

## Module (Network Security)

Resit details: Resubmission of coursework (100%)

Submission: PDF document submitted to Web-CT or by email by 2 Sept 2005.

## Coursework Assignment

Module: Network Security

Title: Automated Threat Detection and Reconfiguration

### Introduction

Security is a complex issue, and it is not possible to ever completely secure a network. At present, most security systems rely on a human in detecting and thwarting incoming attacks. An improved method is to use network agents, such as an intrusion detection system, around a network and on hosts, which detects malicious types of activity. A simple model of this is shown in Figure 1, where an intrusion detection system, such as Snort, is used to detect possible threats. These alerts are then sent to a reconfiguration agent, which tries reprogram the firewall with the required rules to overcome the threat.

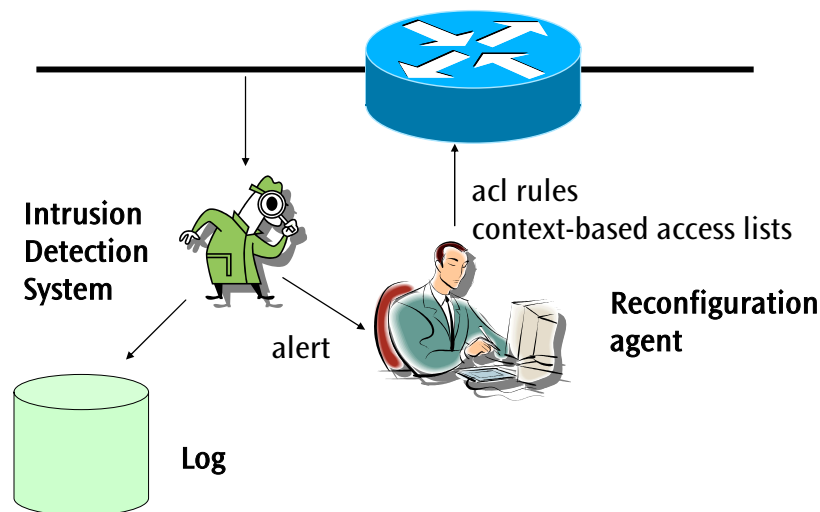


Figure 1: Automated reconfiguration

### Network Design

The host network topology is given in Figure 2. The basic setup is:

- The outside network is allowed access to the TELNET and WWW servers on the DMZ network, and access to a single TELNET server on the inside network (192.168.1.100).
- The inside network should be able to communicate with the outside network.

### Requirement

Design a prototype of a system which will achieve the following:

1. Detects an incoming connection to a specific host on the network on port 1234, and reprogram the firewall to block the originating address from the accessing the host.
2. Detects an incoming email with the word of "Spam" in the subject field, and logs it to a log file.
3. Detects an incoming ping on the network and reprograms the firewall to block the originating address from pinging the host.
4. Detects an incoming port sweep, and blocks the originating address from accessing anything on the network.
5. Detects a possible denial of service (DoS) on a WWW server, and reprograms the firewall to reduce the threat.
6. Detects a possible Distributed DoS on a WWW server, and reprograms the firewall to reduce the threat.
7. Detect the spread a known worm, of which the organisation only knows about the source code. The outline code is given in Appendix A.

The main requirement is to design the rules for the intrusion detection system, and also to create the rules for the Cisco firewall. For a basic implementation, there is no actual requirement for the source code, but it might help to illustrate the operation of the reconfiguration agent. The design section, though, should outline the actual design of the system.

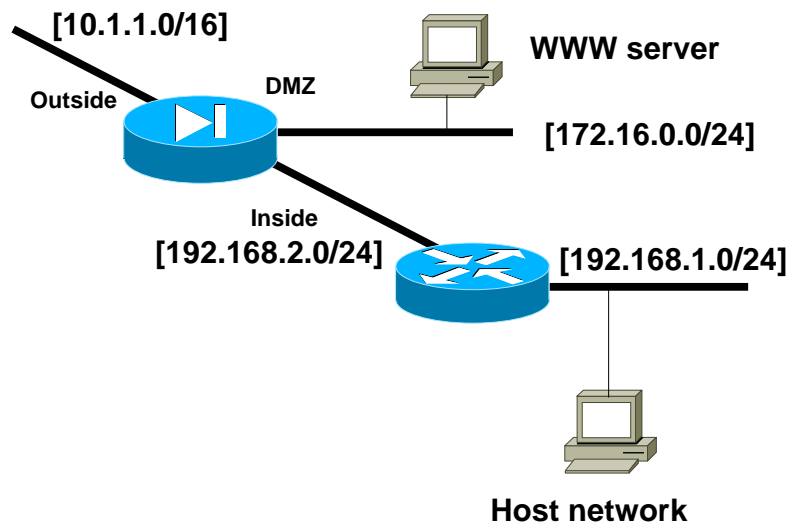


Figure 2: Network topology

## Marking schedule

The coursework should be submitted via Web-CT, in a PDF format, if possible. It will be marked as follows:

- **Introduction** [5%].
- **Theory/Research** [15%]. This should define the main background areas required for the report.
- **Design** [20%]. This shows the actual design of the system.

- **Implementation** [25%]. This shows the elements of the system which have been implemented, and which may show the actual results from the implementation.
- **Conclusions** [15%].
- **References/Presentations** [20%]. All references must be defined in a APA/Harvard format, and should be integrated in the report.

The report should use the APA/Harvard format for all of the references, and, if possible, should include EVERY reference to material sourced from other places.

## Appendix A

The **OhDear!** worm has not been seen yet on the Internet, but a small snippet of its source code have been released. It is known that it is a .NET application, and is written in C#. A snippet from the code is:

```
using System;
using System.Net;
using System.Net.Sockets;

namespace ConsoleApplication3
{
    class Class1
    {
        static void Main(string[] args)
        {
            System.Console.WriteLine("Hello... I'm the Worm");
            IPAddress ip = IPAddress.Parse ("10.11.12.13");
            int iPortNo = System.Convert.ToInt16 (9999);
            // Create the end point
            IPEndPoint ipEnd = new IPEndPoint (ip,iPortNo);
            Socket m_clientSocket = new Socket (AddressFamily.InterNetwork,
                SocketType.Stream, ProtocolType.Tcp );
            m_clientSocket.Connect(ipEnd);
            Console.ReadLine();
            // ... the main code goes here ...
        }
    }
}
```

The system should detect the application program in the payload of any network packet, and, if possible, monitor it when it is invoked.