
@task
def Run_template_of_kamstrup():
    """ Register clone of an existing Kamstrup 382 smart meter. Protocols: Kamstrup """
    #run('conpot --template kamstrup_382')
    run("(nohup conpot --template kamstrup_382 >& /dev/null < /dev/null &) && sleep 1")
@task
def Run_template_of_guardian():
    """ Guardian AST tank-monitoring system. Protocols: guardian_ast """
    #run('conpot --template guardian_ast')
    run("(nohup conpot --template guardian_ast >& /dev/null < /dev/null &) && sleep 1")
@task
def Run_template_of_Proxy():
    """ Sample template that demonstrates the proxy feature. Protocols: Proxy """
    #run('conpot --template proxy')
    run("(nohup conpot --template proxy >& /dev/null < /dev/null &) && sleep 1")
@task
def Run_template_of_IPMI():
    """ Creates a simple IPMI device. Protocols: IPMI """
    #run('conpot --template ipmi')
    run("(nohup conpot --template ipmi >& /dev/null < /dev/null &) && sleep 1")

#@roles('centos')
@task
def Dependencies_in_centos():

```

```

""" Install the required dependencies for CentOS version 7.3-1611 & Conpot 0.5.1 """
run('yum -y update')
run('yum -y install libxslt-devel libxml2-devel python-pip python')
run('yum -y install mariadb-server mysql-connector-python.noarch mariadb-devel')
run('yum -y install git python-lxml.x86_64 python-devel')
run('yum -y groupinstall "Development tools')
run('wget https://bootstrap.pypa.io/get-pip.py && sudo python ./get-pip.py')
run('pip install -U lxml')
run('chkconfig mariadb on')
run('service mariadb start')
#@roles('centos')
@task
def Conpot_centos_installation():
    """ Install Conpot 0.5.1 to CentOS """
    run('git clone https://github.com/mushorg/conpot')
    run('cd conpot/')
    run('python setup.py install')
#@roles('ubuntu')
@task
def Dependencies_in_ubuntu():
    """ Install the required dependencies for Ubuntu 12.04 LTS / 14.04 LTS / 16.04 LTS """
    run('apt-get --assume-yes install libmysqlclient-dev libsmi2ldbl snmp-mibs-downloader python-dev
libevent-dev')
    run('apt-get --assume-yes install libxslt1-dev libxml2-dev python-pip python-mysqldb pkg-config
libvirt-dev')
    run('echo "deb http://packages.dotdeb.org wheezy all" >> /etc/apt/sources.list')
    run('echo "deb-src http://packages.dotdeb.org wheezy all" >> /etc/apt/sources.list')
    run('apt-get -y update')
    run('apt-get --assume-yes --allow-unauthenticated install libmysqlclient-dev')
    run('apt-get --assume-yes --allow-unauthenticated install python-pip')
    run('apt-get --assume-yes install python-setuptools')
    run('apt-get --assume-yes install python3-setuptools')
    run('pip install -U setuptools')
    run('pip install --upgrade pip')
    run('apt-get --assume-yes install sshpass')
    run('apt-get --assume-yes install git')
    run('apt-get --assume-yes install snmp-mibs-downloader')
    #run('pip install --upgrade lxml')
    #put('/home/opcadm/modbus-tk', '/home/osboxes')
    #run('cd /home/osboxes/modbus-tk';python setup.py install')
#@roles('ubuntu')
@task

```

```

def Conpot_ubuntu_installation():
    """ Install the stable version of ConPot """
    run('pip install conpot')
    run('pip uninstall bacpypes -y')
    run('pip install -lv bacpypes\>=\=0.13.8')
    run('pip install --upgrade pysnmp')
    run('pip install --upgrade pysnmp')
    run('conpot')
#@roles('debian')
@task
def Dependencies_in_debian():
    """ Install the required dependencies for Debian 6.0.7 64bit & 7.2.0 64bit """
    run('apt-get install git libsmi2ldbl smistrip libxslt1-dev python-dev libevent-dev')
    run('apt-get install python-pip')
    run('pip install argparse')
    run('echo "deb http://ftp.nl.debian.org/debian squeeze main non-free" >> /etc/apt/sources.list')
    run('apt-get update')
    run('apt-get install snmp-mibs-downloader')
#@roles('debian')
@task
def Conpot_debian_installation():
    """ Install the development version of ConPot """
    run('cd /opt')
    run('git clone https://github.com/mushorg/modbus-tk.git')
    run('cd modbus-tk')
    run('cd ..')
    run('git clone https://github.com/mushorg/conpot.git')
    run('cd conpot')
    run('python setup.py install')
#@roles('intranet')
@task
def Running_command(executable_command):
    """Connect to the remote system(s) are run any command that you type in. You can also choose
conpot template manually:
    1) conpot --template default
    2) conpot --template kamstrup_382
    3) conpot --template guardian_ast
    4) conpot --template proxy
    5) conpot --template ipmi """
    run("%s" % ".join(executable_command)")
@task
def Move_ossec_honeypot1():

```

```

    """ Put ossec agent software in target """
    put('/home/opcadm/ossec-hids-2.8.3.zip','/home/osboxes')
@task
def Move_ossec_honeypot2():
    """ Put ossec agent software in target """
    put('/home/opcadm/ossec-hids-2.8.3.zip','/home/osboxes')
@task
def Move_ossec_honeypot3():
    """ Put ossec agent software in target """
    put('/home/opcadm/ossec-hids-2.8.3.zip','/home/osboxes')
@task
def Deploy_ossec_agent_target():
    """ Deploy agent conpot software in target """
    run('apt-get install unzip')
    run('unzip /home/osboxes/ossec-hids-2.8.3.zip -d /home/osboxes')
    run('chmod -R 777 /home/osboxes/ossec-hids-2.8.3/ossec-hids-2.8.3')
    run("(nohup /home/osboxes/ossec-hids-2.8.3/ossec-hids-2.8.3/install.sh >& /dev/null < /dev/null &)
    && sleep 1")
@task
def Ossec_sucesful_logins_last():
    """ This is high level ad-hoc reporting ossec utility. Show Successful Logins. 'This is a high level
    report for further details look at: /var/ossec/logs/alerts/alerts.log or contact ossec administrator!' """
    run('cat /var/ossec/logs/alerts/alerts.log | /var/ossec/bin/ossec-reportd -f group
    authentication_success')
    run('echo This is a high level report for further details look at: /var/ossec/logs/alerts/alerts.log or
    contact ossec administrator!')
@task
def Ossec_alerts_last_ten():
    """ This is high level ad-hoc reporting ossec utility. Show Alerts Level 10 and Greater. 'This is a
    high level report for further details look at: /var/ossec/logs/alerts/alerts.log or contact ossec
    administrator!' """
    run('cat /var/ossec/logs/alerts/alerts.log | /var/ossec/bin/ossec-reportd -f level 10')
    run('echo This is a high level report for further details look at: /var/ossec/logs/alerts/alerts.log or
    contact ossec administrator!')
@task
def Ossec_check_source_IP():
    """ This is high level ad-hoc reporting ossec utility. Show the srcip for all users. 'This is a high level
    report for further details look at: /var/ossec/logs/alerts/alerts.log or contact ossec administrator!' """
    run('cat /var/ossec/logs/alerts/alerts.log | /var/ossec/bin/ossec-reportd -f group authentication -r
    user srcip')
    run('echo This is a high level report for further details look at: /var/ossec/logs/alerts/alerts.log or
    contact ossec administrator!')
@task

```

```

def Ossec_check_the_systems():
    """ This is high level ad-hoc reporting ossec utility. Show Changed files as reported by Syscheck.
    'This is a high level report for further details look at: /var/ossec/logs/alerts/alerts.log or contact ossec
    administrator!' """
    run('cat /var/ossec/logs/alerts/alerts.log | /var/ossec/bin/ossec-reportd -f group syscheck -r location
    filename')
    run('echo This is a high level report for further details look at: /var/ossec/logs/alerts/alerts.log or
    contact ossec administrator!')
@task
def Java_prerequisites_install():
    """ This is method installs JRE and JDK """
    run('sudo apt-get install default-jre')
    run('sudo apt-get install default-jdk')
    run('java - version')
    run('update-alternatives --config java')
    run('vi /etc/environment (JAVA_HOME="/usr/lib/jvm/java-8-openjdk-amd64")')
    run('source /etc/environment')
    run('echo $JAVA_HOME')
@task
def Change_target_MAC_Siemens():
    """ This is method change the MAC address from the target node to imitate Siemens PLC """
    run('apt-get install -y macchanger')
    run('macchanger eth0 --mac=00:1c:06:03:61:68')
@task
def Change_target_MAC_kamstrup():
    """ This is method change the MAC address from the target node to imitate kamstrup PLC """
    run('apt-get install -y macchanger')
    run('macchanger eth0 --mac=00:80:32:00:2a:72')
@task
def Change_target_MAC_guardian():
    """ This is method change the MAC address from the target node to imitate guardian PLC """
    run('apt-get install -y macchanger')
    run('macchanger eth0 --mac=00:1d:09:8b:8e:94')

@task
def Check_target_MAC_address():
    """ This is method returns target MAC address """
    run('ip link ')
@task
def Start_ossec_agent():
    """ This is method starts ossec agent """
    run('/var/ossec/bin/ossec-control start')

```

```

@task
def Stop_ossec_agent():
    """ This is method stops ossec agent """
    run('/var/ossec/bin/ossec-control stop')
@task
def Status_ossec_agent():
    """ This is the status of ossec agent """
    run('/var/ossec/bin/ossec-control status')
@task
def Check_spoofed_mac():
    """ This is the status of ossec agent """
    run('nmap -sP -n 192.168.56.102')
    run('nmap -sP -n 192.168.56.103')
    run('nmap -sP -n 192.168.56.104')
slave_db.py
# modified by Sooky Peter <xsooky00@stud.fit.vutbr.cz>
# Brno University of Technology, Faculty of Information Technology
import struct
from lxml import etree

from modbus_tk.modbus import Databank, DuplicatedKeyError, MissingKeyError, \
    ModbusInvalidRequestError
from modbus_tk import defines

from conpot.protocols.modbus.slave import MBSlave
import logging

logger = logging.getLogger(__name__)

class SlaveBase(Databank):

    """
    Database keeping track of the slaves.
    """

    def __init__(self, template):
        Databank.__init__(self)
        self.dom = etree.parse(template)

    def add_slave(self, slave_id, unsigned=True, memory=None):

```



```

"""
Add a new slave with the given id
"""
if (slave_id < 0) or (slave_id > 255):
    raise Exception("Invalid slave id %d" % slave_id)
if slave_id not in self._slaves:
    self._slaves[slave_id] = MBSlave(slave_id, self.dom)
    return self._slaves[slave_id]
else:
    raise DuplicatedKeyError("Slave %d already exists" % slave_id)

def handle_request(self, query, request, mode):
    """
    Handles a request. Return value is a tuple where element 0
    is the response object and element 1 is a dictionary
    of items to log.
    """
    request_pdu = None
    response_pdu = ""
    slave_id = None
    function_code = None
    func_code = None
    slave = None
    response = None

    try:
        # extract the pdu and the slave id
        slave_id, request_pdu = query.parse_request(request)
        if len(request_pdu) > 0:
            (func_code, ) = struct.unpack(">B", request_pdu[0])

        logger.debug("Working mode: %s" % mode)

        if mode == 'tcp':
            if slave_id == 0 or slave_id == 255:
                slave = self.get_slave(slave_id)
                response_pdu = slave.handle_request(request_pdu)
                response = query.build_response(response_pdu)
            else:
                # TODO:
                # Shall we return SLAVE DEVICE FAILURE, or ILLEGAL ACCESS?

```

```

    # Would it be better to make this configurable?
    r = struct.pack(
        ">BB", func_code + 0x80, defines.SLAVE_DEVICE_FAILURE)
    response = query.build_response(r)

elif mode == 'serial':
    if slave_id == 0:      # broadcasting
        for key in self._slaves:
            response_pdu = self._slaves[key].handle_request(
                request_pdu, broadcast=True)

        # no response is sent back
        return (None, {'request': request_pdu.encode('hex'),
            'slave_id': slave_id,
            'function_code': func_code,
            'response': ''})
    elif 0 < slave_id <= 247: # normal request handling
        slave = self.get_slave(slave_id)
        response_pdu = slave.handle_request(request_pdu)
        # make the full response
        response = query.build_response(response_pdu)
    else:
        # TODO:
        # Same here. Return SLAVE DEVICE FAILURE or ILLEGAL ACCESS?
        r = struct.pack(
            ">BB", func_code + 0x80, defines.SLAVE_DEVICE_FAILURE)
        response = query.build_response(r)

except (MissingKeyError, IOError) as e:
    logger.error(e)
    # If slave was not found or the request was not handled correctly,
    # return a server error response
    r = struct.pack(
        ">BB", func_code + 0x80, defines.SLAVE_DEVICE_FAILURE)
    response = query.build_response(r)
except ModbusInvalidRequestError as e:
    logger.error(e)
    # TODO: return something here?

if slave:
    function_code = slave.function_code

```

```

return (response, {'request': request_pdu.encode('hex'),
                  'slave_id': slave_id,
                  'function_code': function_code,
                  'response': response_pdu.encode('hex')})

```

## Appendix B

### Base.html

```

<!DOCTYPE html>
<html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Pythonic remote execution</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
  <link rel="stylesheet" href="https://www.w3schools.com/lib/w3-theme-black.css">
  <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">
  <link href="/static/bootstrap/css/bootstrap.min.css" rel="stylesheet">
  <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
  <style type="text/css">
    h1 {

      border-radius: 8px;

    }
    body {
      padding-top: 53.5px;
      padding-bottom: 60px;
    }
    .w3-sidenav a, .w3-sidenav h4{padding:12px;}
    .w3-navbar a{padding-top:12px !important;padding-bottom:12px !important;}

    /* Custom container */
    .container {
      margin: 0 auto;
      max-width: 1000px;
    }
    .container > hr {
      margin: 60px 0;

```

```
}

/* Main marketing message and sign up button */
.jumbotron {
  margin: 80px 0;
  text-align: center;
}
.jumbotron h1 {
  font-size: 100px;
  line-height: 1;
}
.jumbotron .lead {
  font-size: 24px;
  line-height: 1.25;
}
.jumbotron .btn {
  font-size: 21px;
  padding: 14px 24px;
}
/* Supporting marketing content */
.marketing {
  margin: 60px 0;
}
.marketing p + h4 {
  margin-top: 28px;
}

/* Customize the navbar links to be fill the entire space of the .navbar */
.navbar .navbar-inner {
  padding: 0;
}
.navbar .nav {
  margin: 0;
  display: table;
  width: 100%;
}
.navbar .nav li {
  display: table-cell;
  width: 1%;
  float: none;
}
```

```

    .navbar .nav li a {
      font-weight: bold;
      text-align: center;
      border-left: 1px solid rgba(255,255,255,.75);
      border-right: 1px solid rgba(0,0,0,.1);
    }
    .navbar .nav li:first-child a {
      border-left: 0;
      border-radius: 3px 0 0 3px;
    }
    .navbar .nav li:last-child a {
      border-right: 0;
      border-radius: 0 3px 3px 0;
    }

    .w3-btn {margin-bottom:10px;}

}

}

</style>
</head>
<body> <!-- Navbar -->
  <div class="w3-top">
    <div class="w3-bar w3-theme w3-top w3-left-align w3-large"> <a class="w3-bar-item w3-button
w3-right w3-hide-large w3-hover-white w3-large w3-theme-l1"

      href="javascript:void(0)" onclick="w3_open()"><i class="fa fa-bars"></i></a>
      <a href="http://www.fabfile.org/" class="w3-bar-item w3-button w3-theme-l1"></a>
      <a href="http://www.fabfile.org/"

        class="w3-bar-item w3-button w3-hide-small w3-hover-white">Welcome to Fabric Interface.
Find a task!</a>

      <!--<a href="#" class="w3-bar-item w3-button w3-hide-small w3-hover-white">Values</a>
      <a href="#" class="w3-bar-item w3-button w3-hide-small w3-hover-white">News</a>
      <a href="#" class="w3-bar-item w3-button w3-hide-small w3-hover-white">Contact</a>
      <a href="#" class="w3-bar-item w3-button w3-hide-small w3-hide-medium w3-hover-
white">Clients</a>
      <a href="#" class="w3-bar-item w3-button w3-hide-small w3-hide-medium w3-hover-
white">Partners</a>-->
    </div>
  </div>

```

```

<!-- Sidebar -->
<nav class="w3-sidebar w3-bar-block w3-collapse w3-large w3-theme-l5 w3-animate-left"

id="mySidebar"> <a href="javascript:void(0)" onclick="w3_close()" class="w3-right w3-xlarge
w3-padding-large w3-hover-black w3-hide-large"

title="Close Menu"> <i class="fa fa-remove"></i> </a>
<a href="http://127.0.0.1:5000" <h4 class="w3-bar-item"><b>Navigation menu</b> </h4></a>
<a class="w3-bar-item w3-button w3-hover-black" href="http://127.0.0.1:5000/fabfile/example-
fabfile/task/Running_command/">Run ad hoc</a>

<div class="w3-dropdown-hover">
<button class="w3-button">Honeypot control<i class="fa fa-caret-down"></i></button>
<div class="w3-dropdown-content w3-bar-block"
<a class="w3-bar-item w3-button w3-hover-black"></a>
<a class="w3-bar-item w3-button w3-hover-black" href="http://127.0.0.1:5000/fabfile/example-
fabfile/task/Open_firewall_in_target/">Open firewall</a>
<a class="w3-bar-item w3-button w3-hover-black" href="http://127.0.0.1:5000/fabfile/example-
fabfile/task/Check_conpot_running/">Check conpot running</a>
<a class="w3-bar-item w3-button w3-hover-black" href="http://127.0.0.1:5000/fabfile/example-
fabfile/task/Stop_conpot_if_running/">Stop conpot</a>
<a class="w3-bar-item w3-button w3-hover-black" href="http://127.0.0.1:5000/fabfile/example-
fabfile/task/Check_target_in_conpot/">Check target</a>
<a class="w3-bar-item w3-button w3-hover-black" href="http://127.0.0.1:5000/fabfile/example-
fabfile/task/Java_prerequisites_install/">Java install</a>
<a class="w3-bar-item w3-button w3-hover-black"
href="http://127.0.0.1:5000/fabfile/example-fabfile/task/Check_target_MAC_address/">Check MAC
address</a>
<a class="w3-bar-item w3-button w3-hover-black"
href="http://127.0.0.1:5000/fabfile/example-fabfile/task/Check_spoofed_mac/">Check spoofed
mac</a>
</div>
</div>

<div class="w3-dropdown-hover">
<button class="w3-button">Conpot template<i class="fa fa-caret-down"></i></button>
<div class="w3-dropdown-content w3-bar-block"
<a class="w3-bar-item w3-button w3-hover-black"></a>
<a class="w3-bar-item w3-button w3-hover-black" href="http://127.0.0.1:5000/fabfile/example-
fabfile/task/Run_template_of_S7_200/">Run S7 200</a>
<a class="w3-bar-item w3-button w3-hover-black" href="http://127.0.0.1:5000/fabfile/example-
fabfile/task/Run_template_of_guardian/">Run guardian</a>
<a class="w3-bar-item w3-button w3-hover-black" href="http://127.0.0.1:5000/fabfile/example-
fabfile/task/Run_template_of_IPMI/">Run IPMI</a>

```

```
<a class="w3-bar-item w3-button w3-hover-black" href="http://127.0.0.1:5000/fabfile/example-fabfile/task/Run_template_of_kamstrup/">Run Kamstrup</a>
```

```
<a class="w3-bar-item w3-button w3-hover-black" href="http://127.0.0.1:5000/fabfile/example-fabfile/task/Run_template_of_Proxy/">Run Proxy</a>
```

```
</div>
```

```
</div>
```

```
<div class="w3-dropdown-hover">
```

```
<button class="w3-button">Ossec control <i class="fa fa-caret-down"></i></button>
```

```
<div class="w3-dropdown-content w3-bar-block">
```

```
<a class="w3-bar-item w3-button w3-hover-black"></a>
```

```
<a class="w3-bar-item w3-button w3-hover-black" href="http://127.0.0.1:5000/fabfile/example-fabfile/task/Ossec_alerts_last_ten/">Ossec alerts</a>
```

```
<a class="w3-bar-item w3-button w3-hover-black" href="http://127.0.0.1:5000/fabfile/example-fabfile/task/Ossec_succesful_logins_last/">Ossec succesful logins</a>
```

```
<a class="w3-bar-item w3-button w3-hover-black" href="http://127.0.0.1:5000/fabfile/example-fabfile/task/Ossec_check_the_systems/">Ossec syscheck</a>
```

```
<a class="w3-bar-item w3-button w3-hover-black" href="http://127.0.0.1:5000/fabfile/example-fabfile/task/Ossec_check_source_IP/">Ossec src-ip</a>
```

```
<a class="w3-bar-item w3-button w3-hover-black" href="http://127.0.0.1:5000/fabfile/example-fabfile/task/Deploy_ossec_agent_target/">Deploy ossec agent</a>
```

```
<a class="w3-bar-item w3-button w3-hover-black" href="http://127.0.0.1:5000/fabfile/example-fabfile/task/Start_ossec_agent/">Start ossec agent</a>
```

```
<a class="w3-bar-item w3-button w3-hover-black" href="http://127.0.0.1:5000/fabfile/example-fabfile/task/Stop_ossec_agent/">Stop ossec agent</a>
```

```
<a class="w3-bar-item w3-button w3-hover-black" href="http://127.0.0.1:5000/fabfile/example-fabfile/task/Status_ossec_agent/">Status ossec agent</a>
```

```
</div>
```

```
</div>
```

```
<div class="w3-dropdown-hover">
```

```
<button class="w3-button">Move Ossec agent <i class="fa fa-caret-down"></i></button>
```

```
<div class="w3-dropdown-content w3-bar-block">
```

```
<a class="w3-bar-item w3-button w3-hover-black"></a>
```

```
<a class="w3-bar-item w3-button w3-hover-black" href="http://127.0.0.1:5000/fabfile/example-fabfile/task/Move_ossec_honeypot1/">Move agent honeypot1</a>
```

```
<a class="w3-bar-item w3-button w3-hover-black" href="http://127.0.0.1:5000/fabfile/example-fabfile/task/Move_ossec_honeypot2/">Move agent honeypot2</a>
```

```
<a class="w3-bar-item w3-button w3-hover-black" href="http://127.0.0.1:5000/fabfile/example-fabfile/task/Move_ossec_honeypot3/">Move agent honeypot3</a>
```

```
</div>
```

```
</div>
```

```
<div class="w3-dropdown-hover">
```

```
<button class="w3-button">Conpot installation <i class="fa fa-caret-down"></i></button>
```

```

    <div class="w3-dropdown-content w3-bar-block"
      <a class="w3-bar-item w3-button w3-hover-black"></a>
      <a class="w3-bar-item w3-button w3-hover-black" href="http://127.0.0.1:5000/fabfile/example-
fabfile/task/Conpot_ubuntu_installation/">ubuntu</a>
      <a class="w3-bar-item w3-button w3-hover-black" href="http://127.0.0.1:5000/fabfile/example-
fabfile/task/Conpot_centos_installation/">centos</a>
      <a class="w3-bar-item w3-button w3-hover-black" href="http://127.0.0.1:5000/fabfile/example-
fabfile/task/Conpot_debian_installation/">debian</a>
    </div>
  </div>
  <div class="w3-dropdown-hover">
    <button class="w3-button">Install Dependencies <i class="fa fa-caret-down"></i></button>
    <div class="w3-dropdown-content w3-bar-block"
      <a class="w3-bar-item w3-button w3-hover-black"></a>
      <a class="w3-bar-item w3-button w3-hover-black" href="http://127.0.0.1:5000/fabfile/example-
fabfile/task/Dependencies_in_ubuntu/">ubuntu</a>
      <a class="w3-bar-item w3-button w3-hover-black" href="http://127.0.0.1:5000/fabfile/example-
fabfile/task/Dependencies_in_centos/">centos</a>
      <a class="w3-bar-item w3-button w3-hover-black" href="http://127.0.0.1:5000/fabfile/example-
fabfile/task/Dependencies_in_debian/">debian</a>
    </div>
  </div>
  <div class="w3-dropdown-hover">
    <button class="w3-button">Change MAC<i class="fa fa-caret-down"></i></button>
    <div class="w3-dropdown-content w3-bar-block"
      <a class="w3-bar-item w3-button w3-hover-black"></a>
      <a class="w3-bar-item w3-button w3-hover-black" href="http://127.0.0.1:5000/fabfile/example-
fabfile/task/Change_target_MAC_Siemens/">Siemens MAC</a>
      <a class="w3-bar-item w3-button w3-hover-black" href="http://127.0.0.1:5000/fabfile/example-
fabfile/task/Change_target_MAC_kamstrup/">Kamstrup MAC</a>
      <a class="w3-bar-item w3-button w3-hover-black" href="http://127.0.0.1:5000/fabfile/example-
fabfile/task/Change_target_MAC_guardian/">Guardian MAC</a>
    </div>
  </div>
</nav>
<!-- Overlay effect when opening sidebar on small screens -->
<div class="w3-overlay w3-hide-large" onclick="w3_close()" style="cursor:pointer"

  title="close side menu" id="myOverlay"></div>
<!-- Main content: shift it to the right by 250 pixels when the sidebar is visible -->

<div class="w3-main" style="margin-left:200px">
  <div class="w3-row w3-padding-64">

```



```

<div class="w3-twothird w3-container">
  <h1 align="left" >Choose a task to be executed in a remote system:</h1>

  {% with messages = get_flashed_messages() %}
  {% if messages %}
    <div class="alert alert-error">
      {% for message in messages %}
        <li>{{ message }}</li>
      {% endfor %}
    </ul>
  </div>
  {% endif %}
  {% endwith %}

  {% block content %}{% endblock %}
</div>
<div class="w3-third w3-container">
  <p class="w3-padding-large w3-padding-32 w3-center"></p>
  <!--<p class="w3-padding-large w3-padding-64 w3-center"></p-->
</div>
</div>
<div class="w3-row">
  <!--<div class="w3-twothird w3-container">
    <h1 class="w3-text-teal">Heading</h1>
    <p>
</p>
</div>
  <!--<div class="w3-third w3-container">
    <p class="w3-border w3-padding-large w3-padding-32 w3-center">AD</p>
    <p class="w3-border w3-padding-large w3-padding-64 w3-center">AD</p>
  </div>
</div>
<div class="w3-row w3-padding-64">
  <div class="w3-twothird w3-container">
    <h1 class="w3-text-teal">Heading</h1>
    <p>
</p>
</div>-->
  <!--<div class="w3-third w3-container">
    <p class="w3-border w3-padding-large w3-padding-32 w3-center">AD</p>

```

```

    <p class="w3-border w3-padding-large w3-padding-64 w3-center">AD</p>
  </div>-->
</div>
<!-- Pagination -->
<div class="w3-center w3-padding-32">
  <div class="w3-bar"> <a class="w3-button w3-black" href="#">1</a> <a class="w3-button w3-
hover-black"
  href="#">2</a> <a class="w3-button w3-hover-black" href="#">3</a> <a
  class="w3-button w3-hover-black" href="#">4</a> <a class="w3-button w3-hover-black"
  href="#">5</a> <a class="w3-button w3-hover-black" href="#">></a> </div>
</div>
<footer id="myFooter">
  <div class="w3-container w3-theme-l2 w3-padding-32">
    <h4>Pythonic remote execution</h4>
  </div>
  <div class="w3-container w3-theme-l1">
    <p>Powered by <a href="https://github.com/bitprophet/alabaster" target="_blank"> Sphinx
1.6.7 & Alabaster 0.7.11 | <a href="https://github.com/daniellawrence/fabric-web" target="_blank">
Fabric-web |<a href="http://www.fabfile.org/" target="_blank"> Fabric</a></a></a></p>
  </div>
</footer>
<!-- END MAIN --> </div>
<script>
// Get the Sidebar
var mySidebar = document.getElementById("mySidebar");

// Get the DIV with overlay effect
var overlayBg = document.getElementById("myOverlay");
// Toggle between showing and hiding the sidebar, and add overlay effect
function w3_open() {
  if (mySidebar.style.display === 'block') {
    mySidebar.style.display = 'none';
    overlayBg.style.display = "none";
  } else {
    mySidebar.style.display = 'block';
    overlayBg.style.display = "block";
  }
}
// Close the sidebar with the close button
function w3_close() {
  mySidebar.style.display = "none";
  overlayBg.style.display = "none";
}

```

```

}
</script>
</body>
</html>

```

### Index.html

```

{% extends "base.html" %}
{% block content %}
{% if task_list %}
<ul>
  {% for fabfile_name, fabfile in task_list.items() %}
  <Br />
  <!-- <h3">
    <a href='/fabfile/{{fabfile.alias}}'>{{fabfile.alias|replace('-', ' ')|title}}</a>
  </h3>
  <pre>{{fabfile.doc}}</pre>-->
  {% for task_name, task in fabfile.tasks.items() %}
  <div style="float: left; width: 33%; font-size:0.8vw;"><ul><strong><p class="big"><div class="w3-
container"><button class="w3-button w3-white w3-border w3-round-large"><a
href='/fabfile/{{fabfile.alias}}/task/{{task_name}}/'>{{task_name|replace('_', '
')}}</a></div></p></strong></ul></button></div>
  {% endfor %}
  {% endfor %}
</ul>
{% endif %}
{% endblock %}

```