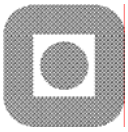
# Identifying and Responding to External Threats in a PCS Network

Zi bin Cheah A. B. M. Omar Faruk
**{cheah, faruk}@stud.ntnu.no**

**Department of Telematics**
**Norwegian University of Science and Technology**

**NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**FACULTY OF INFORMATION TECHNOLOGY, MATHEMATICS AND ELECTRICAL ENGINEERING**



# PROJECT ASSIGNMENT

**Student's name:** A. B. M. Omar Faruk, Zi Bin Cheah

**Course:** TTM4530 (Information Security, Specialization Project)

**Title:** Identifying and Responding to External Threats in a PCS Network

**Text:**

The assignment is a traditional "red team/blue team" exercise in the process control domain. We, the blue team, will configure a test network to simulate a process control system (PCS) and connected computer networks on an offshore oil installation, including mechanisms for remote access and collaboration with an onshore operations center. Security mechanism such as Intrusion Detection System, Firewall and Honeynet will be part of the test network-setup.

After having configured the security countermeasures, the objective of this project will be to detect and respond to external attacks performed by the red team, i.e. proper mechanism to log and respond to attacks launched is to be developed.

**Deadline:** 19-12-2007

**Handed in:**

**Carried out at:** Department of Telematics

**Supervisor:** Svein J. Knapskog

**Trondheim, …….. 2007**

**Professor**

# Abstract

This project is a traditional "red team/blue team" exercise in the process control domain. We, the blue team, will configure a test network to simulate a process control system (PCS) and connected computer networks on an offshore oil installation, including mechanisms for remote access and collaboration with an onshore operations center. Security mechanisms such as intrusion detection and prevention system, firewall and honeypot will be part of the test network-setup.

The first milestone is to setup a PCS network and its proper security. After the network is up, the second milestone of this project will be to detect and respond to external attacks performed by the red team, i.e. proper mechanisms to log and respond to attacks launched are to be developed. The red team is specialized in discovering vulnerabilities in our PCS network.

The objective of this project is not to setup a super-secured PCS network. We will seek to implement common security mechanisms available on most PCS networks. Red team should be able to attack the network so that we will perform incident handling and response.

# Acknowledgement

Working with this project has been an excellent opportunity to acquire new knowledge on PCS network, PCS security and honeypots. It has been a good learning experience putting theory into work in this project. Combining theory and practice have been motivating and fun.

SINTEF is the largest independent research organization in Scandinavia. We would like to thank our supervisors at **SINTEF**, **Maria Bartnes Line** and **Martin Gilje Jaatun** from the Department of Software Engineering, Safety and Security for proposing this assignment and for their valuable input and guidance during the project.

A substantial part of project is based on <The SeSa Method for Assessing Secure Remote Access to Safety Instrumented Systems> written by Tor Olav Grøtan, Martin Gilje Jaatun, Knut Øien and Tor Onshus. We would like to extend our gratitude with the authors of the paper.

We would also like to thank our professor **Svein Knapskog** from the Telematics Department of NTNU for arranging this wonderful project with SINTEF.

We would also like to thank **red team** for the collaboration effort.

# Table of Contents

# List of Figures

# List of Tables

# Abbreviations

**COM** – Component Object Model

**DCOM** – Distributed Component Object Model

**DCS** – Distributed Control System

**DHCP** – Dynamic Host Configuration Protocol

**DMZ** – Demilitarized Zone

**DoS** – Denial of Service

**DOS** – Disk Operating System

**DNS** – Domain Name System

**HTTPS** – Hyper Text Transfer Protocol Secure

**HTTP** – Hyper Text Transfer Protocol

**ICMP** – Internet Control Message Protocol

**IDS** – Intrusion Detection System

**IP** – Internet Protocol

**IPS** – Intrusion Prevention System

**IPSec** – IP Security

**IR** – Incident Response

**IRMA** – Incident Response Management

**NAT** – Network Address Translation

**NIC** – Network Interface Card

**NIST** – National Institute of Standards and Technology

**NVD** – National vulnerability database

**OLE** – Object Linking and Embedding

**OPC** – OLE for Process Control

**OPC DA** – OPC Data Access

**OSVD** – Open Source Vulnerability Database

**PCS** – Process Control System

**PLC** - Programmable Logic Controller

**RDP** – Remote Desktop Protocol

**RPC** – Remote procedure call

**RTU** - Remote Terminal Unit

**SCADA** – Supervisory Control and Data Acquisition

**SSH** – Secure Shell

**TCP** – Transmission Control Protocol

**UDP** – User Datagram Protocol

**US-CERT** – United States Computer Emergency Readiness Team

# 1. Introduction

A brief overview of the project is explained.

## 1.1 Background

Industrial processes are vital for the industry they serve. These industries range from power, oil, water and others. These processes usually require high precision and coordination and are housed in highly protected areas.

As these industries evolve, industrial process controls become more and more important. Process Control System (PCS) **[1]** are given great attention. Traditionally, these systems are highly guarded and are based on proprietary technology. Attacks are seldom and far in between as an attacker might need extensive knowledge about the technology and physical and/or logical access to break into it. This is made even harder when PCS are isolated from the outside world. There is no entry point to mount an attack as the network is not connected to anything other than components inside the network.

Even when not subject to attacks, we have heard in the news that malfunction of these processes can cause catastrophic disaster such as the Chernobyl leak. It is not envisaged that anything as serious as that might happen in the near future, but it demonstrates that ill-intentioned attack might have serious consequences.

The trend for PCS networks is to connect to the Internet and to be based on common open standards. All this will gear PCS towards higher efficiency, but at the same time extremely exposed to attacks. Now attackers can, through the Internet, probe the PCS network and launch attacks towards it.

OLE for Process Control (OPC) **[2]** is a popular mechanism to exchange process control data among numerous data sources in PCS network. OPC is implemented in a client-server architecture using COM/DCOM **[3][4]** technology by Microsoft. COM/DCOM vulnerabilities and threats are used to spread virus and other malware. Most of the vendors are still using COM/DCOM based OPC server and client in PCS network. It is important to prevent and detect the threats and vulnerabilities due to OPC implementation.

It will be important to identify these threats mounted from the Internet towards the PCS network. Incident response is also important to make sure that when an attack takes place, we can deploy the best way to escalate warning and minimize damage. Incident handling will also help us to mitigate future attacks.

## 1.2 Problem Statement

The security of PCS has come into question as they are increasingly seen as extremely vulnerable and enticing for attackers.

There are several weaknesses that make such systems susceptible **[5]**.

- The lack of concern about security and authentication in the design, deployment and operation of existing networks.
- The mistaken belief that they have the benefit of security by obscurity through the use of specialized protocols and proprietary interfaces.
- The mistaken belief that they are secure because they are supposedly physically secured
- The mistaken belief that they are secure because they are supposedly disconnected from the Internet.
- Knowledge gap between IT professionals and process control professionals is also a factor. Both have different understanding expertise when it comes to processes and technology.

Some of the aforementioned protections are already defeated. PCS networks are shifting towards a standardized implementation of protocols and interfaces are demonstrated by OPC. Also, systems nowadays are becoming more interconnected to the Internet. These are signs that the “barrier-to-intrude” becomes lower by the day.

To make matters greyer, there could be extra incentive for the blackhat hacking community to penetrate and disrupt PCS systems. By attacking these networks, the damages are more “real” as power, oil, water, traffic, amongst others become interrupted.

### 1.2.1 Problems by example

The following example illustrates the problems that an unintentional error in a PCS can cause. **[6]**

#### CSX Train Signaling System

*In August 2003, the Sobig computer virus was blamed for shutting down train signaling systems throughout the East Coast of the United States. The virus infected the computer system at CSX Corporation's Jacksonville, Florida, headquarters, shutting down signaling, dispatching, and other systems. According to an Amtrak spokesman, 10 Amtrak trains were affected. Train service was either shut down or delayed up to 6 hours.*

#### Davis-Besse power plant

*The Nuclear Regulatory Commission confirmed that in January 2003, the Microsoft SQL Server worm known as Slammer infected a private computer network at the idled Davis-Besse nuclear power plant in Oak Harbor, Ohio, disabling a safety monitoring system for nearly 5 hours. In addition, the plant's process computer failed, and it took about 6 hours for it to become available again.*

The unintentional incidents above shows that if malicious attacks are targeted at the same system, the attacker might be able to create the same outcomes as elaborated by the two example above.

Furthermore, we have come to realized that malicious attacks on SCADA/PCS are happening. As these systems become more connected, they become more and more exposed. Below we present several actual attacks that are targeted at SCADA / PCS.

#### Maroochy Shire Sewage Spill

*In the spring of 2000, a former employee of an Australian organization that develops manufacturing software applied for a job with the local government, but was rejected. Over a 2-month period, this individual reportedly used a radio transmitter on as many as 46 occasions to remotely break into the controls of a sewage treatment system. He altered electronic data for particular sewerage pumping stations and caused malfunctions in their operations, ultimately releasing about 264,000 gallons of raw sewage into nearby rivers and parks.*

#### Los Angeles Traffic Lights

*According to several published reports, in August 2006, two Los Angeles city employees hacked into computers controlling the city's traffic lights and disrupted signal lights at four intersections, causing substantial backups and delays. The attacks were launched prior to an anticipated labor protest by the employees.* **[7]**

#### Worcester Air Traffic Communications

*In March 1997, a teenager in Worcester, Massachusetts, disabled part of the telephone network using a dial-up modem connected to the system. This disabled phone service to the airport control tower, airport security, the airport fire department, the weather service, and the carriers that use the airport. Also, the*

*tower's main radio transmitter and another transmitter that activates runway lights were shut down, as well as a printer that controllers use to monitor flight progress. The attack also disrupted phone service to 600 homes in a nearby town.*

### Harrisburg, Pennsylvania, Water System

*In October 2006, a foreign hacker penetrated security at a water filtering plant. The intruder planted malicious software that was capable of affecting the plant's water treatment operations. The infection occurred through the Internet and did not seem to be an attack that directly targeted the control system.* **[8]**

As shown, attacks on PCS are very real. The unintentional failure of PCS as mentioned (by the first two incidents) shows the varieties of attacks that might take place on such systems. Other examples (subsequent four incidents) prove that attacks are already happening and the consequences can be lethal. It is necessary and important to protect train service, traffic signaling system, air traffic service, water plant from different types of threats and attacks. These crucial systems should at least prepare themselves to detect or prevent known attacks.

## 1.3 Project Planning

Figure below depicts the overall project progress that we have embarked on.

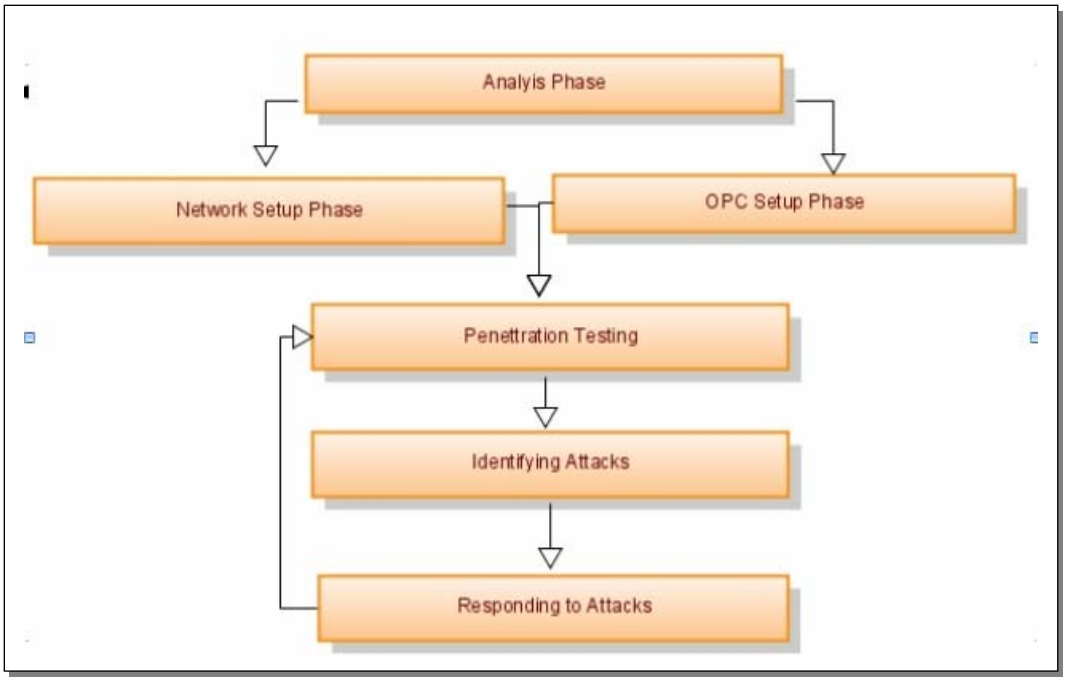



**Figure 1 : Overall Project Flow**

## 1.4 Report Structure

Section 2 gives a theoretical background on components that we will be working on. We seek to explain in brief important concepts relevant in our project. Among them are:

- Process Control System (PCS)
- Object Linking and Embedding (OLE) **[9]** for PCS
- Honeypots
- Intrusion Detection and Prevention, Firewall
- Threats
- Incident handling

Section 3 is the laboratory setup of the PCS network. We setup the network in SINTEF's **[10]** lab. We will try to understand the topology and configurations involved.

Section 4 shows the known weaknesses if the services offered PCS network. We used three different vulnerability databases for this purpose. We also discussed vulnerabilities exploited by red team.

Section 5 is about how we prepared, detected and responded to different incidents launched by red team and external attackers. Mainly we considered the technical aspects of bringing the services attacked back to service.

# 2. Theory of Components

Technologies relevant to our project are presented here.

## 2.1 Process Control

Process control is a discipline that deals with controlling the output of a specific industrial process. For example, a process control can include the controlling of the heating of a room temperature. This process has the specific, desired outcome to reach and maintain a defined temperature, say 15 degrees. Here, the temperature is the controlled variable. At the same time, it is the input variable since it is measured by a thermometer and used to decide whether to heat or not to heat. The desired temperature of 15 degrees is the set point. The state of the heater (e.g. the setting of the valve allowing hot water to flow through it) is called the manipulated variable since it is subject to control actions.

Larger and more complex systems can be controlled by a Distributed Control System (DCS) **[11]** or SCADA system. SCADA is the acronym for "Supervisory Control and Data Acquisition". SCADA can be referred as a large-scale, distributed process control system. SCADA systems are typically used to perform data collection and control at the supervisory level **[5]**.

## 2.2 OLE for Process Control (OPC)

OLE for Process Control (OPC) **[2]** provides a standard mechanism to exchange process control data among numerous data sources. For example, many different data sources such as PLC, DCS, database and other devices are used in process control system. Data is gathered from different connections (e.g. serial, Ethernet) and different operating systems (e.g. Windows, UNIX and DOS). The OPC standard hides the complexities of the device-dependent communication protocols (e.g. Modbus **[12]**) which ensure interoperability. It also reduces implementation cost and helps to build a scalable system **[13]**. In the past, vendors gathered different process control data in their applications using their own device interfaces. Format of the data was only known and understood by the applications of specific vendor. So, users were forced to return to that vendor whenever they needed a change or extension in the system. In contrast, process control data can be collected from any data source to any OPC compliant application using the common set of interfaces defined by OPC. OPC Foundation **[14]**, a non-profit international organization is responsible for maintaining the specification of common set of interfaces. OPC data access (OPC DA) is one of the commonly used specifications which standardize real-time data access **[15]**.

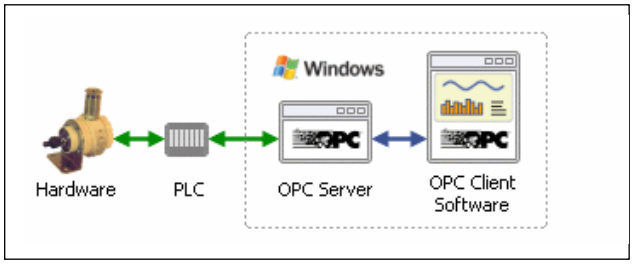



**Figure 2 : OPC Client-Server Architechture [16]**

Microsoft's COM **[4]** and DCOM **[3]** technology is utilized by OPC to exchange data using client-server architecture. The **OPC server** is an application which converts the hardware communication protocol used by a device or data source into the OPC protocol. OPC server implements COM/DCOM objects and their interfaces according to the distinct OPC specification. The **OPC client** is a program that needs to connect to the device or data source to retrieve data for monitoring, analyzing and planning a process control system **[16]**. An OPC Client can connect to OPC Servers provided by one or more vendors (e.g. Matrikon **[17]**, Cogent Real-Time Systems **[18]**). Different connection scenario for OPC clients and servers are summarized below **[16]**.

1) **OPC Aggregation:** Single OPC client connects to multiple OPC servers.
2) **OPC Tunnelling:** Single OPC client connects to an OPC server without using DCOM.
3) **OPC Bridging:** Connecting two OPC servers to share each other's data.

## 2.3 Honeypot

Honeypots are host(s) setup to detect and trap attackers. Often a computer, honeypots can be any devices connected to the network including printers, PDAs and also virtualization services.

The primary purpose of a honeypot is to gather information on threats. This information has different value to different organizations. For example, academic research institutions may use honeypots to gather data for research, such as worm activity. Security organizations may use honeypots to capture and analyze malware for anti-virus, IDS signatures or learn new ways to counter changing threats. Government organizations may use honeypots to learn more about who is targeting them or why. ISP's may use honeypots to capture, analyze, and terminate botnets that are on their networks. Security responders can use honeypots for incident response, collecting information on compromised systems.

We illustrate honeypot concept with a imaginary network topology. In this imaginary network, we have three honeypots. Since honeypots are not production systems, inbound traffic to these honeypots are highly suspicious. Outbound traffic is also worth analyzing since honeypots do not produce traffic as they are not production systems, so outbound traffic could be trigger by an attacker who has successfully control our honeypots.

The gateway in between the honeypot's network and the production network are setup to capture data that goes into the honeypot subnetwork. This gateway is commonly regarded as a honeywall.

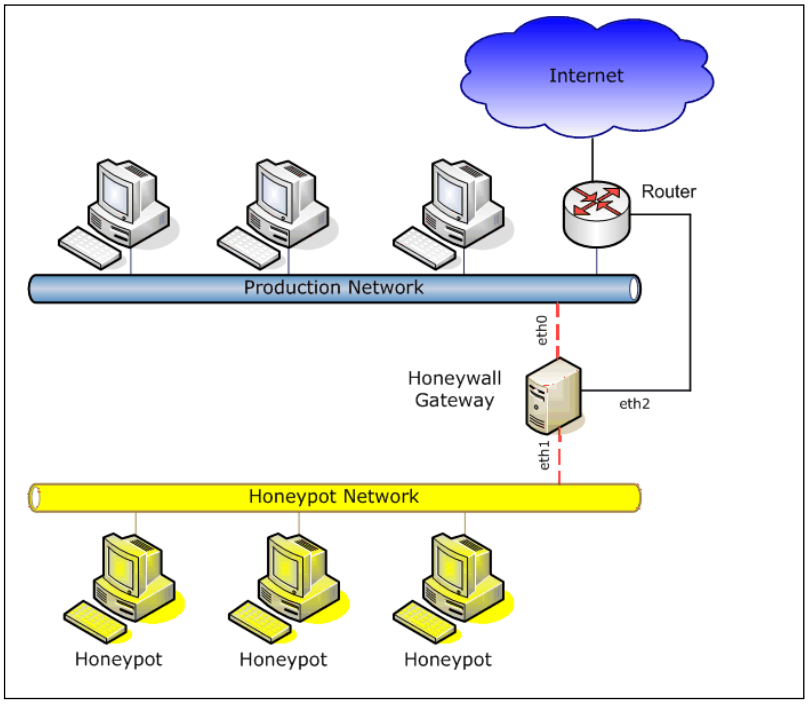



**Figure 3 : Imaginary topology with honeywall and honeypots**

Based on our research we observed a few important principles of honeypot -

- **A honeypot is not a production system**
- **Every flow going to (or coming from) honeypots are suspicious by nature.**
- **Honeypots' trap must be difficult to recognize by a potential hacker.**
- **Honeypot can choose to hide amongst production systems**

There are many ways to implement honeypots. We will be using software from the open source project called the honeynet project to setup our honeypots.

## 2.4 Honeywall

Honeywall is a host that is placed between honeypot(s) and non-honeypot components in a network. It is regarded as a “bridge” between honeypots and the rest.

This “bridge” appears to be invisible to anyone looking at the logical setup of the network. Attackers and everyone else do not know the existence of the honeywall inside of the network. This anonymity is achieved by, amongst other things, having no IP in its interface. Honeywall setup is an important setup of the honeypot framework. In theory, no one should know the existence of this “bridge” unless he or she walks into our laboratory and sees the honeywall computer.

We will be using tools from the Honeynet Project **[19]** to create our honeywall. The Honeynet project is an open source project that encourages the development of honeypots activities.

## 2.5 Intrusion Detection

An intrusion detection system (IDS) generally detects malicious behaviors that can compromise the security and trust of a computer network. This includes network attacks against vulnerable services, data driven attacks on applications, host based attacks such as privilege escalation, unauthorized logins and access to sensitive files, and malware.

When an attack is detected, IDS may choose a multitude or response. It can simply record down the attack in a log. It can also email or text messages to the administrator. Some IDS implement prevention mechanism to try to drop traffic that is seemed as malicious before any harm is done. Sometimes, an IDS application that implements this preventive measure is regarded as an Intrusion Prevention System (IPS). Many people see it pointless to argue if IPS or IDS is the better name.

Generally, IDS is differentiated into network-based IDS and host-based IDS. Network-based IDS monitors the network and tries to detect attacks on the TCP/IP layer. Host-based IDS are more fine-grained as it is installed on the computer that it seeks to protect. Host-based IDS will be able to look at traffic packets on a finer level, sometimes at the application level. They usually serve different detection purpose. For example network-based IDS may be able to detect a denial of service attack on a network but not host-based IDS. On the other hand, host-based IDS can perform tasks such as monitoring encrypted packets since packets are decrypted after it enters the computer.

One plus point about network-based IDS is that it can be installed by the administrator and protects the whole network, while host-based IDS has to be installed by the user on their computer and eats up the user's computing resources.

## 2.6 Firewall

Firewall is needed by a network to block out malicious attempts. The difference between a firewall and an IDS is that an IDS allows connection/data to come in but scans the data to see if they are malicious. As for firewall, they will choose to deny the entry of a data altogether (even before scanning if the content of the data is malicious). The denial can be made based on packet ports, IP and other measurements.

Generally, there are two types of firewalls. The first is the network layer firewall. This type of firewall operates on a TCP/IP layer filtering. It will deny or allow packets based on rulesets that checks on the traffics' TCP/IP layer information. The second type is application layer firewall. An application layer firewall may inspect contents of the traffic and perform denial based on contents directed at different computer applications. These applications include browser and telnet.

Firewalls often have network address translation (NAT) functionality, as it allows computers behind a network to use private addresses. Firewalls often have such functionality to hide the true addresses of protected hosts. Hiding the addresses of protected devices has become an increasingly important defense against reconnaissance activities by attackers.

Based on our network configuration, we will have two firewalls. The first one resides in the gateway and the other in the honeywall. They play different roles, and sometimes can be confused together

## 2.7 iptables

iptables is our choice of firewall implementation. We will be running iptables in honeywall and the gateway. Therefore, we have two firewalls in our network. iptables are preinstalled in many Linux Distributions.

Firewall not only blocks traffic, but shapes traffic too. iptables is extremely flexible; we can do many things with it. For example, we can deny or allow traffic based on ports, host and IP. For our network we use the *filter table* and *NAT table* in iptables.

### 2.7.1 Filter Table

As the name suggests, filter table filters out traffic based on different criteria. These criterias include IPs, Ports and Protocols.

This table decides which packets to accept or drop. The general policy for *filter table* is either "*Default Accept"* or "*Default Deny"*. *"Default Accept" allows* all packets to enter the network unless specified otherwise by the administrator. *"Default Deny",* on the other hand, denies all packets unless allowed by the administrator. Policy-implementation are always different for different organizations.

### 2.7.2 NAT Table

NAT (Network Address Translator) **[20]** table has a list of entries as to how IP addresses are changed when they enter or exit the network.

Internal host using different private IP addresses will be able to change their IP addresses to a single public IP before leaving the Internet. Packets from the Internet will, in return, be changed into private IP addresses as it goes through the gateway. This is all the effort of iptables NAT capability. The *NAT table* allows us to use multiple hosts for a single public IP and also to hide these hosts.

## 2.8 Threats

Security threats towards a PCS network can come from within and outside of the network. A threat that is coming from inside the network is regarded as an internal threat, while attacks from the outside are referred as external threats.

Internal threats can be mounted by a disgruntled employee or a mole from a competitor. These attackers always have understanding network topology and the security mechanism laid out to protect the network.

External attacker usually has little knowledge of the network and mounts the attack from outside of the network. Before an external attack can happen, Reconnaissance such as trying to understand the network from the outside will be performed for a period of time.

Activities such as network scanning, port scanning will take place to gather information. An external attacker can also perform social engineering such as calling into the PCS network personnel to solicit for information. Social engineering is a powerful aspect of external attacker as employees often unknowingly leak information about the organization.

There are multitudes of threats an attacker can seek to achieve. We will simply briefly mention it. Amongst other, an attacker can shutdown a service, takeover a network, or compromise the authenticity of information in the network.

## 2.9 Attacks and Incidents

According to Howard and Longstaff, "**Attack** is a series of steps taken by an attacker to achieve an unauthorized result. An attacker uses a ***tool*** to exploit a ***vulnerability*** to perform an ***action*** on a ***target*** in order to achieve an ***unauthorized result***". Steps usually used to launch an attack are described briefly according to the figure below **[21]**.

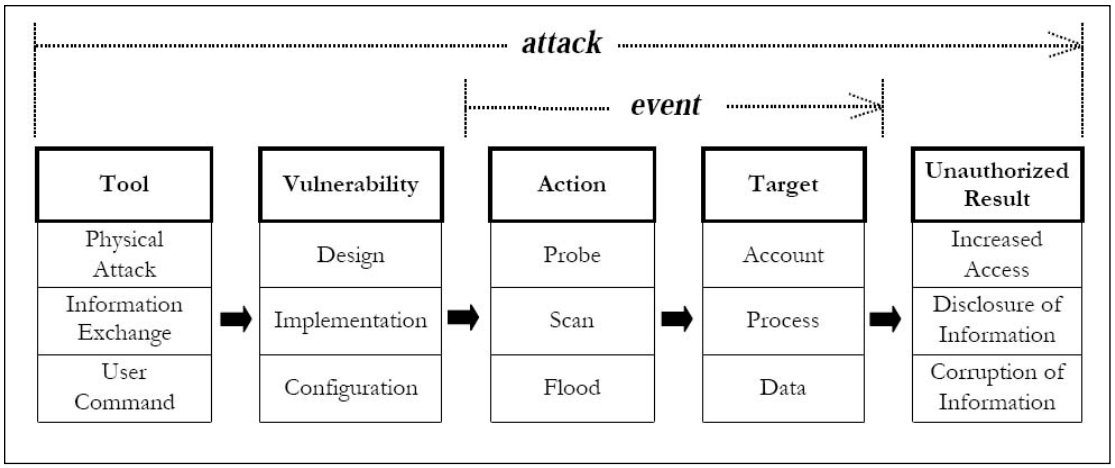



**Figure 4 : Computer and Network Attacks [21]**

The first step to start the attack is to find and use the ***tools*** of attack which can be used to exploit vulnerabilities in a computer or network. Tools can be simple (e.g. user command) or very sophisticated like Trojan horse, computer virus etc. **[21]**

Attacker needs to take advantage of a computer or network ***vulnerability*** to do an unauthorized action. Krsul indicated that vulnerabilities in software are introduced in different stages of software development. It can be introduced at design, implementation or configuration time. **[22]**

To exploit the vulnerabilities of the system, attacker needs to perform an ***action*** or set of actions (e.g. probe, scan, flood) to change the state of one or more ***target***(s). Target can be logical entities (e.g. account, process) or physical entities (e.g. computer, network). **[21]**

Finally, ***unauthorized result*** implies to the last step of a successful attack. Some examples of unauthorized result are : unauthorized use of computer or network resources, unauthorized modification of data etc. **[21]**

Howard and Longstaff defined an incident as a group of attacks that can be distinguished from other attacks because of the distinctiveness of the *attackers*, *attacks*, *objectives*, sites, and timing **[21]**. *Attacker* usually attempts one or more attacks in order to achieve an objective. The report **[23]** from NIST (National Institute of Standards and Technology) defined an incident as a violation or forthcoming threat of violation of computer security policies, acceptable use policies, or standard security practices. Examples of such incidents are: Denial of Service, malicious code etc.

# 3. Network Design

In this chapter we described equipments, network topology and tools we used in laboratory to simulate PCS network. Network interface card configuration, routing configuration are discussed in this section. Finally, OPC server and OPC client configuration is illustrated.

In chapter 4, we will proceed with the actual setup process.

## 3.1 Equipment Used

We have mainly used computers with Linux operating system. Single machine was used with Windows XP operating system which hosts the OPC server. To reduce the amount of I/O devices, two computers share screen, mouse and keyboard using two 2 port KVM switches **[24]**.

Both gigabit (e.g. **Intel 82540EM Gigabit Ethernet Controller**) and fast Ethernet (e.g. **3Com 3C905C Fast Ethernet Controller**) network cards were used as NIC. Different LAN segments are connected through a 3COM Office Connect 10/100MB hub. Following table shows the summary of operating system and hardware specifications of the computers used in the laboratory.

| Host Name | Operating System | Processor | RAM | No of NIC |
|---|---|---|---|---|
| Internet-dmz | Linux Fedora Core 6 | Pentium 4 2GHz | 512MB | 3 |
| router | Linux Fedora Core 6 | AMD Athlon 1.5GHz | 512MB | 3 |
| dmz-admin | Linux Fedora Core 6 | Pentium 4 2GHz | 512MB | 2 |
| dmz-host1 | Linux Fedora Core 6 | Pentium 4 2GHz | 512MB | 1 |
| admin-process | Linux Fedora Core 6 | AMD Athlon 1.5GHz | 512MB | 2 |
| process-host1 | Windows XP SP2 | Pentium 4 2.5GHz | 512MB | 1 |

**Table 1 : Computer Hardware specification**

Brief descriptions about the role and NIC of different machines are given below.

- **Router and Honeywall:** Machine with hostname **'router'** is placed in the border of the network to route traffic between external and internal networks using two NICs. Another NIC is used to connect router with the management interface of Honeywall. Machine with hostname **'Internet-dmz'** plays the role of a Honeywall in which three NICs are configured.
- **Internal Gateways:** There are two gateways in the network to divide the network into subnetworks. Gateway with hostname **'dmz-admin'** is placed between DMZ and admin network. Gateway with hostname **'admin-process'** is placed between admin and process network. Both of them use two NICs.
- **Other Hosts:** One host with hostname 'dmz-host1' in DMZ network simulates the DNS or DHCP server in the PCS network. Another host 'process-host1' is placed in process network where OPC server is configured. Both of them use single NIC.

## 3.2 Network Topology

The diagram **(figure 5)** below depicts the network setup of our network. The network is a typical off shore network setup which is extracted from the report **[1]** discussing with SeSa method. The full network topology proposed by that report is attached in **Appendix A**. Some off shore networks might be disconnected from the outside world and thus do not have Internet connection. In such instances, a breach of network security can only come from within the network that is done by an insider. We are interested to look at external threats; Internet is a good entry point for external threats.

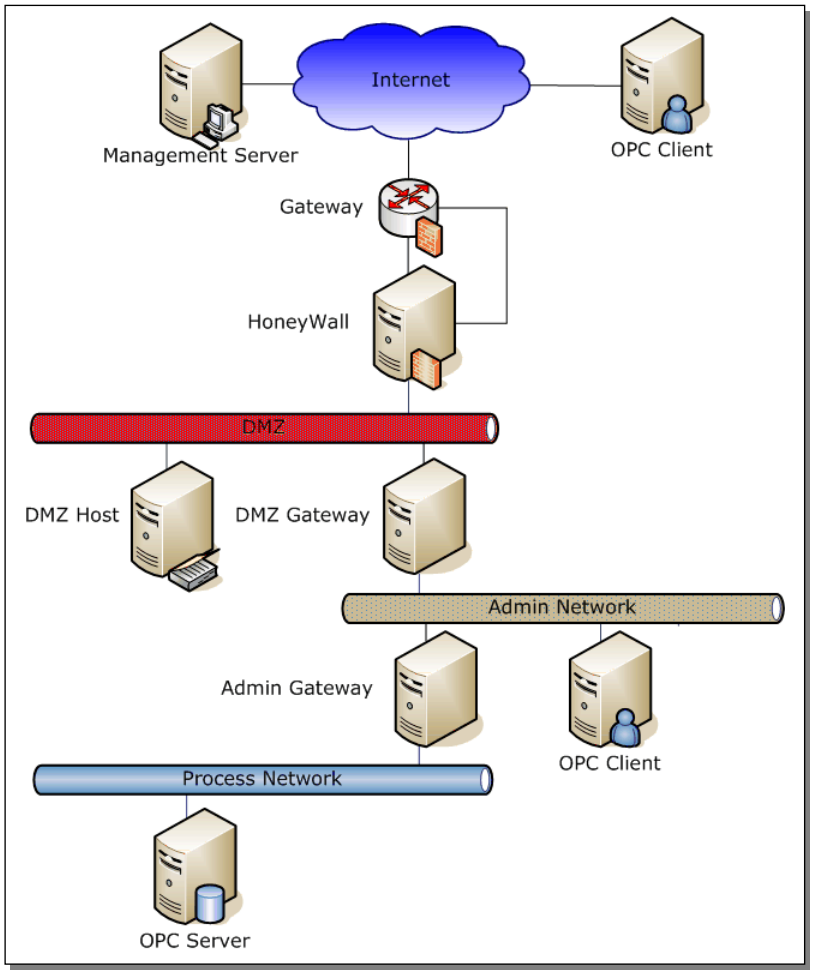



**Figure 5 : Network topology used in the lab**

In this laboratory setup, it is possible that the breach can come from within and externally. For an external attack to happen, we will need to connect this network to the Internet. Host such as the OPC client can try to connect remotely from the Internet. Our network provides open port (21379) for OPC client to connect to our OPC server using OPC tunnelling. Other than that, services such as SSH **[25]** and HTTPS **[26]** are also open for remote connection. An attacker might exploit such open services and try to attack remotely from the Internet.

Our network is divided into three subnetworks, they are

1) **<DMZ network>** subnet ( with a DHCP host)
2) **<Admin Network>** subnet (with an OPC client)
3) **<Process Network>** subnet ( with an OPC server)

The reason for such a segmenting is both ease of management, as well as better attack containment. We will first discuss about the three subnetworks inside our network.

Apart from the three main subnets, there are another two important components in our network are:

1) **Network Gateway**
2) **Honeywall**

The characteristics of a gateway and honeywall can be easily confused. Below is a simplified breakdown of their functions shown in table.

| Gateway | Honeywall |
|---|---|
| Internet Entry point | Honeypot packets collection |
| Visible in the network | Invisible in the network |
| Runs iptables | Run iptables |
| iptables serves as firewall and NAT | iptables serves as IPS (using snort_inline) |

**Table 2 : Functions of Gateway and Honeywall**

## 3.3 DMZ network

DMZ is an abbreviation for **d**e**m**ilitarized **z**one. In common computer networks, a DMZ contains an organization's services that are open/semi-open to the access of public or a large part of the network. Services in a DMZ are usually needed by the external host in the Internet to communicate successfully with hosts inside the network and vice versa. Services that are highly sensitive are never put in DMZ. The purpose of DMZ is to add an additional layer of security to an organization's network by having services that are always accessed housed in this layer. DMZ is setup in our network for the same reason.

Inside our DMZ, we house a DHCP **[27]** host that allows dynamic IP distribution to host inside the network. Of course, DMZ's function spans beyond this. In a usual network setup, HTTP **[28]** servers, printers are some of the many services housed in a DMZ. Remember that services that are housed in DMZ are usually services that can be openly/semi-openly access by the public (or a large part of the network).

We only setup one DHCP host in the DMZ. Dynamic IP distribution is actually the secondary reason the DHCP host is setup for. The main reason is actually to provide a victim in the DMZ layer of our network. An attacker might succeed in compromising the DMZ layer but not layers beyond the DMZ. By layering our network in subnetworks we can effectively track the level of attack and also limit the spread of attack if we choose to.

Firewall 2 in the DMZ serves as a filter between DMZ network and Admin Network. In order for an attacker to infiltrate into Admin network, he first has to compromise the Firewall 2.

## 3.4 Admin Network

This is the second layer of our subnetwork. This layer models a typical off shore network setup whereby hosts in the Admin Network are privileged users that usually have access to the Process Network layer. In our setup, there will be an OPC client and a Firewall in the Admin network.

In order for users in the Admin network to access the Process Layer services, they should have the proper credentials to do so. For example, a manager in the Admin layer might use OPC client software such as a MatrikonOPC Client to access information from an OPC server (which is in the Process Layer). In order for the manager to successfully do this, he probably has to have the proper login/password credentials.

If an attacker gets to this layer, he might be able to take control of hosts inside Admin Network; however, he might still need considerable effort and skills to steal login/password credentials of the manager to reach the final layer (Process layer). Of course, another way an attacker can do is to try to break the firewall in the Admin network and gain access to the Process layer.

## 3.5 Process Network

This is the deepest layer of the network and houses the most sensitive services in an off-shore network. We have setup an OPC server based on a free demo of Matrikon OPC server **[29]**. If an attacker takes over this network, he has basically succeeded in breaching the entire network.

OPC server is a software application that acts as an API (Application Programming Interface) or protocol converter. An OPC Server will connect to a device such as a PLC **[30]**, DCS **[11]** etc or a data source such as a database, User interface, etc and translate the data into a standard-based OPC format. OPC compliant applications such as a spreadsheet, trending application, etc can connect to the OPC Server and use it to read and write device data. An OPC Server is analogous to the role a printer driver plays to enable a computer to communicate with an ink jet printer. An OPC service is based on client-server architecture.

All in all, we just need to know that if an OPC server is compromised, it means that an attacker has full control (or almost) of our off shore network.

## 3.6 Gateway


The gateway is the entry-exit point of our network. Traffic that enters or exits the network has to go through our gateway. If the gateway crashes, it means that the internal network and the Internet is separated from each other. Of course, if we choose to communicate only between hosts inside the network, we probably can choose to ignore the gateway. The gateway's another important task is to serve as a firewall filtering and as a NAT.

Therefore Gateway's three tasks:

- **Internet entry-exit point**
- **Firewall Packet filtering**
- **Firewall NAT**

### 3.6.1 Firewall packet filtering

In the *filter table* of iptables, we used ALLOW ALL. Any traffic can come in and out of the network. The only exception is that we have configured our gateway to deny access to the private IP address of 10.0.0.X. The 10.0.0.X is the private network between the gateway and honeywall; we want this subnet to be invisible logically so that no one can access this subnet.

Usually, users use ICMP (Internet Control Message Protocol) **[31]** to ping a host to determine if they are alive. We have configured the firewall to reply with “icmp port unreacheable” when anyone tries to ping the 10.0.0.X subnets. This will let others think that the subnet doesn't exist. The only way to access this subnet is via HTTPS from the Internet. In this case, the outside host will not know that the subnet exist because NAT has already translated the addressing.

### 3.7.1 Firewall NAT

The *NAT table* in iptables, allows us to forward traffic for certain ports to another destination. In our configuration, the forwarded ports are 443 (HTTPS) **[26]**, 3389 (RDP) **[32]**, 21379 (OPC Tunneller) and 135 (RPC) **[33]** . We are now directing Internet traffic for these ports to a specific host. For example, if I try to use HTTPS services from a remote location to access this off shore network.

The HTTPS service is meant to access the honeywall. We do not know the IP of the honeywall since it is a private IP. We can only try to connect to the IP address of the off shore platform. When the firewall receives traffic meant for the HTTPS port (port 443), it then forward the traffic to the honeywall.

Therefore, when someone uses HTTPS he/she doesn't really know which host inside the network is serving him/her. This is all hidden to the user and NAT is the technique used to allow all these to work. Details about NAT configuration is described in **Appendix A**.

## 3.7 Honeywall


Honeywall is placed between the gateway and all other hosts in the network. It monitors traffic in a stealthy way.

The primary two tasks of honeywall are:

- **Traffic monitoring**
- **Intrusion Prevention**

### 3.7.1 Traffic monitoring

The honeywall should be non-existent from a logical view. In a perfect case, no one should know there is a honeywall host unless that person enters the laboratory and sees our network with his or her own eyes.

Honeywall doesn't have IP on all but one interface. The one interface that has an IP uses the gateway to block anyone trying to access it. The interface with IP is used as a remote management interface and while the other two interfaces without IP are the bridge. Traffic passes through the bridge, just like cars pass through a real bridge. Since the bridge interfaces don't have IP, it acts like a physical medium (say a hub) and when there is traffic, it just passes the traffic along.

Of course, it doesn't just pass the traffic. It reads, copies and analyzes them.

### 3.7.2 Intrusion prevention

Instead of preventing intrusion targeted at us, we are trying to prevent intrusion originating from our network. A special snort function call snort_inline is deployed. This function act as an intrusion prevention function that prevents intrusion that originates from our network.

We want to deploy this prevention mechanism because our network is filled with honeypots. Honeypots are meant to lure attackers to attack the network so that we can learn from these attacks. However, we do not want an attacker that has successfully attacked our network to launch attack from our network against others.

Our goal is not to drop all outbound traffic, only attacks. We want to use a ruleset that only has actual attacks or exploits. We do not want to block outbound informational queries, such as ICMP ping, finger query, or a simple HTTP GET command. If we all outbound traffic, then the attacker will once again not be able to do anything outbound.

There are several ways to limit outbound traffic. The first is to drop all outbound malicious packets, second is to limit outbound traffic to a specific volume. Limiting outbound traffic is especially important towards prevent denial of service attacks from our network towards others.

We do not set the outbound traffic to zero, as this will arouse the hacker's suspicion as to why there is no outbound connection allowed. However, we also don't want to have limitless outbound traffic.

In iptables, we can set the number of times an attacker can initiate a TCP, UDP and ICMP outbound connection. The number of connections we allow depend on the level of risk we are willing to take. Limiting the number of outbound connections prevents attackers from using the honeynet to scan or attack large numbers of other systems, or to launch denial of service attacks.

Connection-limiting can sometimes be a tell-tail for an attacker that the network he is attacking is actually a honeypot. An attacker may realize that they are blocked after a certain number of packets. The default for connection limits in Honeywall CDROM is as follows **[34]**.

# 4. Configuring the Network

In this section, we will highlight some important instructions related to the setup process.

## 4.1 Network Interface Configuration

Our PCS network has three subnets. They have private IP range of 192.168.0.X, 192.168.1.X and 192.168.2.X. We also have one other subnet with 10.0.0.X range for walleye's remote access.

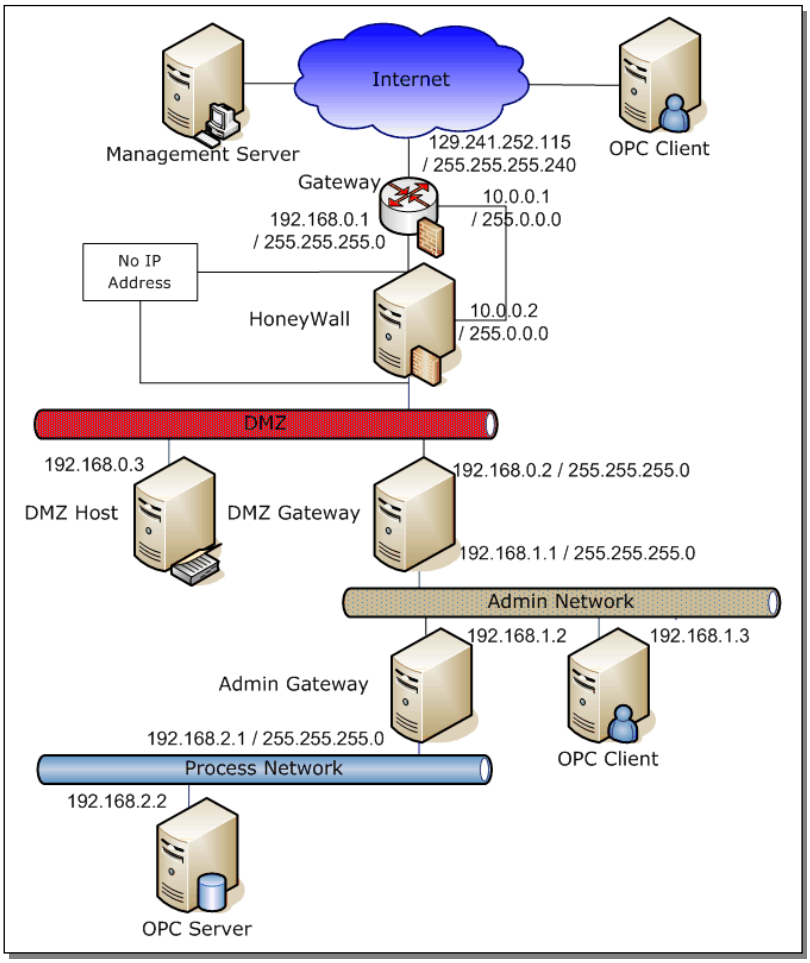



**Figure 6 : Network Topology with IP address**

Following table summarizes the IP address, subnet for each network interface of individual node in the network.

| Host Name | Network Interface | IP Address | Subnet Mask |
|---|---|---|---|
| Internet-dmz | eth0 | No IP address | No Subnet Mask |
| | eth1 | No IP address | No Subnet Mask |
| | eth2 | 10.0.0.2 | 255.0.0.0 |
| router | eth0 | 129.241.252.115 | 255.255.255.240 |
| | eth1 | 10.0.0.1 | 255.0.0.0 |
| | eth2 | 192.168.0.1 | 255.255.255.0 |
| dmz-admin | eth0 | 192.168.0.2 | 255.255.255.0 |
| | eth1 | 192.168.1.1 | 255.255.255.0 |
| dmz-host1 | eth0 | 192.168.0.3 | 255.255.255.0 |
| admin-process | eth0 | 192.168.1.2 | 255.255.255.0 |
| | eth1 | 192.168.2.1 | 255.255.255.0 |
| process-host1 | Local Area Connection 2 | 192.168.2.2 | 255.255.255.0 |

**Table 3 : Network Interface Configuration Summary**

To configure the IP address of each node of the network initially we used ***ifconfig*** command **[35]** in the following way. In this example IP address and net mask of interface eth0 is specified.

```
ifconfig eth0 192.168.0.1 netmask 255.255.255.0
```

In that case, whenever computer is started we have to use that command again to configure the network details for each network interface. This problem can be easily solved by storing IP address and other network information in a static way using configuration files associated with each network interface card. This is known as static IP address configuration which is commonly used in PCS network. For example, we can specify the IP address, net mask and other configuration of an interface eth0 in the file **/etc/sysconfig/network-scripts/ifcfg-eth0**. This path may vary in different Linux distribution. Here is one sample configuration file.

```
DEVICE=eth0
BOOTPROTO=static
HWADDR=00:04:76:25:99:24
BROADCAST=192.168.1.255
IPADDR=192.168.1.2
NETMASK=255.255.255.0
NETWORK=192.168.1.0
ONBOOT=yes
TYPE=Ethernet
```

Another advantage of this approach is configuring the same topology for further investigation is very quick and easy. Because configuration files are replaced and interfaces are physically added or removed according to the topology. All the network configuration files are given in **Appendix A.**

## 4.2 Routing Configuration

In this section we will describe the routing table configured in each node of the network. Main objective of the routing table was to communicate with each node placed in different network. We also configured routing tables in such way that OPC server is accessible from Internet. OPC Client from Internet can connect with OPC Server in our test network. Now routing table of different nodes are illustrated below.

**Router:** Router connects the external network (e.g. Internet) with the internal lab network. Router can route traffic to internal three sub-networks (192.168.0.0, 192.168.1.0 and 192.168.2.0). It can also forward the traffic targeted for management interface of Honeywall which is in 10.0.0.0 network. Finally, for all other traffic default gateway is 129.241.252.113; which is used to connect with the Internet. Full routing table of the router is given below.

```
[root@router ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
129.241.252.112 0.0.0.0         255.255.255.240 U     0      0        0 eth0
192.168.2.0     192.168.0.2     255.255.255.0   UG    0      0        0 eth2
192.168.1.0     192.168.0.2     255.255.255.0   UG    0      0        0 eth2
192.168.0.0     0.0.0.0         255.255.255.0   U     0      0        0 eth2
10.0.0.0        0.0.0.0         255.0.0.0       U     0      0        0 eth1
0.0.0.0         129.241.252.113 0.0.0.0         UG    0      0        0 eth0
```

**Honeywall:** Honeywall bridges two interfaces to transparently monitor traffic between external and internal network. In our configuration we used eth0 and eth1 for this purpose. The status of bridge is shown below.

```
[root@Internet-dmz roo]# /usr/sbin/brctl show
bridge name     bridge id               STP enabled     interfaces
br0             8000.0008742b0a28       no              eth0
                                                        eth1
```

Apart from bridging, management interface of Honeywall can be accessed via https. To access the management interface Honeywall is connected with router using totally different network 10.0.0.0 and default gateway for Honeywall is router (10.0.0.1). Complete Routing table of the Honeywall is given below.

```
[roo@Internet-dmz ~]$ /sbin/route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        0.0.0.0         255.0.0.0       U     0      0        0 eth2
0.0.0.0         10.0.0.1        0.0.0.0         UG    0      0        0 eth2
```

**DMZ Gateway:** This gateway also forwards the traffic from three sub-networks to appropriate destination. The default gateway for this node is the router (192.168.0.1) so that it can route traffic from and to the Internet. Routing table of the DMZ gateway is shown below.

```
[root@dmz-admin ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.2.0     192.168.1.2     255.255.255.0   UG    0      0        0 eth1
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 eth1
192.168.0.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
0.0.0.0         192.168.0.1     0.0.0.0         UG    0      0        0 eth0
```

**Host in DMZ Network:** Host placed in DMZ network can be used to configure DHCP or DNS for the PCS network. It can communicate with other nodes in the test network and Internet via default gateway which is router (192.168.0.1).

```
[root@dmz-host1 ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.0.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
0.0.0.0         192.168.0.1     0.0.0.0         UG    0      0        0 eth0
```

**Admin Gateway:** This gateway forwards the traffic for process and admin network to proper destination. Its default gateway is the DMZ gateway (192.168.1.1) which is used to connect with the Internet.

```
[root@admin-process ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.2.0     0.0.0.0         255.255.255.0   U     0      0        0 eth1
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
0.0.0.0         192.168.1.1     0.0.0.0         UG    0      0        0 eth0
```

**Host in Process Network:** OPC Server is placed as a host in process network. Routing table extracted from the Windows XP machine is shown below. Although it looks complicated, but default gateway is pointed to Admin gateway (192.168.2.1). That gateway can forward traffic from or to the Internet. Usually it's the OPC client which will connect from the Internet to monitor PCS network remotely.

```
C:\Documents and Settings\Student>route print
===========================================================================
Interface List
0x1 ........................... MS TCP Loopback interface
0x2 ...00 0a 5e 19 f6 17 ...... Generic Marvell Yukon Chipset based Ethernet
Controller - Packet Scheduler Miniport
===========================================================================
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0      192.168.2.1     192.168.2.2       10
        127.0.0.0        255.0.0.0        127.0.0.1       127.0.0.1       1
      192.168.2.0    255.255.255.0      192.168.2.2     192.168.2.2       10
      192.168.2.2  255.255.255.255        127.0.0.1       127.0.0.1       10
    192.168.2.255  255.255.255.255      192.168.2.2     192.168.2.2       10
        224.0.0.0        240.0.0.0      192.168.2.2     192.168.2.2       10
  255.255.255.255  255.255.255.255      192.168.2.2     192.168.2.2       1
Default Gateway:       192.168.2.1
===========================================================================
Persistent Routes:
  None
```

**Routing in Linux:** In case of Linux, we can use ***route*** command to add, remove or modify route. Again the problem is we have to use the command again and again each time computer is restarted. Here are examples of adding route and default gateway in the routing table using command.

```
route add -net 192.168.2.0 netmask 255.255.255.0 gw 192.168.1.2
route add default gw 192.168.0.1
```

The same task can be done using configuration files to store static routes and default gateway. In this way we can reuse the configuration files later and we don't have to configure routing each time computer is restarted. One example of configuring default gateway is to add following lines in specific interface related file, for example in **/etc/sysconfig/network-scripts/ifcfg-eth0**

```
GATEWAY=192.168.1.1
```

The above line can be added in **/etc/sysconfig/network** file instead. We added static routes in router using the file **/etc/sysconfig/network-scripts/route-eth2** in following way. Files used to configure the routing are given in **Appendix A**.

```
GATEWAY0=192.168.0.2
NETMASK0=255.255.255.0
ADDRESS0=192.168.1.0
```

Packet forwarding is enabled in router, DMZ gateway and admin gateway. To enable packet forwarding we changed the following parameter in **/etc/sysctl.conf** file according to following way.

```
net.ipv4.ip_forward = 1
```

**Routing in Windows XP:** In case of Windows XP, we configured default gateway while configuring the IP address of the machine using graphical interface mentioned in **Appendix A**. We can add static route using route command. For example, a static route to the 10.0.0.0 network that uses a subnet mask of 255.0.0.0, a gateway of 10.0.0.1, we use following command:

```
route add 10.0.0.0 mask 255.0.0.0 10.0.0.1
```

## 4.3 Honeywall Installation

In earlier sections, we have introduced honeywall and its purpose in the network. We now look at the installation steps. For a more detailed installation guide, you may also go to the honeynet project website **Error! Reference source not found.**.

We will use the Honeynet Roo CD for installation.

- First, we make sure our host meets the minimal requirements for Roo CD. **[36]**
- We then download the Roo ISO here[http://www.honeynet.org/tools/cdrom/] and create a CD.
- The installation will be fully automated once the CD is booted. The honeywall project Splash screen **(figure 7)** is loaded is and continued with the installation process. **(figure 8)**

The default configuration file **honeywall.conf** used for the installation can be referred in **Appendix A**.


**Figure 7 : Honeywall splash screen**

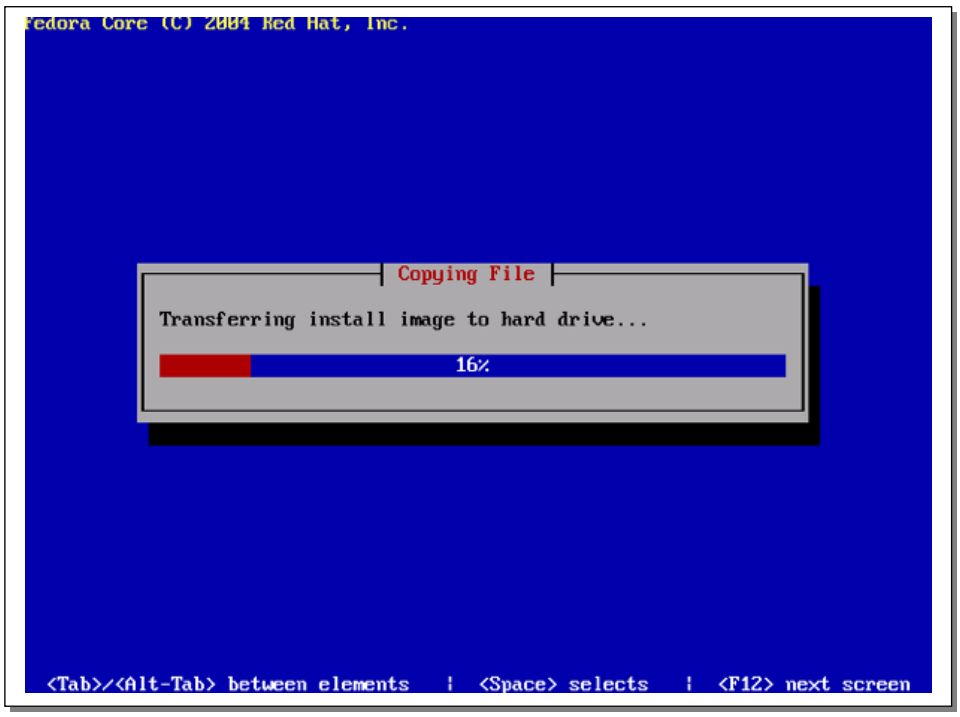



**Figure 8 : Honeywall installation screen**

## 4.4 Honeywall Physical Connections

According to the **figure 9**, the honeywall has three interfaces. Two of the three interfaces are meant to be setup as bridge-connecting the router and the DMZ network. These two interfaces will not have IPs and can be setup to be a bridge via the honeywall menu. The third interface connecting to the router is optional. It is needed and given an IP if we want to have walleye remote management of our honeywall.

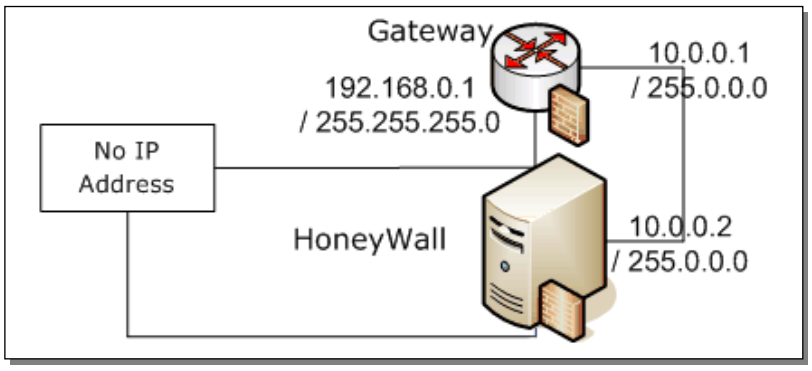



**Figure 9 : Honeywall with three interfaces**

## 4.5 Honeywall Walleye Explained

Remote access can only be accessed after allowing walleye in honeywall. Also, the gateway would also need to allow https traffic to be routed to the local honeywall IP for walleye to work. The login credentials are given below.

**Login Page :** https://129.241.252.115/
**Login :** roo
**Password :** bL!1team

Walleye interface looks like **figure 10**.

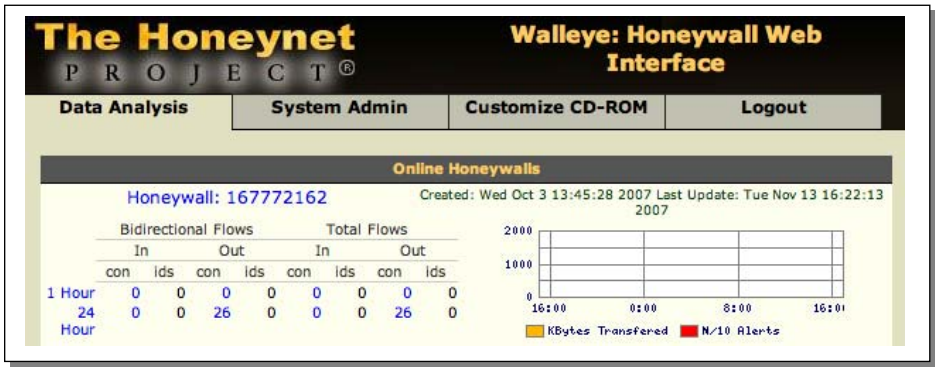



**Figure 10 : Honeywall management interface - Walleye**

## 4.6 Honeywall Walleye Connection-limiting

In honeywall, we can set four types of protocol for connection-limiting:TCP (Transmission Control Protocol), UDP (User Datagram Protocol), ICMP and All Protocols. Connection-limiting is meant to prevent connections out of the honeypot networks. This will prevent attacks being stage from our network if the honeypots is taken over by attackers. The connection-limit can be refreshed by the minute, hourly, daily and weekly.

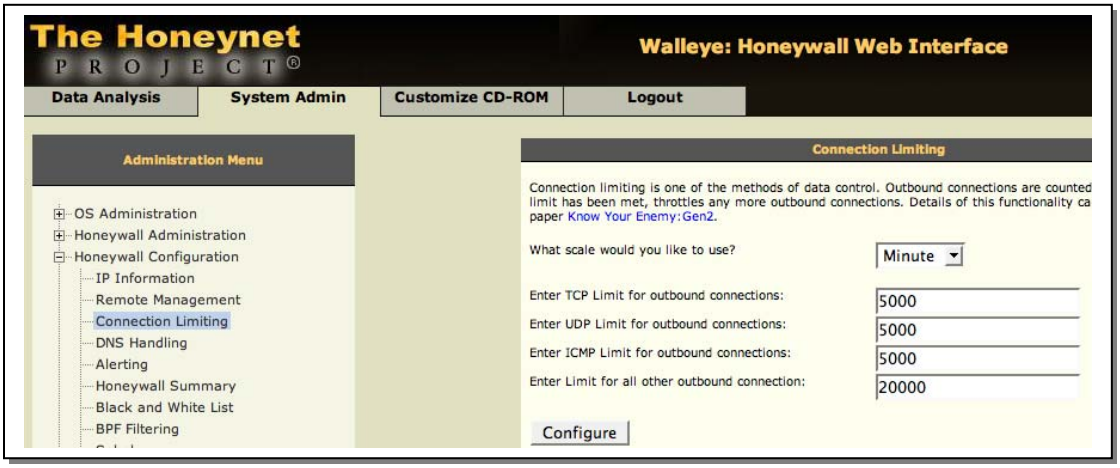



**Figure 11 : Walleye – Connection Limiting**

## 4.7 Honeywall firewall configuration

We will run the honeywall's iptables and ALLOW all packets inbound and outbound. We do not want honeywall's iptables to serve as a firewall, but rather as an IPS. Therefore, iptables will allow all traffic in and out of the network. The only traffic that it drops are malicious traffic coming from our network. *Snort_inline* is the function that instructs which packets should be dropped. We will leave the firewalling to the gateway instead.

The honeywall firewall is started automatically. We can auto start it via startup script ***/etc/init.d/rc.firewall*.** Honeywall's iptables does not work with "*sbin/service iptables*" command. It will not work and will also show the iptables' status to be *alive* even when iptables is not started. (Giving the false impression that iptables is running)

After starting the iptables, we look at the log in ***/var/log/iptable/***. We attempt to ping from the gateway (192.168.0.1) to a honeypot (192.168.2.1)

We read

```
...
Nov 14 14:19:29 roo-test kernel : INBOUND ICMP IN=br0 OUT=br0 PHYSIN=eth0
PHYSOUT=eth1 SRC=192.168.0.1 DST=192.168.2.1 LEN=84 TOS=0x00 PREC=0X00 TTL=64
ID=0 DF PROTO=ICMP TYPE=8 CODE=0 ID=169-9 SEQ=1
...
```

We now confirm that iptables in honeywall is running and logging traffic.

## 4.8 Honeywall Configuration issues

The honeypot/honeywall implementation which we are using is from the honeynet project. In the process of configuration, we encountered some bugs with the honeywall configuration.

### 4.8.1 Bridge interface not working in *Menu*

We configured Honeywall using the Menu (**usr/sbin/menu**). However, we are not able to configure internal and external interface using Menu's normal configuration where we are suppose to type in the interface addresses. Instead, we needed to choose 'reconfigure system', which reconfigures the whole honeywall configuration to setup the interface addresses.

### 4.8.2 Old sensors remained in Walleye

In Walleye, traffic statistics called sensors are displayed. Whenever, we change the IP addresses to be monitored, it generates a new sensor without deleting the old one.

This issue is also documented in Honeynet project's homepage as stated below: **[19]**

*"...Sensor identification is based on the management IP address of your Honeywall. If you change the IP address of your Honeywall, you will have multiple sensors listed (there is no way to delete old ones). This is a known issues...."*

### 4.8.3 Honeywall firewall restart for every Menu exit

The firewall restarts every time we exit Menu (**usr/sbin/menu**). It will be better if we can choose to restart or not everytime we exit Menu.

### 4.8.4 Honeywall firewall Quick

Honeywall uses Fedora Core 6 as its base. We can start/stop a firewall using the command

```
-etc-init.d-rc-firewall start/stop/status.
```

or

```
sbin/service iptables start/stop/status
```

Both commands are suppose to render the same effect. However, the latter command is unable to start/stop the firewall. Also, using the latter command, we can display the "status" even when a firewall is not running.

## 4.9 Gateway firewall configuration

Our gateway's firewall also uses **iptables**. There are two tables that we will configure- ***NAT table*** and ***filter table***.

For *filter table*, we will accept all traffic in the input, output and forward chain. The only traffic that we will reject is the destination IP address of 10.0.0.X. We make sure that no traffic entering the gateway will get a reply from address 10.0.0.X, we also make sure that no traffic that is forwarded from the gateway to 10.0.0.X will get any reply. The former is to prevent our gateway's 10.0.0.1 interface from replying to any query. The latter is to prevent gateway from forwarding traffic to 10.0.0.2, which is the interface of honeywall's remote management.

```
./iptables -A INPUT -d 10.0.0.0/255.0.0.0 -i eth2 -j REJECT --reject-with icmp-port-unreachable
./iptables -A FORWARD -d 10.0.0.0/255.0.0.0 -i eth2 -j REJECT --reject-with icmp-port-unreachable
```

We now move on to the NAT table. NAT will help us ensure that traffic from private IP inside our network can reach the Internet and vice versa. The NAT table consists of prerouting, postrouting and output chain. We accept all as default.

Since we have four private subnets – 192.168.0.X, 192.168.1.X, 192.168.2.X and 10.0.0.X. We will need to configure them to change ip addresses from these private subnets ips to our public ip (129.241.252.115). Traffic originating from these networks to the Internet will their way to the Internet with NAT's help. Traffic from the Internet will also be able to reach the hosts with private ips. Since we are changing the ips after the traffic has traverse the routes (routed) in the network, these rules should be written in the postrouting chain.

```
./iptables -t nat -A POSTROUTING -s 10.0.0.0/255.0.0.0 -o eth0 -j SNAT --to-source 129.241.252.115
./iptables -t nat -A POSTROUTING -s 192.168.0.0/255.255.255.0 -o eth0 –j SNAT --to-source 129.241.252.115
./iptables -t nat -A POSTROUTING -s 192.168.1.0/255.255.255.0 -o eth0 –j SNAT --to-source 129.241.252.115
./iptables -t nat -A POSTROUTING -s 192.168.2.0/255.255.255.0 -o eth0 –j SNAT --to-source 129.241.252.115

./iptables -t nat -A PREROUTING -i eth0 -p tcp -m tcp --dport 135 -j DNAT --to-destination 192.168.2.2
./iptables -t nat -A PREROUTING -i eth0 -p tcp -m tcp --dport 135 -j DNAT --to-destination 192.168.2.2
./iptables -t nat -A PREROUTING -i eth0 -p tcp -m tcp --dport 135 -j DNAT --to-destination 192.168.2.2
```

Last but not least, we setup the rule to route traffic from port 443 to the host with private ip 10.0.0.2. This will allow HTTPS traffic to be routed from the gateway to the honeywall, thus allowing HTTPS remote management.

Since the traffics are entering from the Internet into our network, we set this rule in the pre-routing chain (traffic that comes from the network reaching the gateway should be set instead in the post-routing chain).

```
./iptables -t nat A PREROUTING -i eth0 -p tcp -m tcp --dport 443 -j DNAT --to-destination 10.0.0.2
```

We used "***./sbin/iptables-save***" to save all the commands that you have just typed. You may also use "***./sbin/iptables-restore***" to load commands in a file into iptables. A complete reference of the command and how the iptables status should look like when we issue "***./sbin/service iptables status***", please refer to the Appendix A.

## 4.10 OPC Server and Client Configuration

We used different OPC tools from Matrikon **[17]**. Mainly, we used OPC Tunneller **[37]** and OPC Simulation Server **[29]** from Matrikon to run OPC server in our process network. This is placed in the host in process network. In general, we have to follow these steps to run OPC simulation server from Matrikon successfully.

- Install and run server-side component of MatrikonOPC Tunneller
- Install and run MatrikonOPC simulation server

OPC Explorer **[38]** from Matrikon is used as OPC client in our project. To connect with OPC Server it is necessary to install and configure OPC Tunneller. This is because OPC tools from Matrikon rely on OPC Tunnelling instead of COM/DCOM to communicate with OPC Server. These are the steps to connect with one or more OPC Servers from Matrikon OPC Explorer.

- Install and run client-side component of MatrikonOPC Tunneller
- Install and run MatrikonOPC Explorer
- Connect with OPC Server using OPC Tunneller
- Connect with one or more OPC Server shown in MatrikonOPC Explorer

By default, Matrikon OPC Tunneller does not use encryption or authentication to connect with another OPC Tunneller. But we can use shared secret to encrypt and authenticate communication between client and server side OPC Tunnellers. This can be done in the last step of configuration. All these tools mentioned above are downloaded from Matrikon Website **[39]**. Details about the steps to run OPC server and client are discussed below.

### 4.10.1 Install and Configure OPC Server

First step to run OPC server is to install and run MatrikonOPC Tunneller. At first we downloaded OPC Tunneller from the Matrikon website mentioned earlier. Then while installing the tunneller it is important to check following steps.


**Figure 12 : Installing OPC Tunneller for Server – Step1**

In the abovementioned step of installation we should select **custom** as installation type. Then in the next step we should select **server-side gateway** only as we are going to run OPC server at this time.

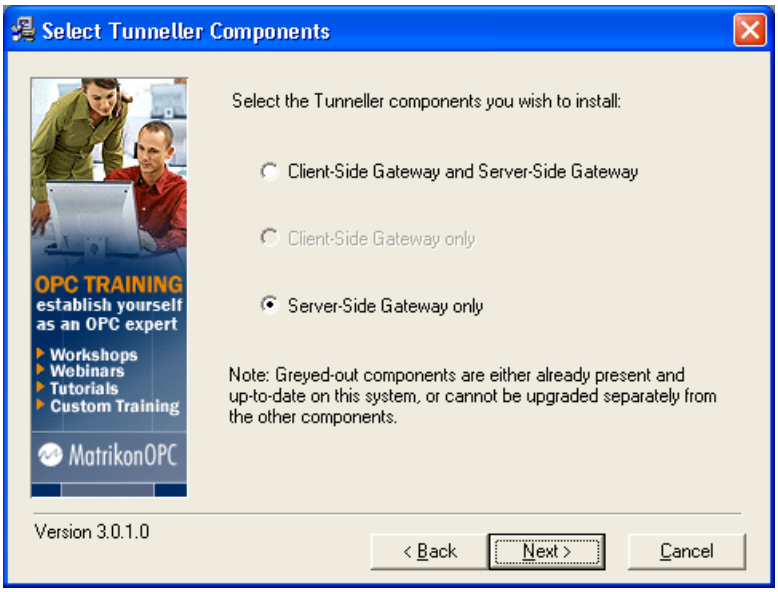



**Figure 13 : Installing OPC Tunneller for Server – Step 2**

After successful installation of OPC Tunneller we can be able to find Tunneller group in the **all programs** section under **start** menu. An example figure is shown below.

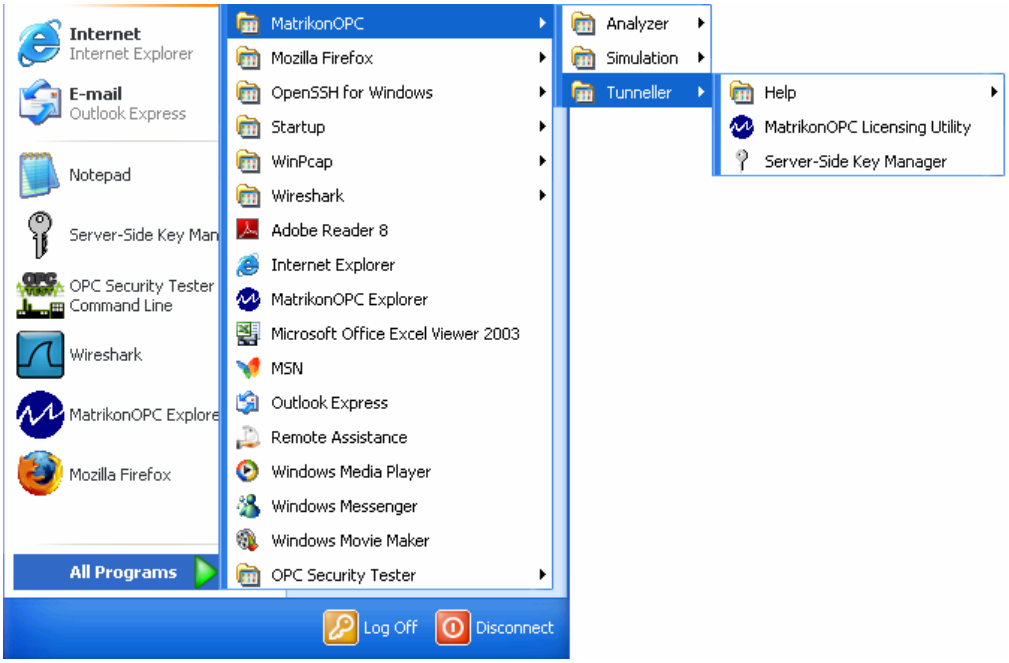



**Figure 14 : After Installing OPC Tunneller for Server**

Next step is to install OPC simulation server from Matrikon. It can be done easily according to the instructions given while installing. There are no special steps which we had to follow while installing OPC Tunneller. By default the name of the OPC server is **Matrikon.OPC.Simulation.1**. After installing OPC simulation server we can find it similarly in **all programs** under **start** menu. An example screenshot is given below.

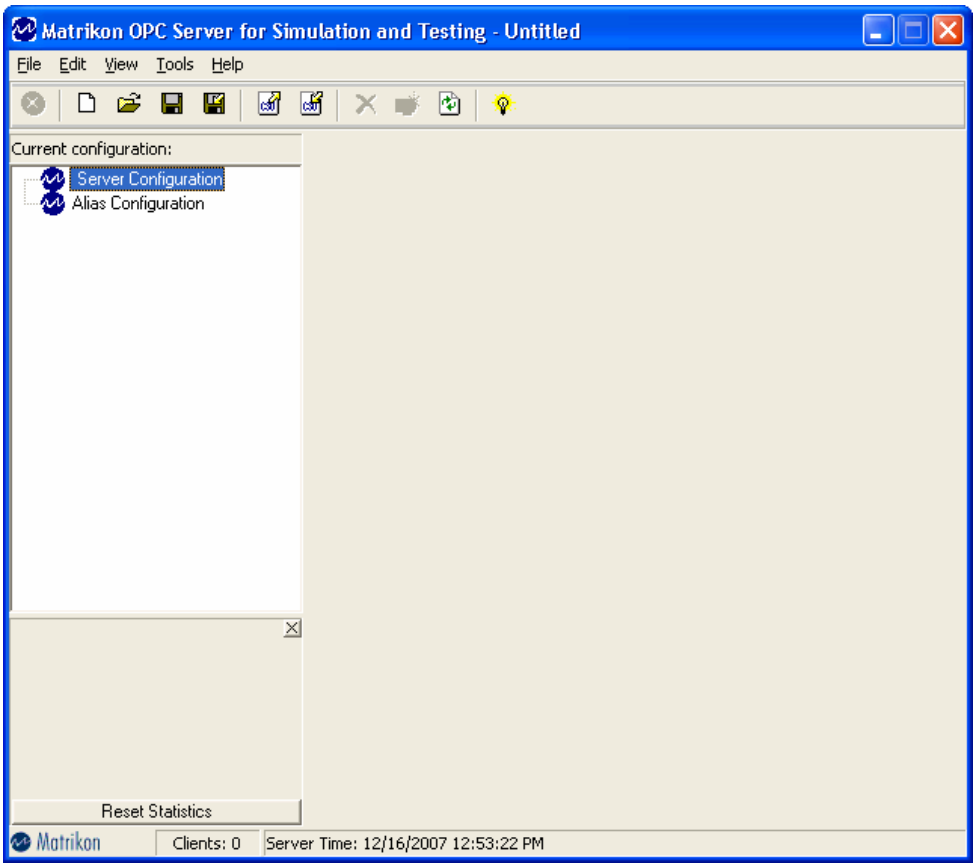



**Figure 15 : After Installing OPC Simulation Server**

This simulation server can be used to test, develop and configure application using simulated data. Testing applications in real or live OPC servers may result loss of production. This server from Matrikon is used to mimic the real environment so that in the event of problem no real process data is lost.

### 4.10.2 Install and Configure OPC Client

MatrikonOPC Explorer is an OPC client designed to help during installation, testing, and configuration of OPC compliant servers. This utility is used to view OPC server items and test OPC network connections. At first we installed client-side component of OPC Tunneller. This is done while installing and it is already mentioned earlier in case of installing server-side component of OPC Tunneller. After successfully installing OPC Tunneller with client-side components we can be able to find **Client-Side Gateway Configuration Utility** under **Start** -> **All Programs** -> **MatrikonOPC** -> **Tunneller** section. A sample screenshot in the situation when there is no connection with OPC servers is shown below.

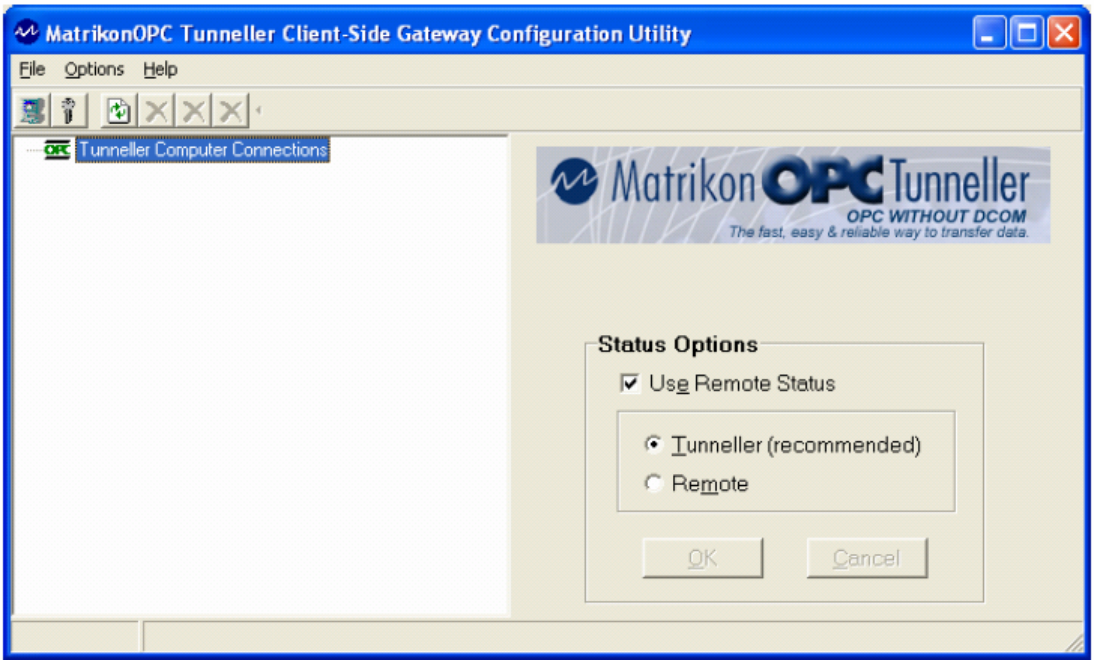



**Figure 16 : After Installing OPC Tunneller for OPC Client**

Next step is to install OPC explorer which is OPC client from Matrikon. Installation is simple and there are no special steps to consider. If anyone follows the installation instructions it will be installed. After installing OPC Explorer we will find it under **Start** -> **All Programs** group with the name **MatrikonOPC Explorer**. An example screenshot of OPC Explorer is shown below.

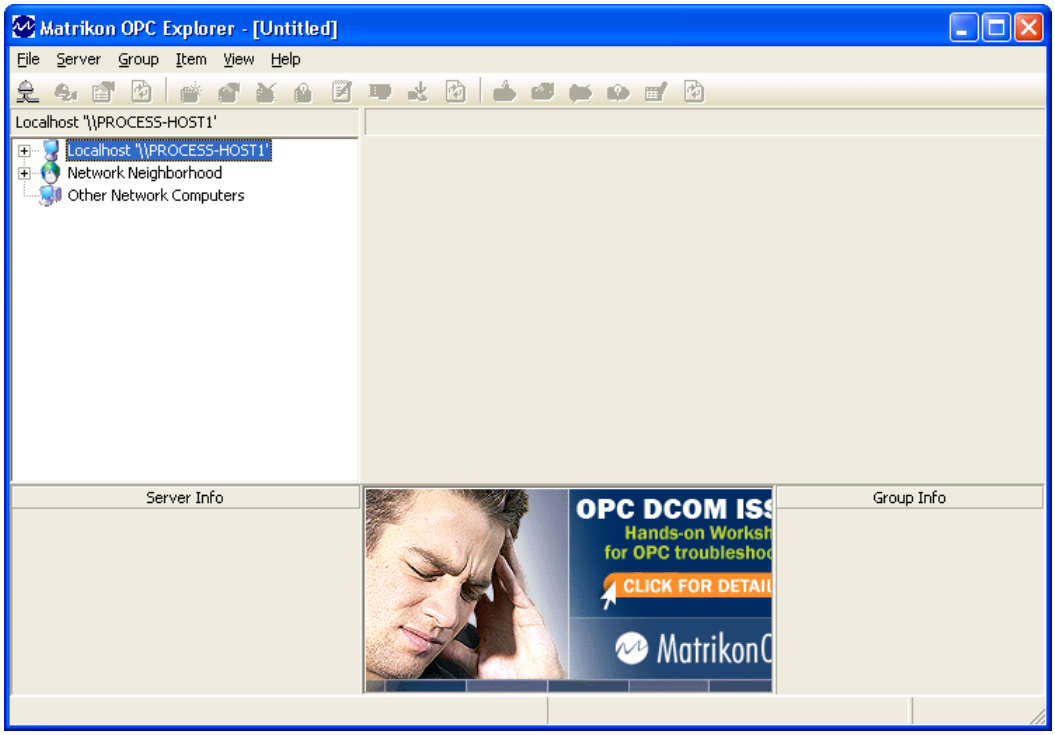



**Figure 17 : After Installing and Running MatrikonOPC Explorer**

So far we have installed all the components necessary to start using OPC server and client. Now we need to connect with OPC Tunneller using **Client-Side Gateway Configuration Utility** which can be found under **Start** -> **All Programs** -> **MatrikonOPC** -> **Tunneller** group. To create a new connection we press Ctrl+N or go to **File** menu and then click on Add **Remote Tunneller Connection** option. Then following screen in **figure 18** will be displayed.

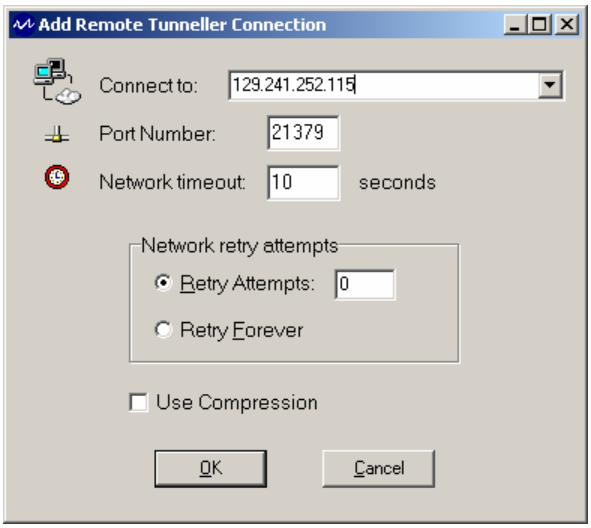



**Figure 18 : Connecting with OPC Server-side Tunneller**

In our case we want to connect with the public IP address of router (129.241.252.115) which will later forward the request to OPC server via NAT configuration. After successful connection we will get list of OPC server available in that machine. One example screenshot is shown below.

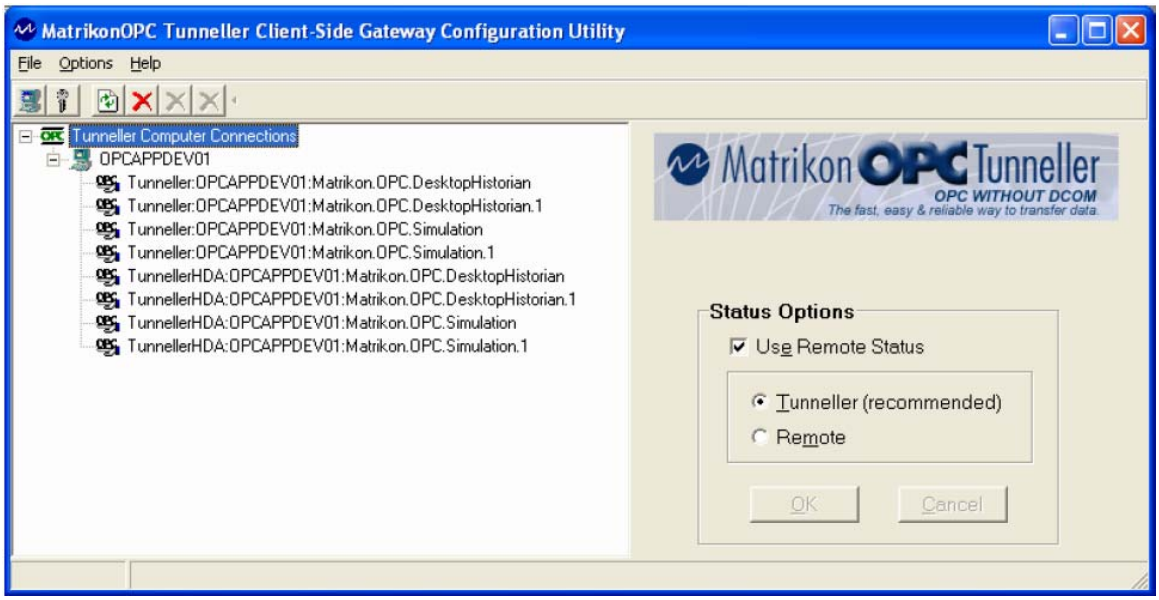



**Figure 19 : After Successfully Connecting with OPC Server-side Tunneller**

Next step is to open OPC Explorer and then it will show list of OPC servers available. Then users may select any OPC server and try to connect with them. One example screenshot of OPC explorer with list of servers is shown below.

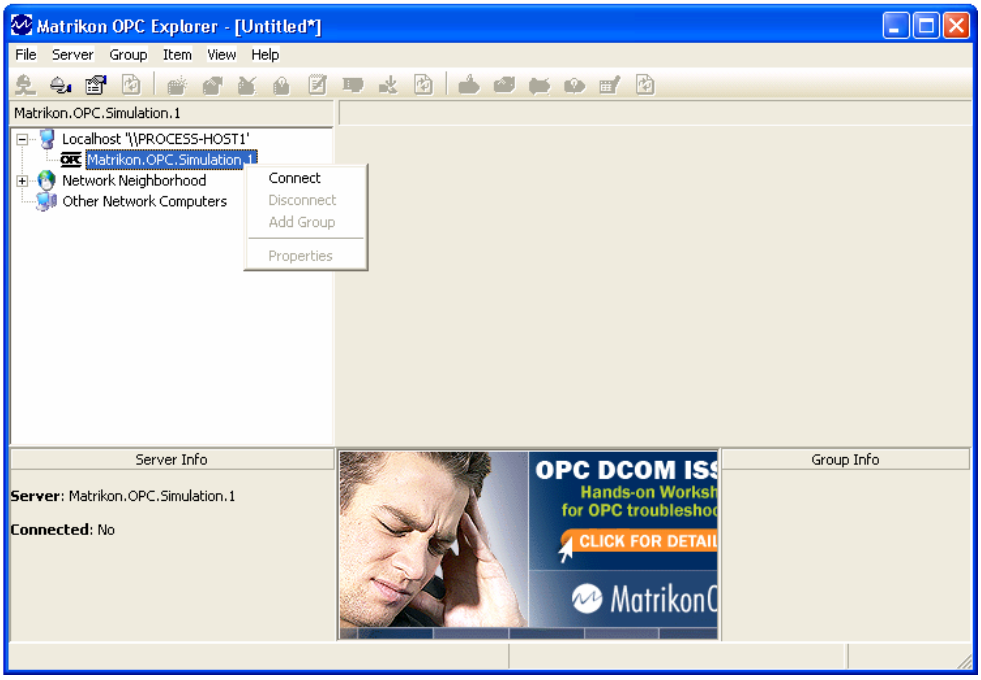



**Figure 20 : Connecting with OPC Server from OPC Explorer**

In this way we installed and configured the OPC server and client to communicate with each other. Now we can add groups and items in OPC simulation server after connecting with the server. To add group in the server right click on the name of server and click on **Add Group** which will show the following screen.

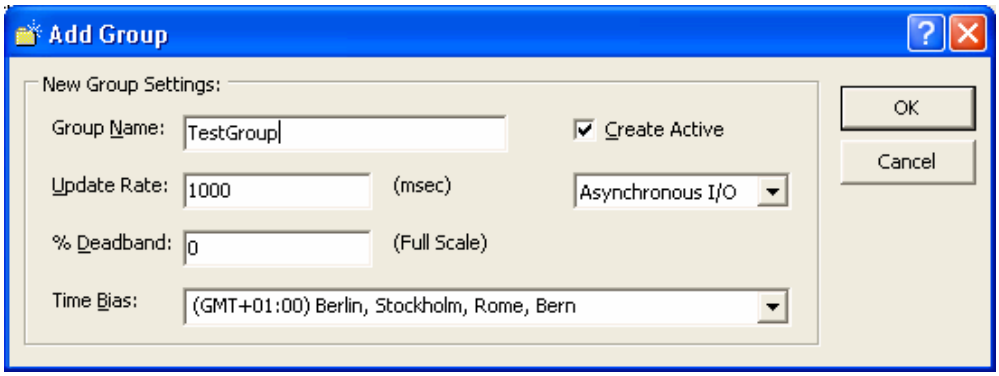



**Figure 21 : Adding Group in OPC Server**

Now in each group we can add items by right clicking on the group and clicking on add items. After clicking on add items it will display a window containing detail different data types, name and refresh rate etc. An example screenshot is shown in the **figure 22** .

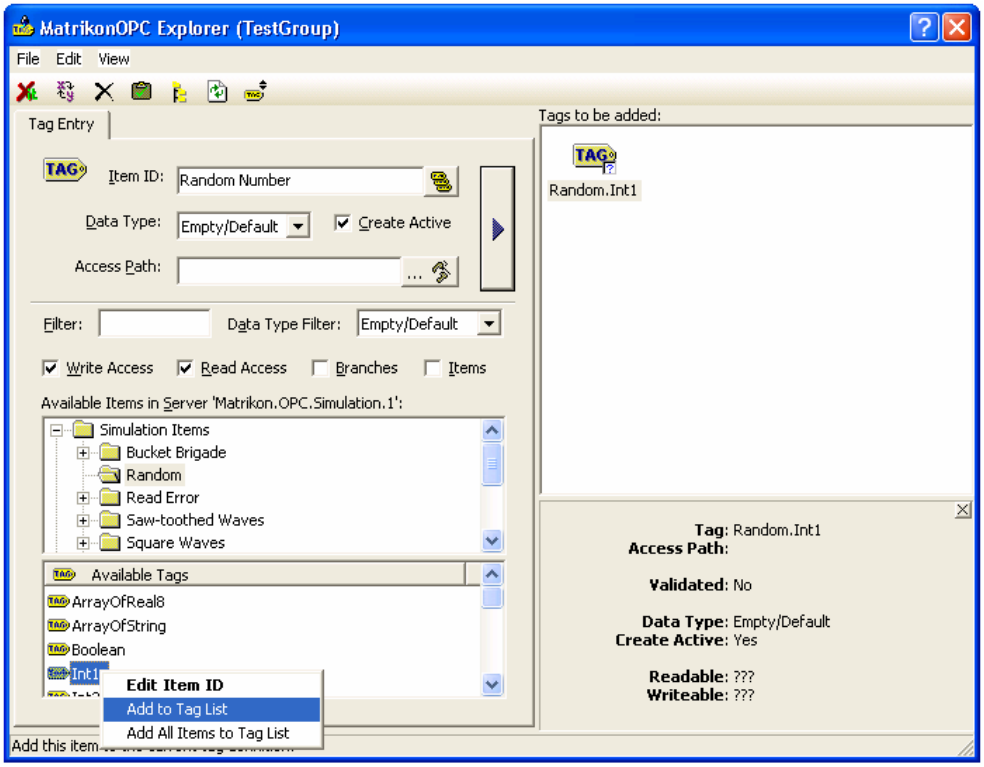



**Figure 22 : Adding Items in the Group of OPC Server**

After successfully adding and validating the new items we can be able to get information about those items in OPC explorer. In our example we have added an item which will show random integer value. Sample screenshot is shown below.

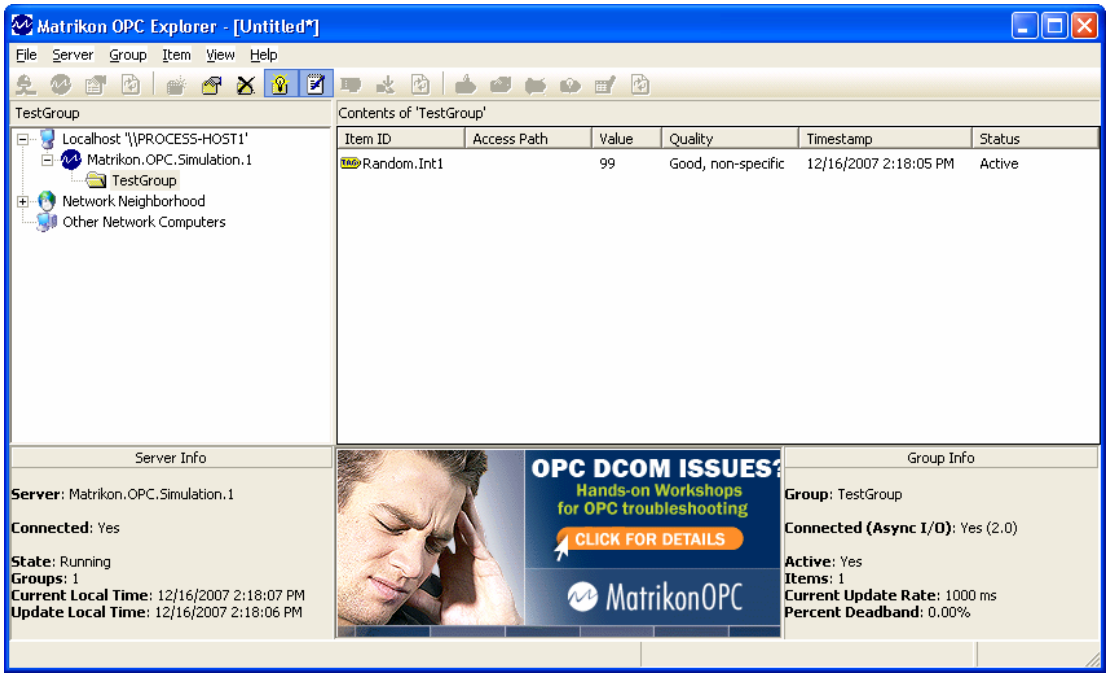



**Figure 23 : Status of Items shown in OPC Explorer**

### 4.10.3 Securing OPC Tunneller

By default there is no security mechanism in OPC tunneling. To configure security mechanism for OPC Tunneller we can use pre-shared secret based on IP address of OPC client or range of IP addresses for network. To configure shared secret in OPC server-side tunneller, we have to add IP address of the OPC clients and shared secret. We can do this by opening **Server-Side Key Manager** which is under **Start** -> **All Programs** -> **MatrikonOPC** -> **Tunneller** group. An example screenshot is shown below.

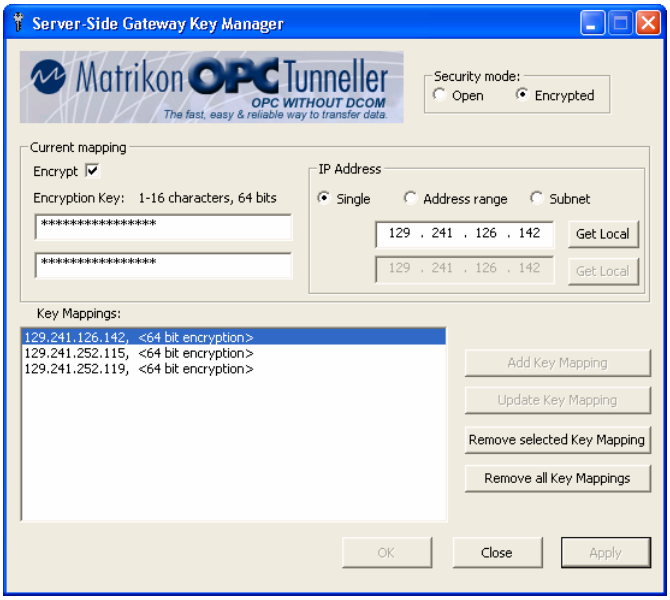



**Figure 24 : Configuring Shared Secret in OPC Server**

In the similar way we add the IP address of the OPC server and shared secret for each of the server we want to connect. Thus both client side and server side now know about the shared secret and they can communicate using encrypted data. Sample screenshot for client side configuration for shared secret is shown below.

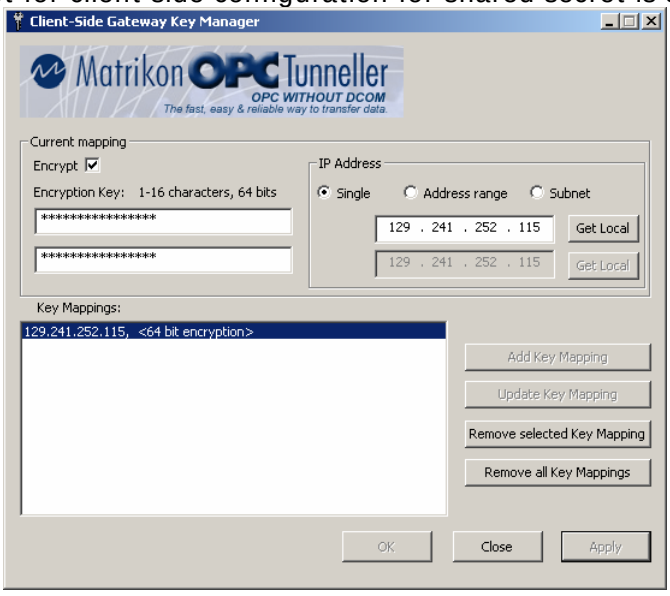



**Figure 25 : Configuring Shared Secret in OPC Client**

Using OPC tunnelling with encryption, we can easily overcome threats and security issues caused by COM/DCOM implementation. OPC tunnelling is not yet standardized by OPC foundation, but it can be better solution for future OPC implementation and standardization.

# 5. Services and Vulnerabilities

In this section we will discuss about the services configured and exposed by the simulated PCS network. Later we summarized the known vulnerabilities specific to each of these services. Finally, exploits and vulnerabilities used by red team are outlined briefly.

## 5.1 Configured Services

Our goal was to configure OPC services in the PCS network. But in the first phase of the project we spent time in configuring the whole network. Later we configured OPC server in the network. But we configured several other services for test purpose. In this way red team could try to find vulnerabilities in those services. For example, after configuring the gateway which connects both external and internal network; we configured telnet, secure shell etc. A screenshot from existing services of the gateway is shown below.


**Figure 26 : List of running services**

Later after configuring the whole network we configured Matrikon OPC server. Then to test and monitor the OPC server from remote location we configured remote desktop service too. Apart from those, we have also configured HTTPS service for Honeywall management interface, FTP service **[40]** , VNC service **[41]** etc.

## 5.2 Service Specific Vulnerabilities

After installing and configuring services, its time to figure out how red team might explore the vulnerabilities in network. The main target would be the services offered by the network and the external gateway. Mainly we found different types of known vulnerabilities in all the configured services. We will discuss those briefly based on popular vulnerability databases like National vulnerability database (NVD) **[42]** , Open Source Vulnerability database (OSVD) **[43]** and the database provided by United States Computer Emergency Readiness Team (US-CERT) **[44]** and articles **[45] [46]**.

### 5.2.1 Telnet Specific Vulnerabilities

In National vulnerability database (NVD), we searched using **telnet** as keyword and found **166** reported vulnerabilities. Using the same keyword in Open Source Vulnerability database (OSVD) we found **124** vulnerabilities. In case of database provided by United States Computer Emergency Readiness Team (US-CERT) showed **47** vulnerabilities. Different types of vulnerabilities are found in different operating system and devices. Some examples are: **[43][44]**

- Microsoft Telnet Server Buffer Overflow
- Microsoft Windows 2000 Telnet Session Timeout DoS
- Microsoft Windows Telnet Service Handle Leak DoS
- Dragon Telnet Server Username Remote Overflow
- Multiple Telnet Clients vulnerable to buffer overflow via the env_opt_add() function in telnet.c
- MIT Kerberos 5 telnet daemon allows login as arbitrary user

There are different types of vulnerabilities related to telnet service. But mainly buffer overflow and Denial of Service attacks can be launched using different flaws of the implementation of the protocol. Details about all these vulnerabilities can be found in the websites mentioned under reference section.

### 5.2.2 SSH Specific Vulnerabilities

Similarly we searched using **ssh** as keyword and found **193** reported vulnerabilities in National vulnerability database (NVD). Using the same keyword in Open Source Vulnerability database (OSVD) we found **141** vulnerabilities. In case of database provided by United States Computer Emergency Readiness Team (US-CERT) showed **64** vulnerabilities. Different types of vulnerabilities are found in applications implementing SSH protocol. Some examples are: **[43][44]**

- SSH host key authentication can be bypassed when DNS is used to resolve localhost
- SSH-1 allows client authentication to be forwarded by a malicious server to another server
- SSH1 may generate weak passphrase when using Secure RPC
- Weak CRC allows RC4 encrypted SSH1 packets to be modified without notice
- Passwords sent via SSH encrypted with RC4 can be easily cracked
- OpenSSH vulnerabilities in challenge response handling
- OpenSSH UseLogin option allows remote execution of commands as root
- SSH sshd Connection Saturation DoS

In case of secure shell, there are vulnerabilities related to the authentication and encryption. Besides there are known vulnerabilities in the OpenSSH and sshd application. Both of them implement SSH protocol. More Details about all these vulnerabilities can be found in the websites.

### 5.2.3 FTP Specific Vulnerabilities

In National vulnerability database (NVD), we searched using **ftp** as keyword and found **928** reported vulnerabilities. Using the same keyword in Open Source Vulnerability database (OSVD) we found **591** vulnerabilities. In case of database provided by United States Computer Emergency Readiness Team (US-CERT) showed **189** vulnerabilities. Some examples are: **[43][44]**

- Microsoft Windows FTP client does not properly validate received file names
- FileCOPA FTP server vulnerable to buffer overflow
- WS_FTP Server vulnerable to buffer overflow via long string sent as argument to ftp command
- Ability Server vulnerable to buffer overflow
- Rhinosoft FTP Voyager File List Overflow
- WS_FTP Pro Directory Listing Overflow
- WS_FTP Pro Directory Listing Overflow

FTP service is mainly vulnerable to buffer overflow due to either long command string or long directory listing. Mainly WS_FTP and some other popular FTP software have these flaws. Details about these vulnerabilities related to FTP can be found in the websites mentioned under reference section.

### 5.2.4 HTTPS Specific Vulnerabilities

In the database provided by United States Computer Emergency Readiness Team (US-CERT), we searched using **ftp** as keyword and found **175** reported vulnerabilities. Using the same keyword in National vulnerability database (NVD) we found **166** vulnerabilities. The Open Source Vulnerability database (OSVD) showed **25** vulnerabilities. Some examples are: **[42] [44]**

- Microsoft IE HTTPS proxy basic authentication URL transmitted in clear text.
- Microsoft Internet Explorer does not properly handle cached HTTPS contents.
- Cisco Content Service Switch reboots when HTTPS POST request is sent to web management interface.
- Multiple buffer overflows in the AirDefense Airsensor M520 which allow remote authenticated users to cause a denial of service via a crafted query string in an HTTPS request.

Most of results we found related to HTTPS were vulnerabilities due to implementation flaw in web browser or crafted query string. But the search result did not correctly show total no of vulnerabilities in HTTPS protocol as most of the vulnerabilities contains link to HTTPS sites. Details about these vulnerabilities related to HTTPS can be found in the websites mentioned under reference section.

### 5.2.5 OPC Specific Vulnerabilities

To find list of vulnerabilities related to OPC we have to also consider DCOM and RPC related vulnerabilities. This is because OPC is implemented using DCOM which uses RPC. Here we searched using **opc**, **dcom** and **rpc** as separate keywords and found related vulnerabilities. From all these databases, we found maximum no of vulnerabilities **(555)** in RPC. Next highest vulnerabilities **(71)** are found in case of DCOM. And total **40** OPC specific vulnerabilities were reported. Here are some examples of vulnerabilities: **[43][44]**

- NETxAutomation NETxEIB OPC Server fails to properly validate OPC server handles
- Microsoft Windows RPC DCOM Interface Overflow
- Microsoft Windows DCOM RPC Object Identity Information Disclosure
- Microsoft Windows DNS RPC buffer overflow
- Microsoft Windows RPC service vulnerable to denial of service
- Microsoft Windows RPCSS Service contains heap overflow in DCOM request filename handling
- Microsoft Windows 2000 RPC Authentication Unspecified Information Disclosure

Here these databases shows vulnerabilities related to buffer overflow, Denial of Service, information disclosure; especially in case of RPC. OPC specific vulnerabilities are described more details in the whitepaper **[45]** from Digital Bond. Main points of their discussion regarding OPC specific vulnerabilities are summarized below.

**Lack of Authentication in OPC server browser:** Configuration guidance from many vendors, recommends allowing remote Anonymous Login so OPCEnum will work when DCOM Authentication is sent to “None”. If a buffer overflow of some type were discovered in the OPC Server Browser code, the result could be arbitrary code execution or denial of service against any computer running the OPC Server Browser. Fortunately any such buffer overflows in the OPC Server Browser code is not discovered so far. **[45]**

**Lack of Integrity in OPC communications:** The default DCOM settings do not provide message integrity for OPC communications. If the underlying network is compromised and the attacker can sniff and insert traffic, it is likely that rogue messages could be injected once the client and server have authenticated during the initial connection establishment. A number of “man in the middle” tools and techniques are available and it is likely that these could be modified or enhanced to conduct attacks against OPC communication. **[45]**

**Lack of Confidentiality in OPC traffic:** Although DCOM supports message encryption, most of OPC vendors did not recommended enabling Packet Privacy for their OPC Server or the OPC Server Browser. Some vendors recommend VPN tunnelling as a means of providing secure remote access. Matrikon used client and server side tunnelling component with encryption for this purpose. **[45]**

In another article **[46]**, Lluis Mora discussed vulnerabilities of OPC server. For example: attacks using invalid server handle, invalid or crafted configuration file, resource starvation etc. They also developed a tool **‘OPC Security Tester’ [47]** to test these vulnerabilities.

## 5.3 Exploits Used by Red Team

Red team tried to use the vulnerabilities of the services, operating system to launch different types of attack. Their main objective was to take complete control of the network if possible. And as blue team, our goal was to protect the network and detect and prevent any kind of attacks if possible. Before discussing the measures we have taken to prepare, detect and respond to the attacks launched by red team, we want to discuss about the list of attacks and exploits used by red team. After studying different logs in our network and discussing with red team we confirmed about following attacks. These are:

- Scanning and Probing the network
- Sniffing and Monitoring the traffic
- Denial of Service Attacks

### Scanning and Probing the Network

Many tools are available to gather information about network, operating system and services offered by network. Network administrators use these tools to manage network services, monitor hosts or service uptime etc. But as these tools are freely available, attackers can also use these tools for their own purpose. Red team used Nmap **[48]** and Nessus **[49]** for scanning the simulated PCS network. Nmap is a free and open source utility for network exploration. Nessus is a popular remote security and vulnerability scanner. Nessus is also used to test list of vulnerabilities for specific remote host. Details about these tools and the scan result are available in the report of red team. This can be thought as first step of any attacker to explore list of services and ports exposed by a network.

### Sniffing Traffic

Network packet analyzer (e.g. Wireshark **[50]**) can be used to capture network packets and analyze them for planning attacks. Red team used these tools to sniff both traffic from or to OPC server and other network traffic. In case of OPC traffic, by default communication between server and client side tunnellers are not encrypted. In that case they could **read the information** about OPC server and groups and items added or updated in OPC server etc. They also used man-in-the-middle attack due to lack of access control mechanism. In case all types of traffic they tested using **ARP spoofing** attack which is a vulnerability that allows an attacker to spoof the MAC Address to sniff, modify, redirect or stop the traffic to the IP address of the target system.

### Denial of Service Attacks

Red team used different kind of Denial of Service attack during the last phase of the project. A denial-of-service (DoS) attack **[51]** is an attempt to make a computer resource unavailable to its intended users. They used **SYN-Flooding** and **ARP spoofing** to launch Denial of Service attack. TCP uses three-way handshake (SYN, SYN-ACK, ACK) to establish a session. In case of **SYN-Flooding**, the attacker sends several packets but does not send the "ACK" back to the server or spoofing the source IP address of SYN message, server sends SYN-ACK to false IP address and never receives ACK. **ARP spoofing** can stop the traffic to the spoofed target system. Red team also used Hydra **[52]** and tried to find login password using telnet and HTTPS. But telnet service crashed and in case of HTTPS it was not successful.

# 6. Incident Handling

This chapter will describe how blue team prepared, analyzed and detected different external attacks launched by red team and other external attackers. At first overviews of existing incident handling methods are discussed. Later the technical measures taken by blue team for different types of incident are illustrated. Finally based on our experience, we have a few recommendations for incident handling in PCS network.

## 6.1 Overview of Existing Methods

Dealing with different typed of computer security incidents is very difficult. There are several ways that incidents can occur and many types of impact they can have on the target system. There are existing methods to handle incidents in efficient manner. Two of them are:

- Incident Response Management (IRMA)
- Incident Response (IR)

### 6.1.1 Incident Response Management (IRMA)

IRMA includes all aspects of proper incident response planning – technical, cultural, and organisational issues **[53]**. IRMA proposed a framework to consider technological, organization and cultural issues while performing incident response. The framework consists of following steps: **[53]**

- Identification of attack or incident
- Analysis of root cause - categorised by direct influencing factor
- Identification of changes in governing values based on risk assessment of end result
- Establishment of actions/barriers based on processes both bottom up and top down

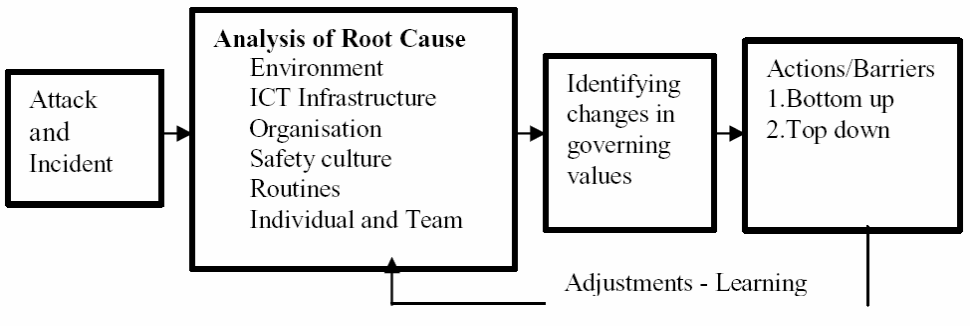



**Figure 27 : Steps in IRMA Framework [53]**

Current methodology of IRMA does not include complete specification of response management with example. Whereas detailed specification of Incident Response (IR) is available. Next section describes about that method.

### 6.1.2 Incident Response (IR)

Incident response process has been defined differently by different organizations. But in this report, we will use the process suggested by NIST (National Institute of Standards and Technology). According to their guide **[23]**, incident response process has four phases. These are:

- Preparation
- Detection and Analysis
- Containment, Eradication and Recovery
- Post-Incident Activity

**Preparation:** During preparation phase incident response team attempts to prepare and minimize the number of incidents or attacks. The team should use necessary tools, software and resources to detect the incident as fast as possible. One important task in this phase is to identify critical components and resources in the system by conducting periodic risk assessment. This assessment should identify risks posed by known threats and vulnerabilities. Later risks should be sorted according to priority. Finally the team should try to mitigate those risks. **[23]**


**Figure 28 : Phases in Incident Response [23]**

**Detection and Analysis:** This phase is composed of several sub phases. At first category of incident can be used to follow guidance based on common or specific category. Next part is to find and detect incidents from different sources. Example sources are: alert from IDS or antivirus, logs from operating system, application or honeypots, publicly available vulnerability website or database etc. After detecting the incident it is necessary to verify the symptoms and find the main reason behind the incident. When the incident is verified, it is documented properly and prioritized. Finally the team needs to notify appropriate persons in the organization to play their roles. **[23]**

**Containment, Eradication and Recovery:** After detecting and analyzing the incident, we should contain it to prevent further damages. Containment should not be delayed which would help attacker to start launching new attacks in the system. Next step is to gather evidence to resolve the incident. System owners might want to identify the attacker in this phase. After containment of an incident, eradication may remove components of the incident, for example: delete malicious code. In some cases eradication may be skipped or performed in recovery phase. In recovery phase, files may be restored from backups; compromised files are replaced by clean files. We should also introduce high level of system logging and network monitoring as a part of recovery process. **[23]**

**Post-Incident Activity:** This is one of the most important parts of incident response. The team and organization should learn and improve to handle new threats efficiently playing their respective roles. A follow-up report for each incident can be used for future case. In this phase, metrics related to incident handling should be collected. Some example matrices are: No of incidents handled, time spent per incident etc. Incident handling checklist should be maintained to speed up the incident handling process. **[23]**

After studying different methods of incident handling, we decided to create a mock organization with different roles. Then based on the mock organization, any of the aforementioned incident handling methods could be tested in our project. Due to lack of time we could not test that incident handling method from organizational perspective. But we have used incident handling method from technical perspective using three phases. These are: Preparation, Detection and Analysis, Response. In the next three sections we will describe these phases.

## 6.2 Preparation Phase

In the first phase, we prepared our network to become ready to detect attacks. But we have found a huge list of vulnerabilities specific to the services which are described in section 4.2. It is not possible to detect all the attacks that exploit those vulnerabilities in short time. So we have decided to mainly focus on the known attacks which can be detected or logged by snort_inline and operating system log files. According to the suggestion from NIST, we have done with vulnerability assessment in external gateway and OPC server. These two nodes are playing most important role in our network. Later we have configured snort_inline and other logging options to detect attacks. Preparation phase can be divided into three parts.

- Vulnerability Assessment
- Configuring **snort_inline** and **walleye**
- Enable logging for system and services

### 6.2.1 Vulnerability Assessment

We have used Nessus version 3 to analyze vulnerabilities of each service offered by our network. After running nessus using the IP address of external gateway and with option **"Enable all plugins with default settings (Even dangerous plugins are enabled)"** the result displayed 5 open ports, 26 notes, **1 warnings** and **1 holes**. But if we use the option **"Enable all but dangerous plugins with default settings (Recommended)"** it displayed 5 open ports, 28 notes, **1 warnings** and **0 holes**. Two main issues discovered after vulnerability assessment. These are:

- **Warning is related to Remote Desktop Protocol:** The remote desktop protocol is used for test purpose to communicate with OPC server from remote location. In real PCS network this is not a usual case. But this protocol is vulnerable to a man-in-the-middle attack. An attacker may exploit the flaw to decrypt communication between client and server to obtain passwords and other information. Nessus suggested **using SSL as transport layer** for this service.

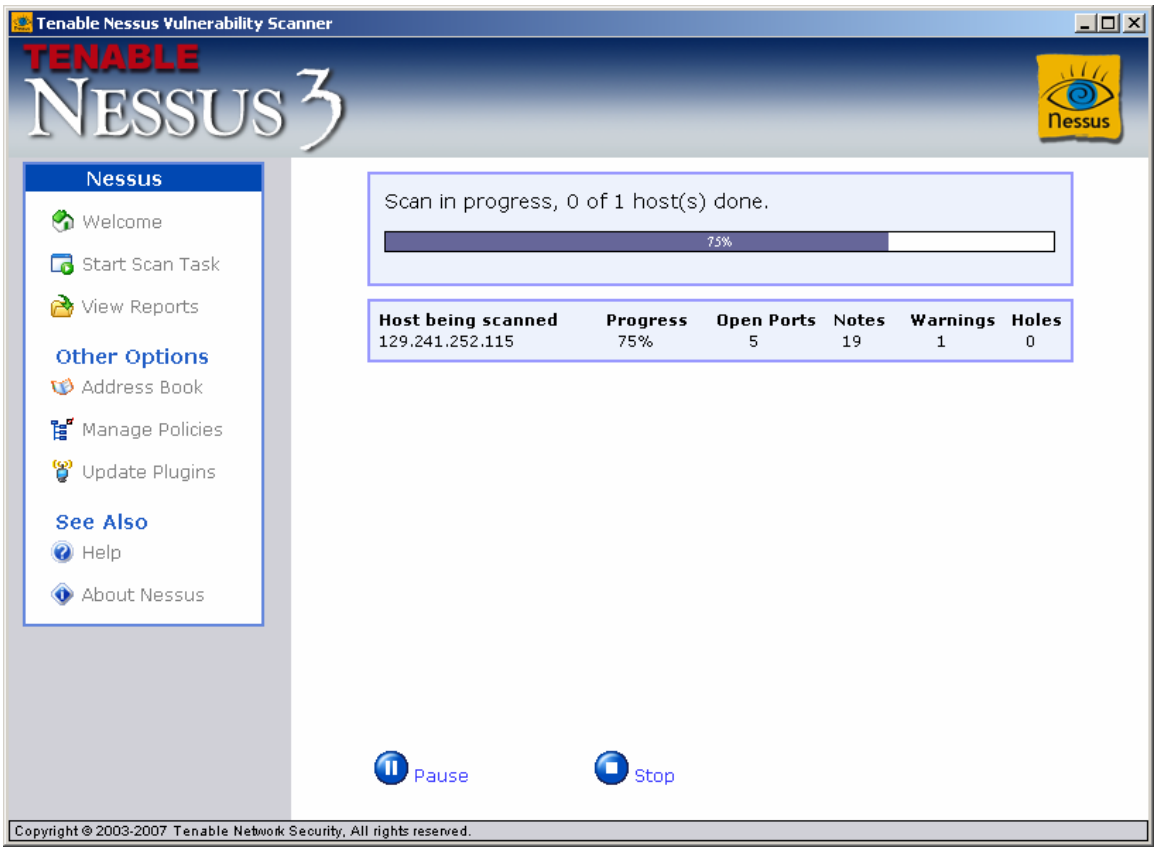



**Figure 29 : Nessus Scanning External Gateway for Vulnerabilities**

- **Hole is due to specially malformed TCP/IP packet:** According to the result from Nessus, it was possible to crash the gateway by sending a specially malformed TCP/IP packet with invalid TCP options. An attacker may use this vulnerability to disable the host remotely. This flaw exists only in the version 2.6 of Linux kernel. Solution proposed by the report is to **upgrade Linux kernel** with version 2.6.7. Screenshot from the result is given below.

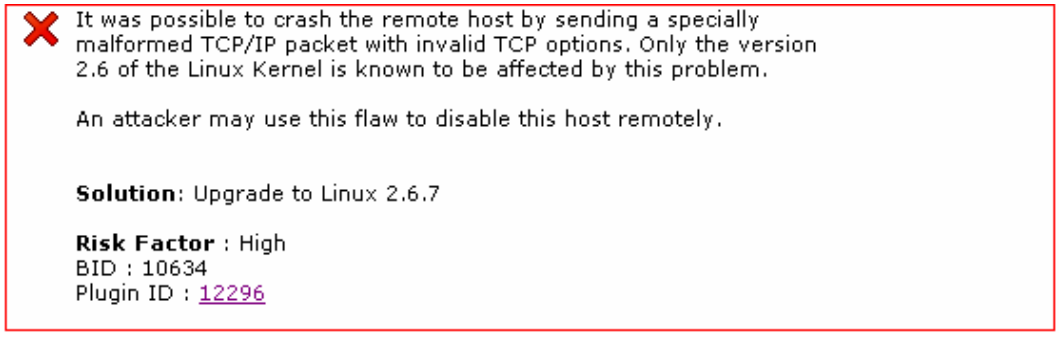



**Figure 30 : Details about hole from nessus report**

We decided to upgrade the kernel and use SSL as transport layer for remote desktop service. Thus vulnerability assessment gave us important feedback about possible threats caused by different services. We also understood the importance of this assessment after testing with different option and plugins offered by Nessus.

### 6.2.2 Configuring snort_inline and walleye

As we have decided to focus on attacks which can be detected and logged by snort_inline, we configure snort_inline properly with logging mechanism for specific attacks. All the logs and alerts can be found in the **/var/log/snort/** location and organized by separate directory for each date. Similarly we also configured Honeywall management interface to monitor incoming and outgoing traffic. These log files from snort_inline and visual interface from walleye will help us to detect external attacks. A sample screenshot from walleye interface with list of connections are shown below.

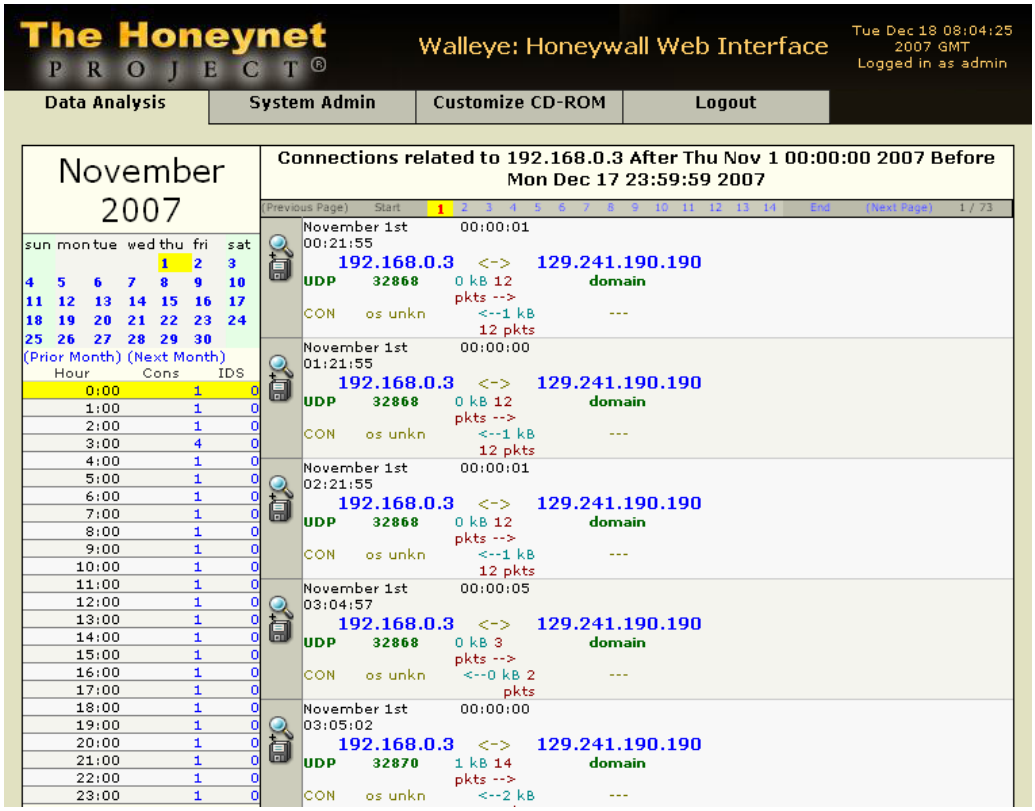



**Figure 31 : Walleye Web Interface**

There are options in snort_inline to detect and prevent network and port scanning by enabling **flow-portscan** preprocessor. But for test purpose we did not enabled that option most of the time. But we also tested enabling that option too.

### 6.2.3 Enable logging for system and services

Linux stores different log files for debugging and helping system administrators to find the problem. Some example log files are:

- /var/log/message: General system related message
- /var/log/secure: Authentication log
- /var/log/boot.log : System boot log
- /var/log/kern.log: Kernel log
- /var/log/utmp or /var/log/wtmp : Login related
- /var/log/btmp: Used to store information about failed logins.

These log files are useful to find error message and cause of the message. We also configure logging option for secure shell service. Sometimes logging will be the last point of intrusion detection. Attackers might leave trace of his/her activity in the log files which can be used later to at least know about the intrusion.

Except these three steps, we also used strong password in our network for login and other authentication mechanism. We used **OPC Security Tester** to identify known vulnerabilities of OPC server. To monitor the OPC server, we used Wireshark to capture live data continuously for three weeks. After preparing in this way, we decided to check all the logs regularly and run vulnerability assessment weekly.

## 6.3 Detecting and Analyzing Incidents

In the second phase of the incident response method, we detected and analyzed incidents based on the alert from snort_inline and other operating system specific logs. We can categorize the incidents based on the source of detection and vulnerabilities exploited. The system log files, OPC traffic sniffing and snort_inline alerts were the primary sources of detecting attacks and incidents. Based on our observation we have found attacks from red team as well as some outside attackers.

### 6.3.1 Network Scanning

We mentioned earlier that we did not prevent network scanning using tools like Nmap and Nessus. And we found alerts from snort_inline with title **"Attempted Information Leak"** which was logged due to Nmap scanning. The example is shown below.

```
[**] [1:469:3] ICMP PING NMAP [**]
[Classification: Attempted Information Leak] [Priority: 2]
04/23-08:20:18.134458 0:41:32:E2:41:25 -> 0:E:51:1D:13:32 type:0x800 len:0x3C
129.241.126.42 -> 129.241.252.115 ICMP TTL:45 TOS:0x0 ID:22908 IpLen:20
DgmLen:28
Type:8 Code:0 ID:62501 Seq:5403 ECHO
```

It is also possible to detect scanning by Nessus using **flow-portscan** option in snort_inline and **sfportscan** option in snort. So it is possible to detect the network scanning done by popular tools. This is not a direct attack but the result from Nessus can be used to launch attacks based on specific vulnerabilities of that host.

### 6.3.2 Sniffing OPC Traffic

Red team captured OPC traffic and could read the information because of no encryption and authentication mechanism in OPC tunneling. Earlier we have mentioned that we used Wireshark to monitor all OPC traffic for three weeks. While monitoring the traffic we discovered the fact that red team exploited. Matrikon OPC Tunneller by default does not provide any authentication or access control for connecting with other tunneller. By default, it also does not encrypt the packet transferred between tunnellers. One example **figure 32** shows the name of OPC server in the packet captured by Wireshark. Another example **figure 33** shows that items added or updated in OPC server is also transferred in plaintext. OPC traffic can be redirected or sniffed via ARP spoofing too. But sniffing traffic cannot be detected by IDS or log files from system. Network administrators need to monitor current traffic and try to find possible exploits and anomalies in traffic. Only IDS or IPS and system logs sometimes cannot detect attacks. All the sniffing attacks or reading packets cannot be traced using any tools. We should be at least careful not to give away information to attackers by using plain text data without any encryption. Red team successfully launched several man-in-the-middle attacks using this exploits. Although it is also necessary to use ARP spoofing for man-in-the-middle attack which is discussed in next section.

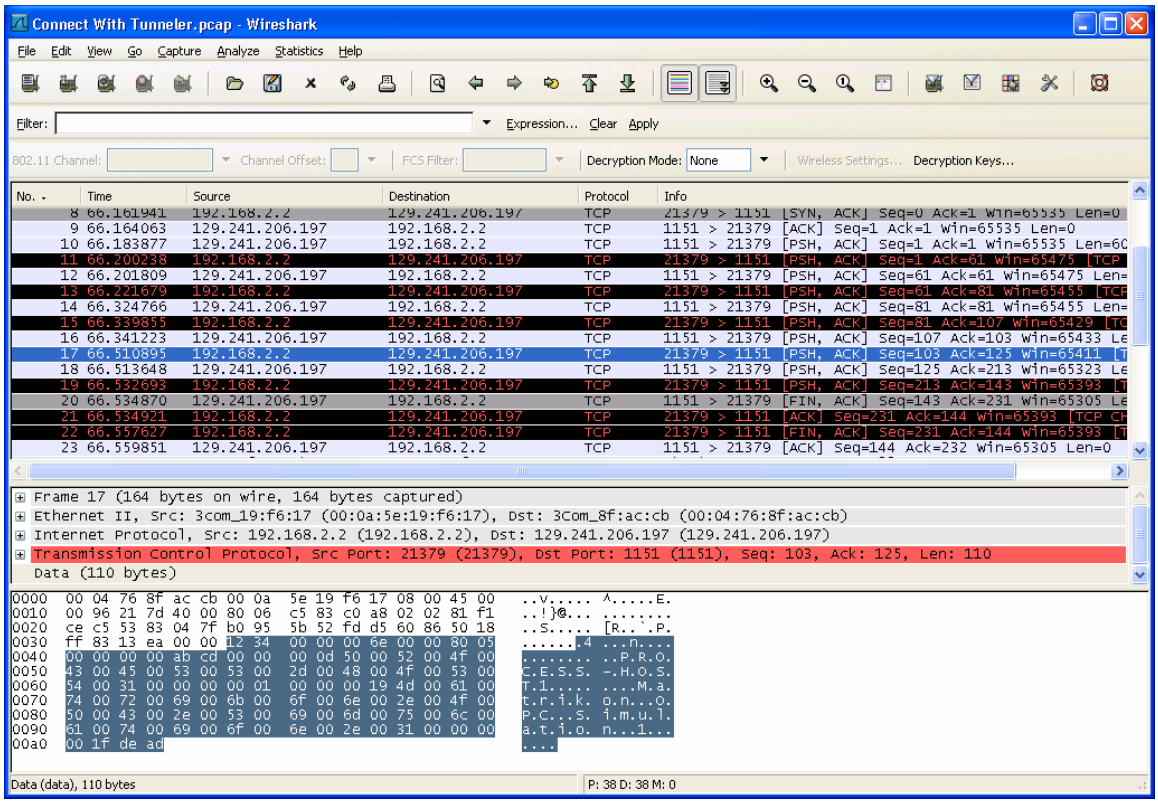



**Figure 32 : Information about OPC Server in Plaintext**

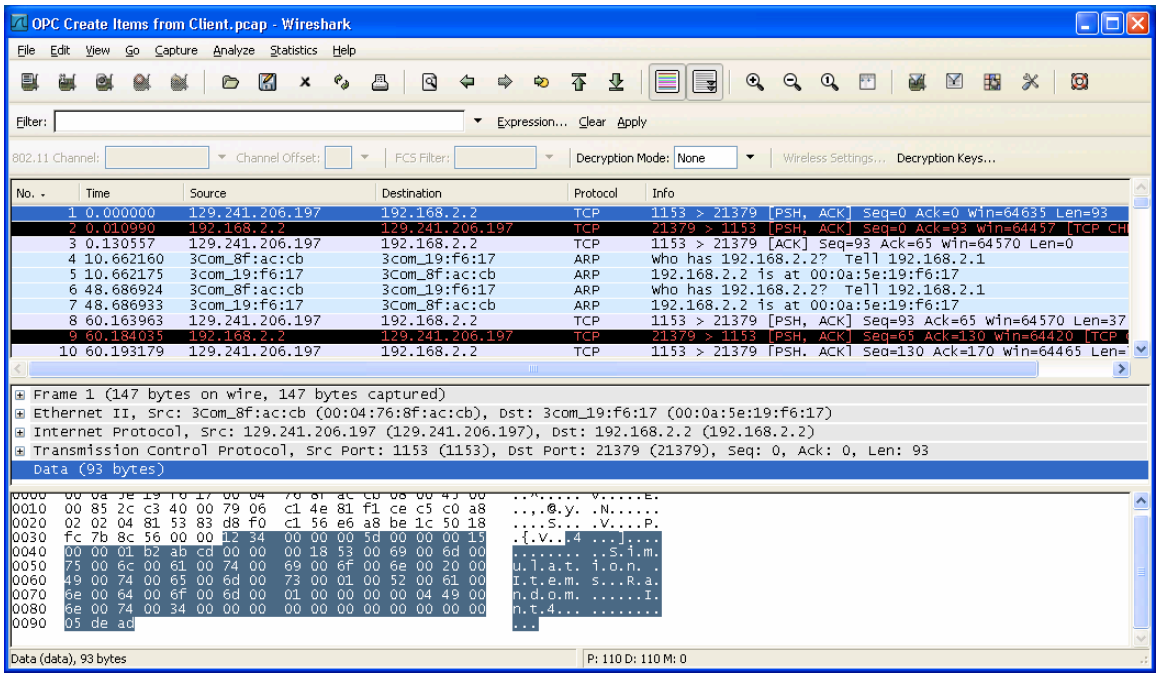



**Figure 33 : Information about new OPC Item in Plaintext**

### 6.3.3 Denial of Service Attack

Red team used ARP spoofing and SYN flooding to launch denial of service attack. Initially, the option to detect ARP spoofing attack by snort_inline was disabled. Due to this reason, it was tough for us to detect ARP spoofing without looking at ARP cache continuously. But another interesting symptom influenced us to test source of Denial of Service attack. The symptom was we could not connect with OPC server from remote location but we could ping our external gateway. This behavior confused us to check different log files and snort_inline alerts. But we failed to identify the exact reason using those logs. Later we started monitoring and experimenting with ARP cache and found that someone poisoned the ARP cache. One example of normal and poisoned ARP cache is shown below.

```
[root@firewall-dmz-admin ~]# arp -a
? (192.168.0.1) at 00:50:8B:0B:C4:59 [ether] on eth0
? (192.168.1.2) at 00:04:76:25:99:24 [ether] on eth1

[root@firewall-dmz-admin ~]# arp -a
? (192.168.0.1) at 00:08:74:2B:09:EF [ether] on eth0
? (192.168.0.3) at 00:08:74:2B:09:EF [ether] on eth0
```

We also tested ARP cache poisoning using **ettercap [54]** and found that it can be used to launch man-in-the-middle-attack, dropping packets. Thus it is an easy tool to prevent users accessing their services. But we can use snort to detect this attack. In that case we could use following example configuration options **arpspoof** to statically configure the IP address and MAC address of all hosts in our network in **/etc/snort.conf** file.

```
# arpspoof
#---------------------------------------
# Experimental ARP detection code from Jeff Nathan, detects ARP attacks,
# unicast ARP requests, and specific ARP mapping monitoring.  To make use of
# this preprocessor you must specify the IP and hardware address of hosts on
# the same layer 2 segment as you.  Specify one host IP MAC combo per line.
# Also takes a "-unicast" option to turn on unicast ARP request detection.
# Arpspoof uses Generator ID 112 and uses the following SIDS for that GID:

#  SID     Event description
# -----   -------------------
#   1       Unicast ARP request
#   2       Etherframe ARP mismatch (src)
#   3       Etherframe ARP mismatch (dst)
#   4       ARP cache overwrite attack

preprocessor arpspoof
# Example
preprocessor arpspoof_detect_host: 192.168.0.2 00:08:74:2B:0A:16
```

Then IDS would generate alert and log it in case of ARP poisoning. But this option is not feasible in case of large networks. In case of SYN Flooding, it's of no use to detect SYN flooding and then start sending RST message to reset connection. There is one preventive measure which uses IP address filtering technique. This technique will be discussed in responding to attack section. In case of crashing telnet service we did not use log for telnet session. And there is no common log where errors from telnet sessions are logged. These first three attacks were launched by red team.

### 6.3.4 Failed Login Attempt via SSH

Earlier we discussed about different log files used by Linux operating system to store information related to login. While checking the **/var/log/secure.1** file we found some interesting information. Here is the extract from that log file.

```
[root@router ~]# tail -f /var/log/secure.1
Dec 16 01:32:49 router sshd[24998]: input_userauth_request: invalid user zoya
Dec 16 01:32:49 router sshd[24997]: pam_unix(sshd:auth): check pass; user
unknown
Dec 16 01:32:49 router sshd[24997]: pam_unix(sshd:auth): authentication
failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=210.97.85.12
Dec 16 01:32:49 router sshd[24997]: pam_succeed_if(sshd:auth): error retrieving
information about user zoya
Dec 16 01:32:51 router sshd[24997]: Failed password for invalid user zoya from
210.97.85.12 port 59580 ssh2
Dec 16 01:32:51 router sshd[24998]: Received disconnect from 210.97.85.12: 11:
Bye Bye
Dec 16 04:02:04 router runuser: pam_unix(runuser:session): session opened for
user beaglidx by (uid=0)
Dec 16 04:02:07 router runuser: pam_unix(runuser:session): session closed for
user beaglidx
Dec 16 04:02:07 router runuser: pam_unix(runuser:session): session opened for
user beaglidx by (uid=0)
Dec 16 04:02:17 router runuser: pam_unix(runuser:session): session closed for
user beaglidx
```

It means that external attackers are trying to login using SSH and failed due to our strong password protection. Later we also found many failed login attempts. Even while writing this report we detected another failed login attempt. So far we know red team targeted telnet and HTTPS for finding login password using brute force attack. They used Hydra for this purpose. But Hydra does not support SSH protocol. So we think that all these attacks are coming from external attackers. Later we checked saved the result from **lastb** command which shows all failed login attempts in the system. The total number of failed logins was **66417** (from 3rd October). They tried to find password with brute force or dictionary attack using **guest**, **admin**, **www**, **host**, **router** as username. Although none of this login attempts have been successful so far; we should be careful about choosing password and known vulnerabilities of SSH implementation. A couple of logs with failed login attempt can be found in **Appendix B.**

### 6.3.5 Other Incidents

We also found trace of several incidents from snort_inline and iptables log in our PCS network. These are mainly due to outbound connection limiting or due to specific known vulnerabilities.

**Outbound connection limit:**
We used a honeypot inside the network (192.168.0.3) to try to browse a website in the Internet. Since we have previously set the connection limit very low (we set the limit to five), once it tries to connect, it hits the threshold almost immediately. Once it reaches the threshold, it starts dropping this outbound traffic.

We can see that the log snippet below shows that the outbound connection is logged and dropped as it has reached threshold of outgoing connections.

```
Nov 14 15:35:27 roo-test kernel: Drop TCP > 5 attemptsIN=br0 OUT=br0
PHYSIN=eth1 PHYSOUT=eth0 SRC=192.168.0.3 DST=209.132.177.50 LEN=60 TOS=0x00
PREC=0x00 TTL=64 ID=17618 DF PROTO =TCP SPT=33200 DPT=80 WINDOW=5840 RES=0x00
SYN URGP=0
```

In above log, the TCP connection is dropped as it has exceeded the maximum five attempts. We have set to allow a maximum of five TCP outbound attempts every minute. We can also change the number of outbound connections as well as the time interval the count should restart. The time interval can be in minutes, hourly, daily and weekly.

**Log Known Vulnerabilities:**

Snort_inline logs known vulnerabilities according to the configuration file. We checked **/var/log/snort/YYYYMMDD/{ snort_full | snort_fast }** directory to find alerts and logs. It logged following two types of attacks. On 14th November, there was a recorded log which is classified as “Attempted Information Leak”.

```
...
[**][1:882:5] WEB-CGI calendar access [**]
[Classification: Attempted Information Leak] [Priority: 2]
11/14-15:34:10.830792 192.168.0.3:47085 -> 64.233.183.104:80
TCP TTL:64 TOS:0x0 ID:40982 IpLen:20 DgmLen:760 DF
***AP*** Seq: 0x8F0A8A7D Ack: 0xC19DFEC9 Win: 0xFFFF TcpLen: 20
...
```

According to snort forum, The Oversize Request-uri directory is a number specified in http_inspect_config. By default this is 500 bytes. When it exceeds it logs following message:

```
...
[**] [119:15:1] (http_inspect) OVERSIZE REQUEST-URI DIRECTORY [**]
10/04-01:29:54.713542 192.168.0.3:51644 -> 64.236.29.103:80
TCP TTL:64 TOS:0x0 ID:16293 IpLen:20 DgmLen:1500 DF
***A**** Seq: 0x34FB31CA  Ack: 0xDF2E75AB  Win: 0x4FD8  TcpLen: 20
...
```

Last few attacks were logged and analyzed by blue team. And we did not find enough information about two known vulnerabilities.

## 6.4 Responding to Attacks

Before responding to the attack, following tables displays the overall summary of the incidents observed by blue team.

| Incident | Vulnerability | Detection | Response |
|---|---|---|---|
| Network Scanning | Know about services, ports and vulnerabilities | Alert or log from snort_inline | Log, Prevent |
| Sniff traffic | Communicatiion using Cleartext | Cannot be detected | Communication with encryption |
| Denial of Service, man-in-the-middle attack | ARP spoofing | ARP cache, snort log | Static ARP, MAC binding on Switch |
| Denial of Service using SYN Flooding | SYN Flooding | netstat command, need to use prevention mechanism | IP Address Filtering, SYN cookies |
| Crash Telnet Service | Using Hydra | System/telnet Log file | Remove Unused services like telnet |
| Failed Login attempts via SSH | Weak username and password | System log files, lastb command | Allow valid user and IP address to use SSH |
| Web.CGI calendar access | | snort_inline log | |
| Oversize Request-Uri Directory | Request-URI length | snort_inline log | Change http_inspect_config value |
| Crash by malformed TCP/IP packet | Available in Linux kernel version 2.6 | Nessus | Upgrade Kernel to version 2.6.7 |
| Man-in-the-middle-attack using remote desktop protocol | Decrypt Packets of Remote Desktop Connection | Nessus | Using SSL in Transport Layer while Using Remote Desktop |

**Table 4 : List of Incidents and their Technical Incident Response**

The list of incidents or attacks we have discovered or detected should be analyzed and take necessary actions to prevent damages. In real PCS network it is vital to respond properly based on specific incident. In the above table red team introduced first four incidents and most severe was Denial of Service attack and man-in-the-middle attack using ARP spoofing. Crashing services like telnet is another important incident which should be carefully handled. Sniffing traffic, especially OPC traffic which is in clear text format is another significant incident to consider. Now we discuss about responding to each of these events.

- Network scanning can be logged or prevented using snort or snort_inline by enabling a preprocessor **flow-portscan** or **sfportscan**.
- Sniffing traffic cannot be detected. So OPC tunneller should be configured to communicate using encryption which is discussed in section **4.10.3**
- Denial of service and man-in-the-middle-attack using ARP spoofing can be detected using **arpwatch [55]** or **snort [56]**. In case of small network static ARP cache in the gateway is a good solution. For large networks, MAC binding

for each address is managed by switch which is free from ARP spoofing. We used static ARP cache according to **Appendix B**.

- SYN flooding attack can be prevented using strict IP address filtering by network router. But we did not implement this filtering due to lack of time. Another way is to use SYN Cookies which allows a server to avoid dropping connections when the SYN queue fills up. Instead, the server behaves as if the SYN queue had been enlarged. We used following command to start SYN cookies.

```
echo 1 > /proc/sys/net/ipv4/tcp_syncookies
```

- Unused services like telnet should be removed from the PCS network.
- To prevent attackers to use brute force attack using SSH, we should only allow valid users with specific IP addresses to login. This can done by configuring firewall and access control list in ssh configuration file.
- Crash by malformed TCP/IP packet with invalid options can be avoided by upgrading Linux kernel to version 2.6.7
- The remote desktop protocol used by OPC server can be exploited using man-in-the-middle attack. This can be fixed by using SSL support which is available in latest version of remote desktop component.
- Finally, Oversize Request-Uri length can be changed to prevent false alarms in this case.
- Only single incident was found which has no detail information in snort website. We could not figure out how we should respond in that case.

## 6.5 Recommendation

After finishing this project, we have learned that it is important to start thinking about security of the PCS network. Most of the PCS network is still using DCOM based OPC server which does not ensure bullet-proof security of the OPC server. Its not possible to directly transit from DCOM based solution to Tunnelling based solution. But organizations should gradually move to more secure solution if their PCS network is already connected to the Internet for remote monitoring. They can follow these steps initially to assess the security of PCS network and its components.

- **Identifying Critical Nodes and Assess Vulnerabilities**: At first organization should identify critical nodes in the PCS network. Based on that result, they can use vulnerability scanners to identify their current security level in those critical nodes.
- **Prioritize and Mitigate Vulnerabilities**: Based on vulnerability analysis they can set priority of all vulnerabilities. Then they can further decide to mitigate that vulnerability sorted by priority order. These first two steps can be continued for several months to understand the importance of identifying and solving existing vulnerabilities.
- **Using Network Based IDS**: Organization should not only run vulnerability analysis periodically and try to mitigate those threats. In case of PCS network, it is very important to monitor the network continuously. Network based intrusion detection can be the starting point to monitor the network.
- **Assess vulnerabilities periodically based on IDS and VA:** Now its time to consider both real time traffic and static traffic. Again vulnerabilities should be sorted with priority. Then those threats should be mitigated.

We think these steps can be helpful for an organization to start understating about the security aspects of a PCS network connected with the Internet. These steps might mitigate vulnerabilities using technical solutions like IDS and VA. But Organization can provide specific security policy and guidelines to while providing the solutions. There are mainly three types of vulnerabilities PCS network should consider initially. These are:

- Operating System Specific Vulnerabilities
- Network Configuration Related Vulnerabilities
- PCS Protocol Specific Vulnerabilities

Finally, the best way to identify, detect and analyze attacks in a PCS network is to follow an incident handling method and plan for the whole organization.

# 7. Future Work

Identifying and detecting external threats was interesting project in which at first we configured a simulated PCS network. Later we used different tools to identify external threats in that network. This project can be extended or continued in different way. Some examples are:

- **Detect Internal Threats:** We have saved and mentioned about all the configuration files used in this project. Detecting internal threats in the same situation can be an interesting project as a future work. We think it is better to focus on specific types of attacks rather than any type of attacks. Then the scope of the project will be more defined and specific.

- **Organization and Cultural aspects in Incident handling:** We could not setup the mock organization and test the various aspects of incident handling method. This can be another future work to analyze external attacks from organizational and cultural point.

- **Using IDS and IPS:** In this project we have used Honeywall and snort_inline as IPS. But we have found that in some cases IDS can be useful when IPS cannot detect and prevent attacks. Using network based IDS and IPS in the PCS network can be another possible future work.

- **Using Host and Network based IDS:** Network based IDS only capture and analyze network traffic. It cannot identify threats targeted to specific host. In PCS network critical components like OPC server can use Host base IDS. And another network based IDS can help protecting the network. Analyzing this hybrid approach in PCS network is another interesting project.

- **Real PCS Environment:** Identifying and detecting attacks in a real PCS environment can be a good topic to work with in future. Instead a test network which can replicate the PCS network with minimal components can be designed first. Later different types of projects can be done in that test network.

- **Non-OPC specific Protocols and Servers:** In this project we configured Matrikon OPC server. Other protocols and servers used in PCS network can be tested and explored in future.

# 8. Conclusion

Through this project, we have realized that PCS networks face many threats that are similar to TCP/IP-based networks. PCS networks are themselves TCP/IP-based networks in many ways. Usually attackers use vulnerabilities of operating system, applications and network configuration. In case PCS network, PCS protocol specific attack is another source of the vulnerabilities. Most of these networks are also using OPC server based on DCOM which has known severe threats. The trend of connecting PCS networks to the Internet also meant that TCP/IP based attacks along with PCS protocol specific attacks will continue to increase in PCS networks. The awareness level in PCS network is still very low as there are not yet reported wide spread attacks on PCS networks. This, however, might change as when the black hat communities develop the mentality to attack PCS network.

In our project we designed and configured a simulated PCS network which provides OPC service using Matrikon OPC Server. Later we have used Honeywall, IPS, vulnerability scanners etc. to identify and detect attacks in the network. We found known vulnerabilities from different vulnerability databases for each of services we have offered by our network. Later we focused on the known attacks which can be detected by IPS and other system logs.

Red team used ARP spoofing, SYN Flooding, cleartext OPC communication to launch denial of service and man-in-the-middle attack. We detected most of attacks from IPS alerts and system logs. Surprisingly, our PCS network has also been attacked by external attackers who attempted to login using SSH. We have taken appropriate technical measures for most of incidents except SYN Flooding. Network Administrators and other personals responsible for incident handling should be always ready to deal with any kind of new incident. So incident handling is a continuous process as new threats are discovered in each day.

Apart from identifying responding to attacks, notification and learning from the threats are equally important. Response to attack from a human level is as important as the technical level. The coordination between the technical response team and other PCS personals are utmost important. Sometimes, it is the ignorance of users that might cause an external threat to happen.

“Identifying and Responding to External Threats in PCS network” has been a very enlightening project.

# References and Further Reading

**[1]** Tor Olav Grøtan, Martin Gilje Jaatun, Knut Øien, and Tor Onshus. The SeSa Method for Assessing Secure Remote Access to Safety Instrumented Systems, 1999.

**[2]** OPC Overview from OPC Foundation, 1998
URL: http://www.opcfoundation.org/Downloads.aspx?CM=1&CN=KEY&CI=282

**[3]** DCOM Technical Overview
URL: http://msdn2.microsoft.com/en-us/library/ms809340.aspx

**[4]** COM: Component Object Model Technologies
URL: http://www.microsoft.com/com/default.mspx

**[5]** Supervisory Control And Data Acquisition (SCADA)
URL: http://en.wikipedia.org/wiki/SCADA

**[6]** GAO-07-1036 Critical Infrastructure Protection: Multiple Efforts to Secure Control Systems are Under Way, but Challenges Remain
URL: http://www.gao.gov/htext/d071036.html

**[7]** Los Angeles County District Attorneys Office Two City Engineers Charged with Allegedly Hacking Into City's Traffic Computer (Los Angeles, California: Jan. 5, 2007)
URL: http://da.co.la.ca.us/mr/010507a.htm

**[8]** ISA Hackers Hit Pennsylvania Water System, (Research Triangle Park, North Carolina: November 2,, 2006).
URL:http://www.isa.org/content/contentgroups/news/2006/november29/hackers_hit_pennsylvania_water_system.htm

**[9]** OLE Concepts and Requirements Overview
URL: http://support.microsoft.com/kb/86008

**[10]** SINTEF Website
URL: http://www.sintef.no/

**[11]** Jan Tore Sørensen. Security in Industrial Networks, 2007.

**[12]** Modbus Protocol Specification
URL: http://www.modbus.org/specs.php

**[13]** OPC Tutorial from Matrikon
URL:http://www.matrikonopc.com/training/opc-multimedia tutorial/opc_tutorial_printable_version.pdf

**[14]** OPC Foundation Website
URL: http://www.opcfoundation.org/

**[15]** OPC Data Access Custom Interface Specification 3.00 from OPC Foundation
URL:http://www.opcfoundation.org/DownloadFile.aspx?CM=3&RI=67&CN=KEY&CI=283&CU=25

**[16]** OPC DataHub Manual
URL: http://www.opcdatahub.com/Docs/bookdho.html

**[17]** MatrikonOPC website
URL: http://www.matrikonopc.com

**[18]** Cogent Real-Time Systems Website
URL: http://www.opcdatahub.com/

**[19]** The Honeynet project website
URL: http://honeynet.org/

**[20]** Traditional IP Network Address Translator, RFC (Request for Comments) 3022
URL: http://tools.ietf.org/html/rfc3022

**[21]** John D. Howard and Thomas A. Longstaff, A Common Language for Computer Security Incidents, Sandia National Laboratories, Sandia Report: SAND98-8667

**[22]** Ivan Victor Krsul, Software Vulnerability Analysis, Ph.D. Dissertation, Computer Science Department, Purdue University, May, 1998.

**[23]** Computer Security Incident Handling Guide, The NIST publication SP 800-61, January 2004

**[24]** KVM Switch Website
URL: http://www.kvm-switches-online.com/

**[25]** The Secure Shell Protocol Architecture, RFC (Request for Comments) 4251
URL: http://tools.ietf.org/html/rfc4251

**[26]** HTTP Over TLS, RFC (Request for Comments) 2818
URL: http://tools.ietf.org/html/rfc2818

**[27]** Dynamic Host Configuration Protocol, RFC 2131
URL: http://www.faqs.org/rfcs/rfc2131.html

**[28]** Hypertext Transfer Protocol HTTP/1.1, RFC 2616
URL: http://www.faqs.org/rfcs/rfc2616.html

**[29]** MatrikonOPC Simulation Server Download URL
URL: http://www.matrikonopc.com/downloads/178/index.aspx

**[30]** PLC Tutor Site
URL: http://www.plcs.net/contents.shtml

**[31]** Internet Control Message Protocol, RFC 777
URL: http://www.networksorcery.com/enp/rfc/rfc777.txt

**[32]** Microsoft Remote Desktop Protocol
URL: http://msdn2.microsoft.com/en-us/library/aa383015.aspx

**[33]** Microsoft Remote Procedure Call
URL: http://msdn2.microsoft.com/en-us/library/aa378651.aspx

**[34]** Know Your Enemy: GenII Honeynets
URL: http://www.honeynet.org/papers/gen2/index.html

**[35]** ifconfig man page
URL: http://linuxreviews.org/man/ifconfig/

**[36]** Honeywall Hardware Requirements
URL: http://www.honeynet.org/tools/cdrom/roo/manual/3-require.html

**[37]** MatrikonOPC Tunneller Download URL
URL: http://www.matrikonopc.com/downloads/174/index.aspx

**[38]** MatrikonOPC Explorer Download URL
http://www.matrikonopc.com/downloads/176/index.aspx

**[39]** MatrikonOPC Download Center
http://www.matrikonopc.com/downloads/index.aspx

**[40]** File Transfer Protocol, RFC 959
URL: http://www.faqs.org/rfcs/rfc959.html

**[41]** RealVNC Website
URL: http://www.realvnc.com/

**[42]** National Vulnerability Database Version 2.0
URL: http://nvd.nist.gov/

**[43]** The Open Source Vulnerability Database
URL: http://osvdb.org/

**[44]** US-CERT Vulnerability Notes Database
URL: http://www.kb.cert.org/vuls/

**[45]** J. Carter D. Peterson E. Byres, M. Franz. OPC Security White Paper 2. Byres Research, 2007.

**[46]** Lluis Mora. OPC Server security considerations. In Proceedings of the SCADA Security Scientific Symposium 2007. Digital Bond Press, 2007.

**[47]** OPC Security Tester
URL: http://www.neutralbit.com/en/rd/opctest/

**[48]** Nmap (Network Mapper) Website
URL: http://insecure.org/nmap/

**[49]** Nessus Website
URL: http://www.nessus.org/nessus/

**[50]** Wireshark Website
URL: http://www.wireshark.org/

**[51]** Understanding Denial-of-Service Attacks
URL: http://www.us-cert.gov/cas/tips/ST04-015.html

**[52]** THC-Hydra Website
http://freeworld.thc.org/thc-hydra/

**[53]** Stig Ole Johnsen, Lillian Røstad, Børge Haugset and Maria B. Dahl, From Incident Response to Incident Response Management, SINTEF

**[54]** Ettercap website
URL: http://ettercap.sourceforge.net/

**[55]** Arpwatch man page
URL: http://linuxcommand.org/man_pages/arpwatch8.html

**[56]** Snort website
URL: http://www.snort.org/

# Appendix A: PCS Network Configuration

## [A.1] Original Network Topology Proposed in SINTEF Report [1]

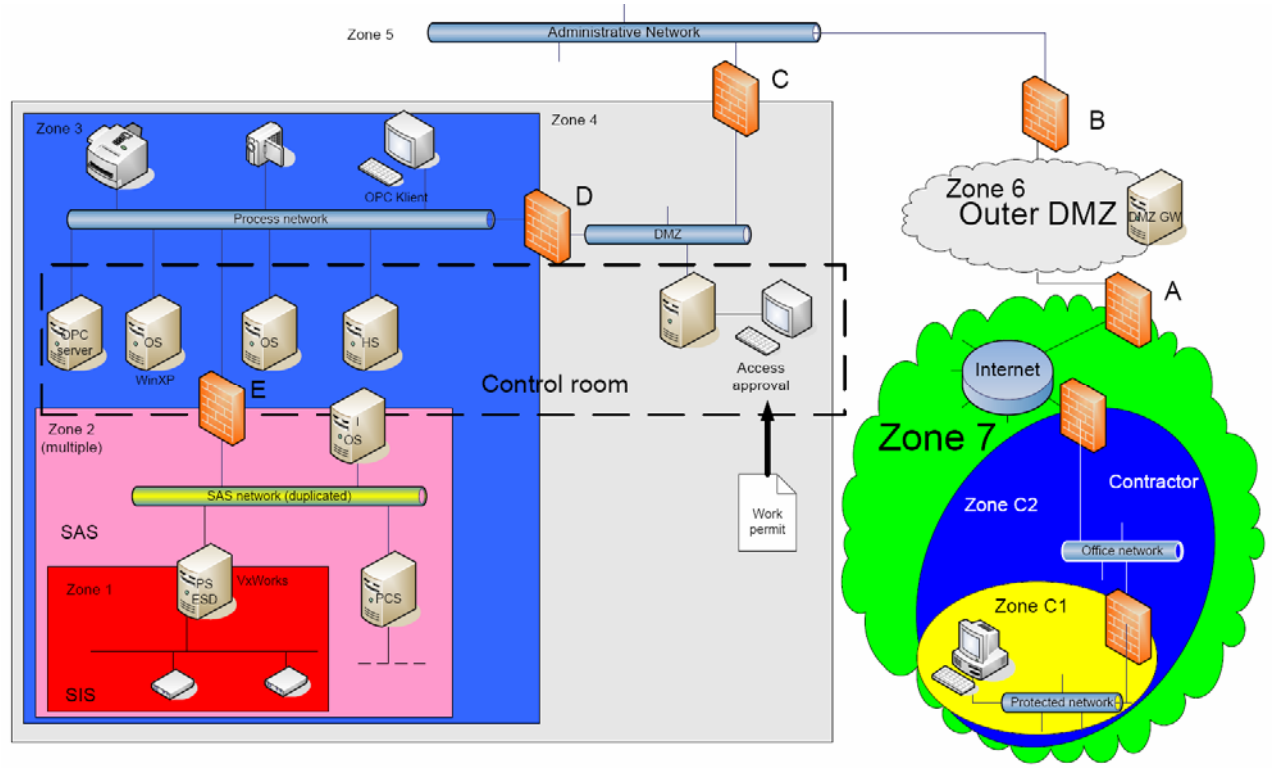



**Figure 34 : Proposed Network topology of the PCS network**

## [A.2] Command used to configure *iptables*

```
# Generated by iptables-save v1.3.5 on Wed Oct 31 11:10:59 2007
*nat
:PREROUTING ACCEPT [147679:11002621 create]
:POSTROUTING ACCEPT [32101:1776845 create]
:OUTPUT ACCEPT [24627:1323123 create]
-A PREROUTING -i eth0 -p tcp -m tcp –dport 443 -j DNAT –to-destination 10.0.0.2
-A PREROUTING -i eth0 -p tcp -m tcp –dport 3389 -j DNAT –to-destination 192.168.2.2
-A PREROUTING -i eth0 -p tcp -m tcp –dport 21379 -j DNAT –to-destination 192.168.2.2
-A PREROUTING -i eth0 -p tcp -m tcp –dport 135 -j DNAT –to-destination 192.168.2.2
-A POSTROUTING -s 10.0.0.0/255.0.0.0 -o eth0 -j SNAT –to-source 129.241.252.115
-A POSTROUTING -s 192.168.0.0/255.255.255.0 -o eth0 -j SNAT –to-source 129.241.252.115
-A POSTROUTING -s 192.168.1.0/255.255.255.0 -o eth0 -j SNAT –to-source 129.241.252.115
-A POSTROUTING -s 192.168.2.0/255.255.255.0 -o eth0 -j SNAT –to-source 129.241.252.115
COMMIT
# Completed on Wed Oct 31 11:10:59 2007[root@localhost sbin create]# cat iptables-30Oct
# Generated by iptables-save v1.3.5 on Wed Oct 31 11:10:59 2007
*nat
:PREROUTING ACCEPT [147679:11002621 create]
:POSTROUTING ACCEPT [32101:1776845 create]
:OUTPUT ACCEPT [24627:1323123 create]
-A PREROUTING -i eth0 -p tcp -m tcp –dport 443 -j DNAT –to-destination 10.0.0.2
-A PREROUTING -i eth0 -p tcp -m tcp –dport 3389 -j DNAT –to-destination 192.168.2.2
-A PREROUTING -i eth0 -p tcp -m tcp –dport 21379 -j DNAT –to-destination 192.168.2.2
-A PREROUTING -i eth0 -p tcp -m tcp –dport 135 -j DNAT –to-destination 192.168.2.2
-A POSTROUTING -s 10.0.0.0/255.0.0.0 -o eth0 -j SNAT –to-source 129.241.252.115
-A POSTROUTING -s 192.168.0.0/255.255.255.0 -o eth0 -j SNAT –to-source 129.241.252.115
-A POSTROUTING -s 192.168.1.0/255.255.255.0 -o eth0 -j SNAT –to-source 129.241.252.115
```

```
-A POSTROUTING -s 192.168.2.0/255.255.255.0 -o eth0 -j SNAT –to-source 129.241.252.115
COMMIT
# Completed on Wed Oct 31 11:10:59 2007
# Generated by iptables-save v1.3.5 on Wed Oct 31 11:10:59 2007
*filter
:INPUT ACCEPT [608556:142937683 create]
:FORWARD ACCEPT [1109761:725156402 create]
:OUTPUT ACCEPT [609814:66187798 create]
-A INPUT -d 10.0.0.0/255.0.0.0 -i eth2 -j REJECT –reject-with icmp-port-unreachable
-A FORWARD -d 10.0.0.0/255.0.0.0 -i eth2 -j REJECT –reject-with icmp-port-unreachable
COMMIT
# Completed on Wed Oct 31 11:10:59 2007

# Generated by iptables-save v1.3.5 on Wed Oct 31 11:10:59 2007
*filter
:INPUT ACCEPT [608556:142937683 create]
:FORWARD ACCEPT [1109761:725156402 create]
:OUTPUT ACCEPT [609814:66187798 create]
-A INPUT -d 10.0.0.0/255.0.0.0 -i eth2 -j REJECT –reject-with icmp-port-unreachable
-A FORWARD -d 10.0.0.0/255.0.0.0 -i eth2 -j REJECT –reject-with icmp-port-unreachable
COMMIT
# Completed on Wed Oct 31 11:10:59 2007
```

## [A.3] Status of *iptables* in router

```
Table: nat
Chain PREROUTING (policy ACCEPT 78696 packets, 6646K bytes)
num   pkts bytes target     prot opt in     out     source               destination
1        12   764 DNAT       tcp  --  eth0   *       0.0.0.0/0            0.0.0.0/0
tcp dpt:443 to:10.0.0.2
2         0     0 DNAT       tcp  --  eth0   *       0.0.0.0/0            0.0.0.0/0
tcp dpt:3389 to:192.168.2.2
3         0     0 DNAT       tcp  --  eth0   *       0.0.0.0/0            0.0.0.0/0
tcp dpt:21379 to:192.168.2.2
4         0     0 DNAT       tcp  --  eth0   *       0.0.0.0/0            0.0.0.0/0
tcp dpt:135 to:192.168.2.2

Chain POSTROUTING (policy ACCEPT 2516 packets, 133K bytes)
num   pkts bytes target     prot opt in     out     source               destination
1         1    89 SNAT       all  --  *      eth0    10.0.0.0/8           0.0.0.0/0
to:129.241.252.115
2       134  8707 SNAT       all  --  *      eth0    192.168.0.0/24       0.0.0.0/0
to:129.241.252.115
3       280 23478 SNAT       all  --  *      eth0    192.168.1.0/24       0.0.0.0/0
to:129.241.252.115
4       121  6469 SNAT       all  --  *      eth0    192.168.2.0/24       0.0.0.0/0
to:129.241.252.115

Chain OUTPUT (policy ACCEPT 2504 packets, 132K bytes)
num   pkts bytes target     prot opt in     out     source               destination

Table: filter
Chain INPUT (policy ACCEPT 88672 packets, 10M bytes)
num   pkts bytes target     prot opt in     out     source               destination
1         0     0 REJECT     all  --  eth2   *       0.0.0.0/0            10.0.0.0/8
reject-with icmp-port-unreachable

Chain FORWARD (policy ACCEPT 12229 packets, 8115K bytes)
num   pkts bytes target     prot opt in     out     source               destination
1         0     0 REJECT     all  --  eth2   *       0.0.0.0/0            10.0.0.0/8
reject-with icmp-port-unreachable

Chain OUTPUT (policy ACCEPT 87952 packets, 8004K bytes)
num   pkts bytes target     prot opt in     out     source               destination
```

# [A.4] Network Interface Configuration Appendix

## Static Configuration for Interfaces of Router

- **Configuration for eth0:** /etc/sysconfig/network-scripts/ifcfg-eth0

```
# 3Com Corporation 3c905C-TX/TX-M [Tornado]
DEVICE=eth0
BOOTPROTO=static
HWADDR=00:04:76:8f:a9:ee
BROADCAST=129.241.252.127
IPADDR=129.241.252.115
NETMASK=255.255.255.240
NETWORK=129.241.252.112
ONBOOT=yes
TYPE=Ethernet
```

- **Configuration for eth1:** /etc/sysconfig/network-scripts/ifcfg-eth1

```
# 3Com Corporation 3c905C-TX/TX-M [Tornado]
DEVICE=eth1
BOOTPROTO=static
HWADDR=00:04:75:80:1b:ee
BROADCAST=10.255.255.255
IPADDR=10.0.0.1
NETMASK=255.0.0.0
NETWORK=10.0.0.0
ONBOOT=yes
TYPE=Ethernet
```

- **Configuration for eth2:** /etc/sysconfig/network-scripts/ifcfg-eth2

```
# Intel Corporation 82557/8/9 [Ethernet Pro 100]
DEVICE=eth2
BOOTPROTO=static
HWADDR=00:50:8b:0b:c4:59
BROADCAST=192.168.0.255
IPADDR=192.168.0.1
NETMASK=255.255.255.0
NETWORK=192.168.0.0
ONBOOT=yes
TYPE=Ethernet
```

## Static Configuration for Interfaces of Honeywall

- **Configuration for eth0:** /etc/sysconfig/network-scripts/ifcfg-eth0

```
# Intel Corporation 82540EM Gigabit Ethernet Controller
DEVICE=eth0
HWADDR=00:08:74:2B:0A:28
ONBOOT=no
```

- **Configuration for eth1:** /etc/sysconfig/network-scripts/ifcfg-eth1

```
# Intel Corporation 82557/8/9 [Ethernet Pro 100]
DEVICE=eth1
HWADDR=00:90:27:BE:E1:FF
ONBOOT=no
```

- **Configuration for eth2:** /etc/sysconfig/network-scripts/ifcfg-eth2

```
# 3Com Corporation 3c905C-TX/TX-M [Tornado]
DEVICE=eth2
HWADDR=00:04:76:8F:AB:72
BOOTPROTO=static
BROADCAST=10.255.255.255
IPADDR=10.0.0.2
NETMASK=255.0.0.0
NETWORK=10.0.0.0
ONBOOT=yes
TYPE=Ethernet
```

### Static Configuration for Interfaces of DMZ Gateway

- **Configuration for eth0:** /etc/sysconfig/network-scripts/ifcfg-eth0

```
# Intel Corporation 82540EM Gigabit Ethernet Controller
DEVICE=eth0
BOOTPROTO=static
HWADDR=00:08:74:2b:0a:16
BROADCAST=192.168.0.255
IPADDR=192.168.0.2
NETMASK=255.255.255.0
NETWORK=192.168.0.0
ONBOOT=yes
TYPE=Ethernet
GATEWAY=192.168.0.1
```

- **Configuration for eth1:** /etc/sysconfig/network-scripts/ifcfg-eth1

```
# 3Com Corporation 3c905C-TX/TX-M [Tornado]
DEVICE=eth1
BOOTPROTO=static
HWADDR=00:01:03:2c:9e:4d
BROADCAST=192.168.1.255
IPADDR=192.168.1.1
NETMASK=255.255.255.0
NETWORK=192.168.1.0
ONBOOT=yes
TYPE=Ethernet
```

### Static Configuration for Interfaces of Host in DMZ Network

- **Configuration for eth0:** /etc/sysconfig/network-scripts/ifcfg-eth0

```
# Intel Corporation 82540EM Gigabit Ethernet Controller
DEVICE=eth0
BOOTPROTO=static
HWADDR=00:08:74:2B:09:EF
BROADCAST=192.168.0.255
IPADDR=192.168.0.3
NETMASK=255.255.255.0
NETWORK=192.168.0.0
ONBOOT=yes
TYPE=Ethernet
GATEWAY=192.168.0.1
```

### Static Configuration for Interfaces of Admin Gateway

- **Configuration for eth0:** /etc/sysconfig/network-scripts/ifcfg-eth0

```
# 3Com Corporation 3c905C-TX/TX-M [Tornado]
DEVICE=eth0
BOOTPROTO=static
HWADDR=00:04:76:25:99:24
BROADCAST=192.168.1.255
IPADDR=192.168.1.2
NETMASK=255.255.255.0
NETWORK=192.168.1.0
ONBOOT=yes
TYPE=Ethernet
GATEWAY=192.168.1.1
```

- **Configuration for eth1:** /etc/sysconfig/network-scripts/ifcfg-eth1

```
# 3Com Corporation 3c905C-TX/TX-M [Tornado]
DEVICE=eth1
BOOTPROTO=static
HWADDR=00:04:76:8f:ac:cb
BROADCAST=192.168.2.255
IPADDR=192.168.2.1
NETMASK=255.255.255.0
NETWORK=192.168.2.0
ONBOOT=yes
TYPE=Ethernet
```

## Static Configuration for Interfaces of Host in Process Network

- **Result from ipconfig command :**

```
C:\Documents and Settings\Student>ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection 2:

        Connection-specific DNS Suffix  . :
        IP Address. . . . . . . . . . . . : 192.168.2.2
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 192.168.2.1
```

- **Configuring Static IP in Windows XP:**

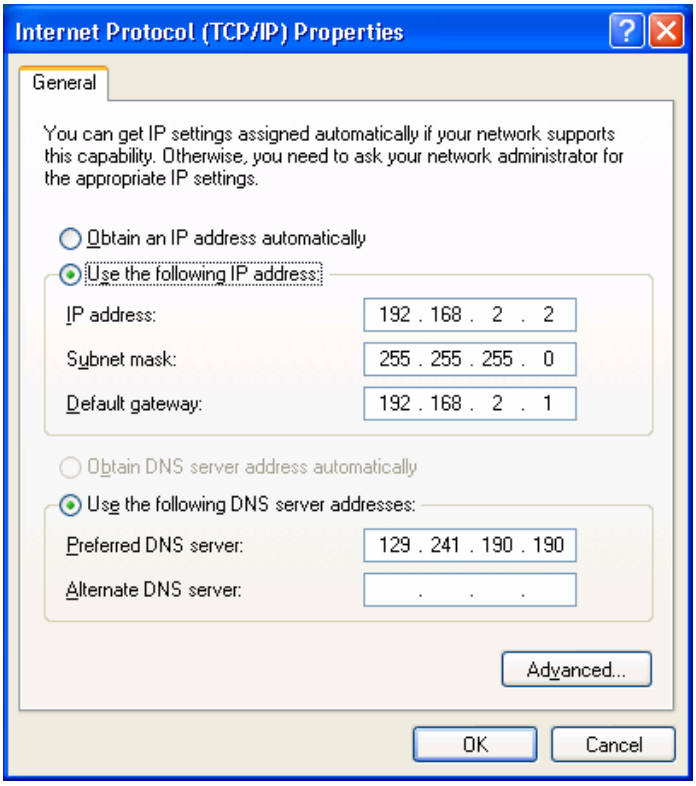



**Figure 35 : Network Interface configuration in Windows XP for OPC Server**

## [A.5] Routing Configuration File Appendix

**Static Routing Configuration for Router:**

- **Configure Default Gateway:** Following line is added in the file /etc/sysconfig/network-scripts/ifcfg-eth0

```
GATEWAY=129.241.252.113
```

- **Add Static Routes in File:** /etc/sysconfig/network-scripts/route-eth2

```
GATEWAY0=192.168.0.2
NETMASK0=255.255.255.0
ADDRESS0=192.168.1.0

GATEWAY1=192.168.0.2
NETMASK1=255.255.255.0
ADDRESS1=192.168.2.0
```

**Static Routing Configuration for Honeywall:**

- **Configure Default Gateway in File:** Following line is added in the file /etc/sysconfig/network

```
GATEWAY=10.0.0.1
```

**Static Routing Configuration for DMZ Gateway:**

- **Configure Default Gateway in File:** Following line is added in the file /etc/sysconfig/network-scripts/ifcfg-eth0

```
GATEWAY=192.168.0.1
```

- **Add Static Routes in File:** /etc/sysconfig/network-scripts/route-eth1

```
GATEWAY0=192.168.1.2
NETMASK0=255.255.255.0
ADDRESS0=192.168.2.0
```

**Static Routing Configuration for Admin Gateway:**

- **Configure Default Gateway in File:** Following line is added in the file /etc/sysconfig/network-scripts/ifcfg-eth0

```
GATEWAY=192.168.1.1
```

**Static Routing Configuration for Host in DMZ Network:**

- **Configure Default Gateway in File:** Following line is added in the file /etc/sysconfig/network-scripts/ifcfg-eth0

```
GATEWAY=192.168.0.1
```

**Static Routing Configuration for Host in Process Network:**
Check Network Interface Configuration in **Appendix A**

## [A.6] Default honeywall.conf file

```
##########################################################################
#
# $Id: honeywall.conf 4552 2006-10-17 01:06:51Z esammons $
#
##############################################
#
# Copyright (C) <2005> <The Honeynet Project>
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or (at
# your option) any later version.
#
# This program is distributed in the hope that it will be useful, but
# WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
# General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307
# USA
#
##############################################

#
# This file is the Honeywall import file (aka "honeywall.conf").
# It is a list of VARIABLE=VALUE tuples (including comments as
# necessary, # such as this) and whitespace lines.
#
# note: DO NOT surround values in quotation marks
#
##########################################################################

############################
# Site variables that are  #
# global to all honeywalls #
# at a site.               #
############################

# Specify the IP address(es) and/or networks that are allowed to connect
# to the management interface.  Specify any to allow unrestricted access.
# [Valid argument: IP address(es) | IP network(s) in CIDR notation | any]
HwMANAGER=10.10.10.0/24

# Specify the port on which SSHD will listen
# NOTE: Automatically aded to the list of TCP ports allowed in by IPTables
# [Valid argument: TCP (port 0 - 65535)]
HwSSHD_PORT=22

# Specify whether or not root can login remotely over SSH
# [Valid argument: yes | no]
HwSSHD_REMOTE_ROOT_LOGIN=no

# NTP Time server(s)
# [Valid argument: IP address]
HwTIME_SVR=
```

```
############################
# Local variables that are #
# specific to each         #
# honeywall at a site.     #
############################

# Specify the system hostname
# [Valid argument: string ]
HwHOSTNAME=roo-test

# Specify the system DNS domain
# [Valid argument: string ]
HwDOMAIN=localdomain

#Start the Honeywall on boot
# [Valid argument: yes | no]
HwHONEYWALL_RUN=no

# To use a headless system.
# [Valid argument: yes | no]
HwHEADLESS=no


# This Honeywall's public IP address(es)
# [Valid argument: IP address | space delimited IP addresses]
HwHPOT_PUBLIC_IP=10.0.0.20

# DNS servers honeypots are allowed to communicate with
# [Valid argument: IP address | space delimited IP addresses]
HwDNS_SVRS=

# To restrict DNS access to a specific honeypot or group of honeypots, list
# them here, otherwise leave this variable blank
# [Valid argument: IP address | space delimited IP addresses | blank]
HwDNS_HOST=

# The name of the externally facing network interface
# [Valid argument: eth* | br* | ppp*]
HwINET_IFACE=eth0

# The name of the internally facing network interface
# [Valid argument: eth* | br* | ppp*]
HwLAN_IFACE=eth1

# The IP internal connected to the internally facing interface
# [Valid argument: IP network in CIDR notation]
HwLAN_IP_RANGE=10.0.0.0/24

# The IP broadcast address for internal network
# [Valid argument: IP broadcast address]
HwLAN_BCAST_ADDRESS=10.0.0.255

# Enable QUEUE support to integrate with Snort-Inline filtering
# [Valid argument: yes | no]
HwQUEUE=yes

# The unit of measure for setting oubtbound connection limits
# [Valid argument: second, minute, hour, day, week, month, year]
HwSCALE=hour

# The number of TCP connections per unit of measure (HwScale)
# [Valid argument: integer]
HwTCPRATE=20
```

```
# The number of UDP connections per unit of measure (HwSCALE)
# [Valid argument: integer]
HwUDPRATE=20

# The number of ICMP connections per unit of measure (HwSCALE)
# [Valid argument: integer]
HwICMPRATE=50

# The number of other IP connections per unit of measure (HwSCALE)
# [Valid argument: integer]
HwOTHERRATE=10

# Enable the SEBEK collector which delivers keystroke and files
# to a remote system even if an attacker replaces daemons such as sshd
# [Valid argument: yes | no]
HwSEBEK=no

# Enable the Walleye Web interface.
#[Valid argument: yes | no]
HwWALLEYE=yes

# Specify whether whether to drop SEBEK packets or allow them to be sent
# outside of the Honeynet.
# [Valid argument: ACCEPT | DROP]
HwSEBEK_FATE=DROP

# Specify the SEBEK destination host IP address
# [Valid argument: IP address]
HwSEBEK_DST_IP=10.0.0.253

# Specify the SEBEK destination port
# [Valid argument: port]
HwSEBEK_DST_PORT=1101

# Enable SEBEK logging in the Honeywall firewall logs
# [Valid argument: yes | no]
HwSEBEK_LOG=no


# Specify whether the dialog menu is to be started on login to TTY1
# [Valid argument: yes | no ]
HwMANAGE_DIALOG=yes

# Specify whether management port is to be activated on start or not.
# [Valid argument: yes | no ]
HwMANAGE_STARTUP=yes

# Specy the network interface for remote management.  If set to br0, it will
# assign MANAGE_IP to the logical bridge interface and allow its use as a
# management interface.  Set to none to disable the management interface.
# [Valid argument: eth* | br* | ppp* | none]
HwMANAGE_IFACE=eth2

# IP of management Interface
# [Valid argument: IP address]
HwMANAGE_IP=10.10.10.66

# Netmask of management Interface
# [Valid argument: IP netmask]
HwMANAGE_NETMASK=255.255.255.0

# Default Gateway of management Interface
```

```
# [Valid argument: IP address]
HwMANAGE_GATEWAY=10.10.10.1

# DNS Servers of management Interface
# [Valid argument: space delimited IP addresses]
HwMANAGE_DNS=

# TCP ports allowed into the management interface.
# Do NOT include the SSHD port.  It will automatically be included
# [Valid argument: space delimited list of TCP ports]
HwALLOWED_TCP_IN=443

# Specify whether or not the Honeywall will restrict outbound network
# connections to specific destination ports.  When bridge mode is utilized,
# a management interface is required to restrict outbound network connections.
# [Valid argument: yes | no]
HwRESTRICT=yes

# Specity the TCP destination ports Honeypots can send network traffic to.
# [Valid argument: space delimited list of UDP ports]
HwALLOWED_TCP_OUT=22 25 43 80 443

# Specity the UDP destination ports Honeypots can send network traffic to.
# [Valid argument: space delimited list of UDP ports]
HwALLOWED_UDP_OUT=53 123

# Specify whether or not to start swatch and email alerting.
# [Valid argument: yes | no]
HwALERT=no

# Specify email address to use for email alerting.
# [Valid argument: any email address]
HwALERT_EMAIL=root@localhost.localdomain

# NIC Module List - Set this to the number and order you wish
# to load NIC drivers, such that you get the order you want
# for eth0, eth1, eth2, etc.
# [Valid argument: list of strings]
#
# Example: eepro100 8139too
HwNICMODLIST=

# Blacklist, Whitelist, and Fencelist features.
# [Valid argument: string ]
HwFWBLACK=/etc/blacklist.txt

# [Valid argument: string ]
HwFWWHITE=/etc/whitelist.txt

# [Valid argument: string ]
HwFWFENCE=/etc/fencelist.txt

# [Valid argument: yes | no]
HwBWLIST_ENABLE=no

# [Valid argument: yes | no]
HwFENCELIST_ENABLE=no

# The following feature allows the roo to allow attackers into the
# honeypots but they can't send packets out...
# [Valid argument: yes | no]
HwROACHMOTEL_ENABLE=no
```

```
# Disables BPF filtering based on the contents of HwHPOT_PUBLIC_IP
# and the black and white list contained within HwFWBLACK and HwFWWHITE
# if the HwBWLIST_ENABLE is on.  Other wise, it just filters based on
# the contents of HwHPOT_PUBLIC_IP
# [Valid argument: yes | no]
HwBPF_DISABLE=no

# This capability is not yet implemented in roo.  The variable
# has been commented out for this reason. dittrich - 02/08/05
# Options for hard drive tuning (if needed).
# [Valid argument: string ]
# Example: -c 1 -m 16 -d
HwHWPARMOPTS=

# Should we swap capslock and control keys?
HwSWAP_CAPSLOCK_CONTROL=no

##############################################################################
# Snort Rule Update Variables
##############################################################################
# Enable or disable automatic snort rule updates
# [Valid argument: yes | no]
HwRULE_ENABLE=no

# Automatically restart snort and snort_inline when automatic updates are
# applied and when calls to update IDS or IPs rules?
# [Valid argument: yes | no]
HwSNORT_RESTART=no

# Oink Code - Required by Oinkmaster to retrieve VRT rule updates
# See: /hw/docs/README.snortrules or
#      http://www.honeynet.org/tools/cdrom/roo/manual/
# for instructions on how to obtain it (Free registration).
# [Valid argument: ~40 char alphanum string]
HwOINKCODE=

# Day automatic snort rule updates should be retrieved (for weekly updates)
# For daily updates, set this to ""
# [Valid argument: sun | mon | tue | wed | thu | fri | sat]
HwRULE_DAY=sat

# Hour of day snort rules updates should be retrieved
# [Valid argument: 0 | 1 | 2 | ... | 23] (0 is Midnight, 12 is noon, 23 is
11PM)
HwRULE_HOUR=3

##############################################################################
# Pcap and DB data retention settings
# Currenrly ONLY used when Pcap/DB purge scripts are called
# Pcap/DB data *is NOT* automatically purged
##############################################################################
# Days to retain Pcap data.  This will be used *IF* /dlg/config/purgePcap.pl
# is called with NO arguments.
# NOTE: Override this by supplying the number of days as an argument ala:
# /dlg/config/purgePcap.pl <days>
HwPCAPDAYS=45

# Days to retain DB data.  This will be used *IF* /dlg/config/purgeDB.pl
# is called with NO arguments.
# NOTE: Override this by supplying the number of days as an argument ala:
# /dlg/config/purgeDB.pl <days>
HwDBDAYS=180
```

```
################################################################################
# NAT mode is no longer supported.
# Don't mess with anything below here unless you know what you're
# doing! Don't say we didn't warn you, and don't try logging a bugzilla
# request to clean up the mess!
################################################################################

# Space delimited list of Honeypot ips
# NOTE: MUST HAVE SAME NUMBER OF IPS AS PUBLIC_IP VARIABLE.
# [Valid argument: IP address]
#HwHPOT_PRIV_IP_FOR_NAT=

# Specify the IP address of the honeywall's internal (i.e. gateway
# IP for NAT) IP address.  This is only used in NAT mode.
# [Valid argument: IP address ex: 192.168.10.1]
#HwPRIV_IP_FOR_NAT=


# Specify the IP netmask for interface alises.  One aliases will be created
# on the external interface for each Honeypot when in NAT mode only.
# [Valid argument: IP netmask]
#HwALIAS_MASK_FOR_NAT=255.255.255.0


# End of honeywall.conf parameters
```

# Appendix B: Detecting and Analyzing Incidents

## [B.1] Logs with failed login attempts

```
[root@router log]# tail -f /var/log/secure
Dec 17 22:24:09 router sshd[21564]: input_userauth_request: invalid user
firebird
Dec 17 22:24:09 router sshd[21563]: pam_unix(sshd:auth): check pass; user
unknown
Dec 17 22:24:09 router sshd[21563]: pam_unix(sshd:auth): authentication
failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=218.18.107.39
Dec 17 22:24:09 router sshd[21563]: pam_succeed_if(sshd:auth): error retrieving
information about user firebird
Dec 17 22:24:12 router sshd[21563]: Failed password for invalid user firebird
from 218.18.107.39 port 48532 ssh2
Dec 17 22:24:12 router sshd[21564]: Received disconnect from 218.18.107.39: 11:
Bye Bye
Dec 17 22:24:16 router sshd[21565]: pam_unix(sshd:auth): authentication
failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=218.18.107.39 user=root
Dec 17 22:24:18 router sshd[21565]: Failed password for root from 218.18.107.39
port 48647 ssh2
Dec 17 22:24:19 router sshd[21566]: Received disconnect from 218.18.107.39: 11:
Bye Bye
Dec 17 22:24:29 router sshd[21568]: Connection closed by 218.18.107.39

[root@router log]# tail -f /var/log/secure.1
Dec 16 01:32:49 router sshd[24998]: input_userauth_request: invalid user zoya
Dec 16 01:32:49 router sshd[24997]: pam_unix(sshd:auth): check pass; user
unknown
Dec 16 01:32:49 router sshd[24997]: pam_unix(sshd:auth): authentication
failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=210.97.85.12
Dec 16 01:32:49 router sshd[24997]: pam_succeed_if(sshd:auth): error retrieving
information about user zoya
Dec 16 01:32:51 router sshd[24997]: Failed password for invalid user zoya from
210.97.85.12 port 59580 ssh2
Dec 16 01:32:51 router sshd[24998]: Received disconnect from 210.97.85.12: 11:
Bye Bye
Dec 16 04:02:04 router runuser: pam_unix(runuser:session): session opened for
user beaglidx by (uid=0)
Dec 16 04:02:07 router runuser: pam_unix(runuser:session): session closed for
user beaglidx
Dec 16 04:02:07 router runuser: pam_unix(runuser:session): session opened for
user beaglidx by (uid=0)
Dec 16 04:02:17 router runuser: pam_unix(runuser:session): session closed for
user beaglidx

[root@router log]# tail -f /var/log/secure.2
Dec 8 14:49:08 router sshd[2558]: Did not receive identification string from
195.1.78.226
Dec 8 14:49:08 router sshd[2559]: Did not receive identification string from
195.1.78.226
Dec 8 14:49:08 router sshd[2560]: Did not receive identification string from
195.1.78.226
Dec 8 14:49:09 router sshd[2561]: Did not receive identification string from
195.1.78.226
Dec 8 14:49:09 router sshd[2562]: Did not receive identification string from
195.1.78.226
Dec 8 14:49:09 router sshd[2563]: Did not receive identification string from
195.1.78.226
Dec 9 04:02:44 router runuser: pam_unix(runuser:session): session opened for
user beaglidx by (uid=0)
```

```
Dec  9 04:02:47 router runuser: pam_unix(runuser:session): session closed for
user beaglidx
Dec  9 04:02:47 router runuser: pam_unix(runuser:session): session opened for
user beaglidx by (uid=0)
Dec  9 04:02:57 router runuser: pam_unix(runuser:session): session closed for
user beaglidx

[root@router log]# tail -f /var/log/secure.3
Dec  1 22:07:45 router sshd[17954]: input_userauth_request: invalid user test
Dec   1 22:07:45 router sshd[17953]: pam_unix(sshd:auth): check pass; user
unknown
Dec    1 22:07:45 router sshd[17953]: pam_unix(sshd:auth): authentication
failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=221.204.251.32
Dec  1 22:07:45 router sshd[17953]: pam_succeed_if(sshd:auth): error retrieving
information about user test
Dec  1 22:07:48 router sshd[17953]: Failed password for invalid user test from
221.204.251.32 port 12093 ssh2
Dec   1 22:07:48 router sshd[17954]: Received disconnect from 221.204.251.32:
11: Bye Bye
Dec  2 04:02:06 router runuser: pam_unix(runuser:session): session opened for
user beaglidx by (uid=0)
Dec  2 04:02:10 router runuser: pam_unix(runuser:session): session closed for
user beaglidx
Dec  2 04:02:10 router runuser: pam_unix(runuser:session): session opened for
user beaglidx by (uid=0)
Dec  2 04:02:20 router runuser: pam_unix(runuser:session): session closed for
user beaglidx

[root@router log]# tail -f /var/log/secure.4
Nov 24 06:02:13 router sshd[30138]: pam_succeed_if(sshd:auth): error retrieving
information about user test
Nov 24 06:02:15 router sshd[30138]: Failed password for invalid user test from
200.226.124.15 port 51089 ssh2
Nov 24 06:02:15 router sshd[30139]: Received disconnect from 200.226.124.15:
11: Bye Bye
Nov 25 00:02:25 router sshd[31904]: Did not receive identification string from
62.128.149.10
Nov  25  01:51:01  router  sshd[32081]:  Accepted  password  for  root  from
129.241.126.142 port 2343 ssh2
Nov 25 01:51:01 router sshd[32083]: pam_unix(sshd:session): session opened for
user root by root(uid=0)
Nov 25 04:02:02 router runuser: pam_unix(runuser:session): session opened for
user beaglidx by (uid=0)
Nov 25 04:02:06 router runuser: pam_unix(runuser:session): session closed for
user beaglidx
Nov 25 04:02:06 router runuser: pam_unix(runuser:session): session opened for
user beaglidx by (uid=0)
Nov 25 04:02:16 router runuser: pam_unix(runuser:session): session closed for
user beaglidx
```

## [B.2] Result from lastb command

```
ROOT     ssh:notty    lekestue.ed.ntnu Tue Dec 18 05:06 - 05:06  (00:00)
ROOT     ssh:notty    lekestue.ed.ntnu Tue Dec 18 05:06 - 05:06  (00:00)
ROOT     ssh:notty    lekestue.ed.ntnu Tue Dec 18 05:06 - 05:06  (00:00)
ROOT     ssh:notty    lekestue.ed.ntnu Tue Dec 18 05:06 - 05:06  (00:00)
bin      ssh:notty    dhcp206-197.ed.n Tue Dec 18 02:33 - 02:33  (00:00)
root     ssh:notty    dhcp206-197.ed.n Tue Dec 18 02:33 - 02:33  (00:00)
nonexist ssh:notty    dhcp206-197.ed.n Tue Dec 18 02:33 - 02:33  (00:00)
root     ssh:notty    dhcp206-197.ed.n Tue Dec 18 02:33 - 02:33  (00:00)
guest    ssh:notty    dhcp206-197.ed.n Tue Dec 18 02:33 - 02:33  (00:00)
nonexist ssh:notty    dhcp206-197.ed.n Tue Dec 18 02:33 - 02:33  (00:00)
n3ssus   ssh:notty    dhcp206-197.ed.n Tue Dec 18 02:33 - 02:33  (00:00)
n3ssus   ssh:notty    dhcp206-197.ed.n Tue Dec 18 02:33 - 02:33  (00:00)
bin      ssh:notty    dhcp206-197.ed.n Tue Dec 18 02:15 - 02:15  (00:00)
root     ssh:notty    dhcp206-197.ed.n Tue Dec 18 02:15 - 02:15  (00:00)
nonexist ssh:notty    dhcp206-197.ed.n Tue Dec 18 02:15 - 02:15  (00:00)
root     ssh:notty    dhcp206-197.ed.n Tue Dec 18 02:15 - 02:15  (00:00)
guest    ssh:notty    dhcp206-197.ed.n Tue Dec 18 02:15 - 02:15  (00:00)
nonexist ssh:notty    dhcp206-197.ed.n Tue Dec 18 02:15 - 02:15  (00:00)
n3ssus   ssh:notty    dhcp206-197.ed.n Tue Dec 18 02:15 - 02:15  (00:00)
n3ssus   ssh:notty    dhcp206-197.ed.n Tue Dec 18 02:15 - 02:15  (00:00)
root     ssh:notty    218.18.107.39    Mon Dec 17 22:24 - 22:24  (00:00)
firebird ssh:notty    218.18.107.39    Mon Dec 17 22:24 - 22:24  (00:00)
firebird ssh:notty    218.18.107.39    Mon Dec 17 22:24 - 22:24  (00:00)
root     ssh:notty    lekestue.ed.ntnu Mon Dec 17 21:01 - 21:01  (00:00)
4Dgifts  ssh:notty    129.241.252.123  Mon Dec 17 12:46 - 12:46  (00:00)
root     ssh:notty    129.241.252.123  Mon Dec 17 12:46 - 12:46  (00:00)
EZsetup  ssh:notty    129.241.252.123  Mon Dec 17 12:46 - 12:46  (00:00)
OutOfBox ssh:notty    129.241.252.123  Mon Dec 17 12:46 - 12:46  (00:00)
bin      ssh:notty    129.241.252.123  Mon Dec 17 12:45 - 12:45  (00:00)
root     ssh:notty    129.241.252.123  Mon Dec 17 12:45 - 12:45  (00:00)
nonexist ssh:notty    129.241.252.123  Mon Dec 17 12:45 - 12:45  (00:00)
root     ssh:notty    129.241.252.123  Mon Dec 17 12:45 - 12:45  (00:00)
guest    ssh:notty    129.241.252.123  Mon Dec 17 12:45 - 12:45  (00:00)
nonexist ssh:notty    129.241.252.123  Mon Dec 17 12:45 - 12:45  (00:00)
n3ssus   ssh:notty    129.241.252.123  Mon Dec 17 12:45 - 12:45  (00:00)
n3ssus   ssh:notty    129.241.252.123  Mon Dec 17 12:45 - 12:45  (00:00)
zoya     ssh:notty    211.157.109.37   Mon Dec 17 03:48 - 03:48  (00:00)
zoya     ssh:notty    211.157.109.37   Mon Dec 17 03:48 - 03:48  (00:00)
zinoviya ssh:notty    211.157.109.37   Mon Dec 17 03:48 - 03:48  (00:00)
zinoviya ssh:notty    211.157.109.37   Mon Dec 17 03:48 - 03:48  (00:00)
zinoviy  ssh:notty    211.157.109.37   Mon Dec 17 03:48 - 03:48  (00:00)
zinoviy  ssh:notty    211.157.109.37   Mon Dec 17 03:48 - 03:48  (00:00)
zinaida  ssh:notty    211.157.109.37   Mon Dec 17 03:48 - 03:48  (00:00)
zinaida  ssh:notty    211.157.109.37   Mon Dec 17 03:48 - 03:48  (00:00)
zhenya   ssh:notty    211.157.109.37   Mon Dec 17 03:48 - 03:48  (00:00)
zhenya   ssh:notty    211.157.109.37   Mon Dec 17 03:48 - 03:48  (00:00)
zhannoch ssh:notty    211.157.109.37   Mon Dec 17 03:48 - 03:48  (00:00)
zhannoch ssh:notty    211.157.109.37   Mon Dec 17 03:48 - 03:48  (00:00)
zhanna   ssh:notty    211.157.109.37   Mon Dec 17 03:48 - 03:48  (00:00)
zhanna   ssh:notty    211.157.109.37   Mon Dec 17 03:48 - 03:48  (00:00)
zakhar   ssh:notty    211.157.109.37   Mon Dec 17 03:48 - 03:48  (00:00)
zakhar   ssh:notty    211.157.109.37   Mon Dec 17 03:48 - 03:48  (00:00)
yustina  ssh:notty    211.157.109.37   Mon Dec 17 03:48 - 03:48  (00:00)
yustina  ssh:notty    211.157.109.37   Mon Dec 17 03:48 - 03:48  (00:00)
yuriy    ssh:notty    211.157.109.37   Mon Dec 17 03:47 - 03:47  (00:00)
yuriy    ssh:notty    211.157.109.37   Mon Dec 17 03:47 - 03:47  (00:00)
yuri     ssh:notty    211.157.109.37   Mon Dec 17 03:47 - 03:47  (00:00)
yuri     ssh:notty    211.157.109.37   Mon Dec 17 03:47 - 03:47  (00:00)
yuri     ssh:notty    211.157.109.37   Mon Dec 17 03:47 - 03:47  (00:00)
yuri     ssh:notty    211.157.109.37   Mon Dec 17 03:47 - 03:47  (00:00)
```

```
yuliya   ssh:notty    211.157.109.37   Mon Dec 17 03:47 - 03:47  (00:00)
yuliya   ssh:notty    211.157.109.37   Mon Dec 17 03:47 - 03:47  (00:00)
yuliy    ssh:notty    211.157.109.37   Mon Dec 17 03:47 - 03:47  (00:00)
yuliy    ssh:notty    211.157.109.37   Mon Dec 17 03:47 - 03:47  (00:00)
yulianna ssh:notty    211.157.109.37   Mon Dec 17 03:47 - 03:47  (00:00)
yulianna ssh:notty    211.157.109.37   Mon Dec 17 03:47 - 03:47  (00:00)
yuliana  ssh:notty    211.157.109.37   Mon Dec 17 03:47 - 03:47  (00:00)
yuliana  ssh:notty    211.157.109.37   Mon Dec 17 03:47 - 03:47  (00:00)
yulian   ssh:notty    211.157.109.37   Mon Dec 17 03:47 - 03:47  (00:00)
yulian   ssh:notty    211.157.109.37   Mon Dec 17 03:47 - 03:47  (00:00)
```

## [B.3] Kernel Upgrade Procedure

```
[root@router ~]# yum check-update | grep -i kernel
kernel.i586                               2.6.22.14-72.fc6       updates
kernel-devel.i586                         2.6.22.14-72.fc6       updates
kernel-headers.i386                       2.6.22.14-72.fc6       updates

[root@router ~]# yum upgrade kernel
Loading "installonlyn" plugin
Setting up Upgrade Process
Setting up repositories
Reading repository metadata in from local files
Resolving Dependencies
--> Populating transaction set with selected packages. Please wait.
---> Downloading header for kernel to pack into transaction set.
kernel-2.6.22.14-72.fc6.i 100% |=========================| 204 kB    00:00
---> Package kernel.i586 0:2.6.22.14-72.fc6 set to be installed
--> Running transaction check
--> Processing Dependency: mkinitrd >= 5.1.19.0.3-1 for package: kernel
--> Restarting Dependency Resolution with new changes.
--> Populating transaction set with selected packages. Please wait.
---> Downloading header for mkinitrd to pack into transaction set.
mkinitrd-5.1.19.0.3-1.i38 100% |=========================|  52 kB    00:00
---> Package mkinitrd.i386 0:5.1.19.0.3-1 set to be updated
--> Running transaction check
--> Processing Dependency: nash = 5.1.19.0.3-1 for package: mkinitrd
--> Processing Dependency: libparted-1.8.so.2 for package: mkinitrd
--> Restarting Dependency Resolution with new changes.
--> Populating transaction set with selected packages. Please wait.
---> Downloading header for parted to pack into transaction set.
parted-1.8.2-2.fc6.i386.r 100% |=========================|  25 kB    00:00
---> Package parted.i386 0:1.8.2-2.fc6 set to be updated
---> Downloading header for nash to pack into transaction set.
nash-5.1.19.0.3-1.i386.rp 100% |=========================|  48 kB    00:00
---> Package nash.i386 0:5.1.19.0.3-1 set to be updated
--> Running transaction check

Dependencies Resolved

================================================================================
 Package                 Arch       Version          Repository        Size
================================================================================
Installing:
 kernel                  i586       2.6.22.14-72.fc6  updates           16 M
Updating for dependencies:
 mkinitrd                i386       5.1.19.0.3-1     updates          412 k
 nash                    i386       5.1.19.0.3-1     updates          1.1 M
 parted                  i386       1.8.2-2.fc6      updates          534 k

Transaction Summary
================================================================================
```

```
Install      1 Package(s)
Update       3 Package(s)
Remove       0 Package(s)

Total download size: 18 M
Is this ok [y/N]: y
```

## [B.4] Example Static ARP Configuration

```
[root@router ~]# arp -v -i eth0 -s 129.241.252.115 00:04:76:8F:A9:EE
arp: SIOCSARP()
[root@router ~]# arp -v -i eth1 -s 10.0.0.1 00:04:75:80:1B:EE
arp: SIOCSARP()
[root@router ~]# arp -v -i eth2 -s 192.168.0.1 00:50:8B:0B:C4:59
arp: SIOCSARP()
[root@router ~]# arp -v -i eth2 -s 192.168.0.3 00:08:74:2B:09:EF
arp: SIOCSARP()
[root@router ~]# arp -v -i eth2 -s 192.168.0.2 00:08:74:2B:0A:16
arp: SIOCSARP()
[root@router ~]# arp -e
Address                          HWtype   HWaddress                 Flags  Mask
Iface
192.168.0.2              ether   00:08:74:2B:0A:16   CM                    eth2
runit-gw.runit.no        ether   00:18:B9:A1:35:45   C                     eth0
router.blueteam.com      ether   00:50:8B:0B:C4:59   CM                    eth2
129.241.252.115          ether   00:04:76:8F:A9:EE   CM                    eth0
10.0.0.1                 ether   00:04:75:80:1B:EE   CM                    eth1
192.168.0.3              ether   00:08:74:2B:09:EF   CM                    eth2

[root@firewall-dmz-admin ~]# arp -v -i eth0 -s 192.168.0.2 00:08:74:2B:0A:16
arp: SIOCSARP()
[root@firewall-dmz-admin ~]# arp -v -i eth1 -s 192.168.1.1 00:01:03:2C:9E:4D
arp: SIOCSARP()
[root@firewall-dmz-admin ~]# arp -v -i eth0 -s 192.168.0.3 00:08:74:2B:09:EF
arp: SIOCSARP()
[root@firewall-dmz-admin ~]# arp -v -i eth0 -s 192.168.0.1 00:50:8B:0B:C4:59
arp: SIOCSARP()
[root@firewall-dmz-admin ~]# arp -e
Address                          HWtype   HWaddress                 Flags  Mask
Iface
192.168.0.1              ether   00:50:8B:0B:C4:59   CM                    eth0
firewall-dmz-admin.blue  ether   00:01:03:2C:9E:4D   CM                    eth1
192.168.0.2              ether   00:08:74:2B:0A:16   CM                    eth0
192.168.0.3              ether   00:08:74:2B:09:EF   CM                    eth0
```

## [B.5] SYN Flooding detected using netstat command

```
tcp        0      0 router.blueteam.com:ssh     ::ffff:129.241.252.11:mmcal FIN_WAIT2
tcp        0      0 router.blueteam.com:ssh     ::ffff:129.241.252.:ncr_ccl FIN_WAIT2
tcp        0      0 router.blueteam.com:ssh     ::ffff:129.:sun-as-iiops-ca FIN_WAIT2
tcp        0      0 router.blueteam.com:ssh     ::ffff:129.241.252.:rib-slm FIN_WAIT2
tcp        0      0 router.blueteam.com:ssh     ::ffff:129.241.252:taserver FIN_WAIT2
tcp        0      0 router.blueteam.com:ssh     ::ffff:129.2:vrts-auth-port FIN_WAIT2
tcp        0      0 router.blueteam.com:ssh     ::ffff:129.241.252.119:4064 FIN_WAIT2
tcp        0      0 router.blueteam.com:ssh     ::ffff:129.241.2:patrolview FIN_WAIT2
tcp        0      0 router.blueteam.com:ssh     ::ffff:129.241.25:rsc-robot FIN_WAIT2
tcp        0      0 router.blueteam.com:ssh     ::ffff:129.241.252.:sdsc-lm FIN_WAIT2
tcp        0      0 router.blueteam.com:ssh     ::ffff:129.241.252.:healthd FIN_WAIT2
tcp        0      0 router.blueteam.com:ssh     ::ffff:129.241.25:nmsigport FIN_WAIT2
tcp        0      0 router.blueteam.com:ssh     ::ffff:129.241.252:mosaixcc FIN_WAIT2
tcp        0      0 router.blueteam.com:ssh     ::ffff:129.2:mt-scaleserver FIN_WAIT2
tcp        0      0 router.blueteam.com:ssh     ::ffff:129.241.252.119:nfs  FIN_WAIT2
tcp        0      0 router.blueteam.com:ssh     ::ffff:129.:zfirm_shiprush3 FIN_WAIT2
```

**Figure 36 : Result from netstat command in case of SYN Flooding**