Implementing Resource Revocation Utility for Network Operations Centre Management Console

O.S.Kayode and B.V.Ghita

Centre for Security, Communications and Network Research,
Plymouth University, Plymouth, UK
e-mail: info@cscan.org

Abstract

In an organization, there exist the possibility of users utilizing prohibited applications or resources. Preventing such activity deemed inappropriate therefore paramount in ensuring effective network administration. This research employed a simulated experimental network topology to develop a utility for terminating the connection of host engaged in activity deemed inimical to network performance. The utility automates network traffic capture; analyses captured traffic to identify prohibited connections (e.g. Peer-to-Peer traffic) and injects TCP reset packets to terminate connections. The research established that instantaneous termination of undesirable connections is effective; even though it could not always achievable due to packet delays, network congestion, false positives and non-identification of some application traffic.

Keywords

Network Operation Center (NOC), Packet Injection, TCP Reset, Automation Scripts, Connection Termination.

1 Introduction

In any enterprise network, it a fallacy to assume that all internal users are well-behaved and aboveboard with regard to their usage of network resources and application. Hence, there exist the possibility that an internal users can either deliberately or inadvertently utilize prohibited application or resources not permitted due to the adverse effect they have on the network. In the same vein, such user may also exceed their allotted network access rights and/or privileges while using authorized network resources.

Adequate network monitoring via a Network Operation Centre (NOC) management console brings such network breaches to the attention of the administrator, who must immediately take necessary action. More often than not, administrators utilize separate network tools to intervene and restore normalcy to the network.

Similarly, the clandestine usage of Peer-to-Peer (P2P) application that is hoarding precious network bandwidth within an enterprise might require a more urgent intervention by the administrator, especially if overtures or attempts to contact the perpetrator fails or is ignored.

These and other such instances highlight the need to incorporate functionalities within NOC management consoles so as to empower administrators with capability to revoke, streamline, limit and enforce network resources utilization in accordance with an organization's network and security policy.

This research aims to develop and incorporate network intervention functionality into a NOC management console, which enables network administrator to perform resource revocation. The utility terminates P2P communication and revoke resources hoarded by greedy applications/host.

The uniqueness and highlight of the developed utility is that it combines several tasks, which were hitherto undertaken with separate tools. It automates the entire process of capturing network traffic; analyses captured traffic to identify prohibited connections (e.g. Peer-to-Peer traffic) and injects TCP reset packets to terminate connections, all with minimal input from the network administrator. Commercial implication of such characteristics makes it a cost-effective utility.

2 Previous Work

There have been various efforts aimed at expanding the functionalities provided by NOC management consoles in a bid to make them more efficient.

In communication systems, Harry et al. (2001) produced a NOC system capable of testing two-way paging device to determine conformity to specified protocol. In one embodiment, the system comprises a transmitter, a receiver and a protocol engine. The protocol engine sends information to and receives information from a two-way communication device for compliance with multiple communication protocol

An inherent function of NOC is to provide security monitoring of network infrastructures. Behaviour profiling is being adopted for network security monitoring to automatically discover suspicious or malicious tendencies from network traffic. Such profiling also provides plausible interpretation of these behaviours to aid network operators in understanding anomalous events within network traffic (Kuai Xu et al. 2008).

Bali (2005) employed Netmates – an open source packet sniffing software, to fuse a network monitor and an Intrusion Detection System into a single GUI. Integration with Snort backend enabled the native packet sniffing functionality of Netmates to be extended by using smart add-ons and plug-ins to enable it serves as a form Network management systems.

The flexibility provided by web interface was harnessed by Mohyuddin (2006) in producing a NOC management console that provides administrators with a single unified interface for displaying network entities. Such approach enabled remote monitoring through standard web site using HTTP protocol. Though independent, the NOC system has functional dependency on some existing Linux features and some freely available network monitoring tools such as Nmap, Tcpdump and Ethereal.

The combination of two tools - PolyMon Network Monitor and NetMaCon, was used by Agbai (2007) to monitor and generate alerts. The PolyMon monitors network devices and generate alerts when the devices fail. The alerts are stored in a database, which NetMaCon accesses to display the visual and geographical location of network problems. Response to alert was limited to showing the physical location of the faulty device on the network.

Researches on P2P have been varied. While some ISPs, for example Packeteer (2009), are known to rate-limit the bandwidth consumed by P2P traffic by deploying traffic shapers in their network, others go further and ban them completely by injecting forged RST packets. (Schoen, 2007).

Other P2P studies have focused on topological characteristics of P2P networks based on flow level analysis (Sen and Wang, 2002), or investigating properties such as bottleneck bandwidths (Saroiu et al., 2002), the possibility of caching (Leibowitz et al., 2002), or the availability and retrieval of content (Bhagwan et al., 2003; Gkantsidis et al., 2003).

Sen et al. (2004) developed a signature-based payload methodology to identify P2P traffic. The authors focus on TCP signatures that characterize file downloads in five P2P protocols based on the examination of user payload.

3 Research Methodology

This section presents an overview of the methodology employed in conducting the research. In summary, this research methodology consisted of the following major tasks:

1. Setting up the network topology for the conduct of experiments.
2. Identifying undesirable activity on the network, through
   - Capture and analysis of traces.
   - Real-time monitoring of network activity.
3. Identifying host engaged in undesirable activity.
4. Manual termination of offending hosts' network connectivity.
5. Automation of tasks 2-4 above using shell script, and subsequent deployment of the script within a Network Operation Center (NOC) management console.

Details of each tasks summarized above is discussed in following sections.

3.1 Experimental Network Topology

The listing of configuration of each host is shown in Table 1. The computer network ***logical*** topology used for the research is depicted in Figure 1.

	NOC PC	CLIENT PC	SERVER
Operating System	Ubuntu Linux Version:10.4	Windows XP Ubuntu Linux	Windows 2003 Server Microsoft IIS
Other Software	EtherApe		Apache Tomcat
	Wireshark	Wireshark	Wireshark
	Tcpdump/Tshark	Tcpdump/Tshark	Tcpdump/Tshark
	PackeTH	BitTorrent Client	
	Snort	Telnet Client	
	Oracle VirtualBox	Oracle Virtual Box	
	Zenity	OmniPeek	
	OmniPeek		

Table 1: Table of Host Configurations

NOC-PC is a Unix-based machine serving as the NOC management console. It is strategically located at an interconnection point/junction, such that all traffic traversing the network can be easily monitored. Typical network monitoring software running on the NOC-PC included:

- Intrusion detection system software such as Snort – both of which are open-source;
- Protocal Analzers such as Tcpdump/tshark or Wireshark, which is configured to run in promiscuous mode in order to capture all network traffic;
- Real-time network monitoring utility such as EtherApe - running to provide a visual graphical display of network traffic pattern.
- PackeTH – Packet injection software via Graphical User Interface (GUI)

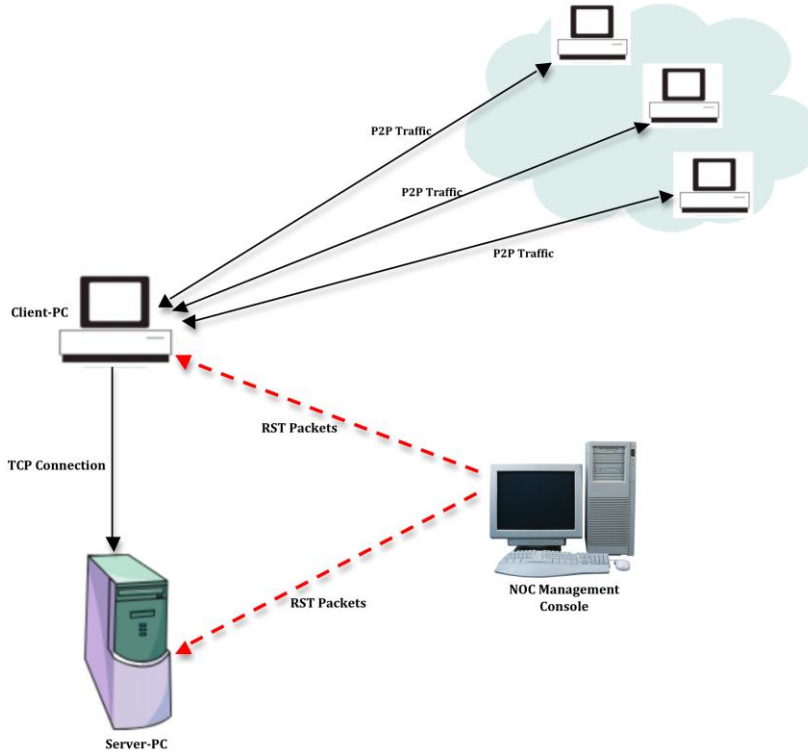

Figure 1: Network topology diagram

After observing network traffic pattern, a baseline of what is considered 'normal' was established. Therefore, any deviation from the established baseline is considered an anomalous situation that warrants further investigation.

Client_PC1 is a Unix PC that is used to carry out undesirable network activity. It is located within the same subnet as the NOC_PC and used to initiate:

- Peer-to-Peer (P2P) connection that has adverse effect on network bandwidth utilization.
- Unauthorized connection to the Server-PC via telnet session.

Server_PC is a Windows server machine is assumed to contain classified information. Two server software (Microsoft IIS on WINDOW 2003 Server and Apache Tomcat) were installed on the machine in order to compare any differences in response to CLIENT-PC connection termination.

3.2 Measurement

The traces generated on NOC_PC was filtered to include the source IP address, destination IP address, protocol, source port and destination ports.

Tcpdump/thark was employed for network measurements. It allows collection of TCP/IP packet headers, and (optionally) packet payloads as well.

In order to create a diverse context within which the experiment was carried out and to replicate the type of traffic found in a typical network, some random traffic were generated from these machine to mimic those obtainable in an enterprise environment. Wildpacket's OmniPeek traffic generator was used for this purpose.

Client_PC1 was used to initiate HTTP connection from a browser to a P2P network for downloading and uploading large file contents. In an attempt to study the TCP connection behaviour, a telnet connection was made to port 80 of servers software running on SERVER-PC and an HTTP request issued. While the connection was kept opened, it was possible to observe from tcpdump/thsark how or if the server closes the connection.

3.3 Injecting RST Packets to Abort Internal and External Connection

While there is an established connection from the Client_PC to the Server_PC, reset (RST) packets were *manually* generated on NOC-PC using PackeTH software. These RST packets were targeted towards:

- Client_PC using spoofed source IP address belonging to SERVER-PC.
- SERVER-PC using spoofed source IP address belonging to Client_PC.
- Both Client_PC and SERVER-PC.

While there is an active P2P connection from the Client_PC to an external P2P network and a download and/or upload process is taking place, reset (RST) packets were ***manually*** generated on NOC-PC using PackeTH software. These RST packets were targeted towards:

- Client_PC using spoofed source IP address.

3.4 Automation of Connection Termination Process

The manual termination process entails using separate software tools and some time interval inevitably elapses before each this software is launched, properly configured and executed. Moreover, the output generated by traffic capture software (e.g. tcpdump) needs to be analyzed to obtain pertinent information before it can be fed as input into another software tool (e.g. PackeTH).

In order to automate the connection termination process so that it can be integrated within the NOC management console, a shell script was developed. Figure 2 shows the flow chart of the logic of the script.

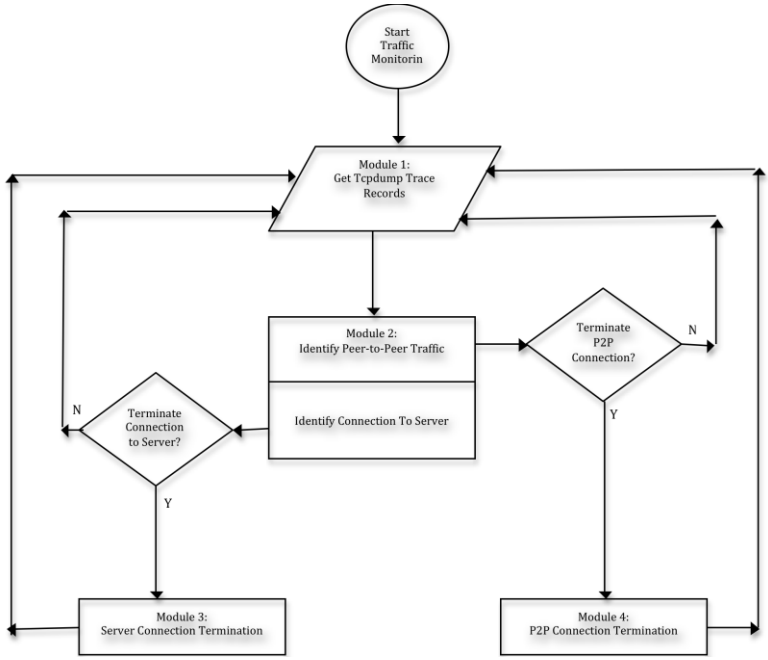

Figure 2: Flowchart for Automation Script

Module One involves the real-time monitoring and capture of network traffic using tshark/tcpdump. Due to memory space constraint Tshark/tcpdump was configured to capture only source and destination IP addresses, source and destination ports. The protocols captured are TCP, UDP, HTTP/HTTPS and DNS.

Module Two processes the captured traffic to identify presence of P2P traffic by comparing the various ports utilized by network protocols against a list of well-known ports normally utilized by P2P traffic. Further analysis of the network traces revealed camouflaged P2P traffic patterns that did not use any of the well-known ports in a bid to avoid detection.

Module Two also identifies any active connection with the SERVER-PC. This is achieved by highlighting connection to Server-PC by searching for the server's IP address as a destination of a traffic flow. The source IP address of such flow is sent then to Module-Four.

The IP address of the host engaged in P2P activity is forwarded to Module-Four for further action. Such connection is then maintained or marked for termination in Module-Four, depending on the network administrator's decision regarding the validity of such session.

Similarly, for any unauthorized connection to SERVER-PC, MODULE-Three performs the actual termination of the connection. The module utilizes an open-source packet injection utility, Packit, to launch the termination process via the command-line.

4 Results and Analysis

The EtherApe software providing real-time monitoring of the test network provided early indication of abnormal traffic flow. As shown in Figure 3, whenever P2P traffic was initiated on the CLIENT-PC, EtherApe displayed corresponding bursts/surge in traffic pattern, thus indicating instances of excessive bandwidth usage.

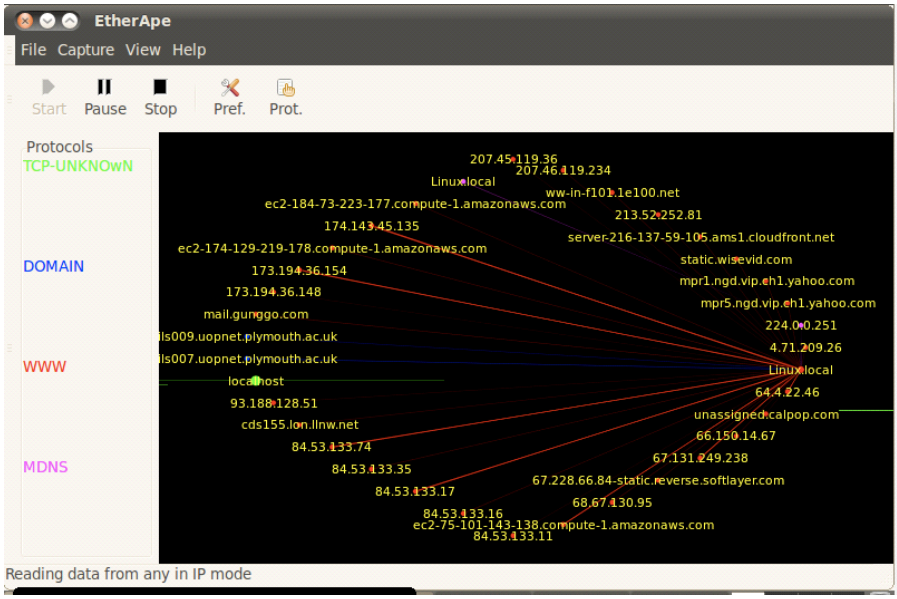

Figure 3: EtherApe Screen showing Peer-to-Peer Traffic.

The open-source packet injectors used for connection termination in this research suffers from lag conditions**.** This refers to the time interval from when RSTs packets are crafted and injected till the time endpoints receive the RST packet. Two separate lag conditions were observed during the course of the experiments for the research.

One lag condition occurred between the time when the administrator decides that a data packet/flow exhibit criteria warranting connection termination, and the time when *Packit* actually sends out the RST packet.

During this interval, further P2P packets downloads and/or upload had taken place. Consequently, it was observed that the RST packet is “out of sequence” because the sequence number is less than that of the preceding data packet. This condition often caused the target to ignore the RST packets.

Since P2P packets were sent back-to-back, the lag condition is more prevalent when the RST packet injection is in use. In the absence of injection, however, such condition is not expected during normal TCP operation.

The other lag condition was observed when, as at the time the RST packets were injected, further packets were already en-route, or were sent shortly afterwards, since the injector could not stop the traffic originator quickly enough. In these cases, the receiver still receives further data packets from the sender ***after*** it has already received the RST.

This second lag condition was identified by observing data packets with larger sequence number than those found on a previously received RST packet. Under normal circumstance, such condition is rare during normal end-host communication.

While every effort was made to ensure accuracy in identifying connections meant for termination, especially since the experiments were conducted in a controlled environment, it must be stressed that such condition may not be guaranteed in a 'live' production network as obtained within an enterprise.

Consequently, deploying the utility within an enterprise environment is bound to result in some false positive alerts. The possibility exist for network connection of incorrect hosts to be identified as being engaged in undesirable activity, and thus be terminated. Similarly, it is also possible for benign traffic to be wrongly identified as P2P traffic and therefore be inadvertently terminated.

Such cases of false positives are not peculiar to this research. For example, commercially developed firewalls and intrusion detection systems (IDS) have also been found to raise false alerts. The underlining factor responsible for such false alerts is the accuracy of the algorithm or signature employed in indentifying hostile/undesirable activity.

The type of network topology within which the experiment was conducted significantly influenced the result of this research. As depicted in Figure 3.1, all the machines in the topology are in a single collision and broadcast domain, hence, possibility exist for Address Resolution Protocol (ARP) broadcast storm to occur. Consequently, injected reset (RST) packets may be delayed en route to their target due to congestions caused by ARP broadcast storms.

The effect of NAT on the connection termination utility can not be overlooked. In an enterprise environment with multiple branch offices/sites, terminating the connection of remote user engaged in clandestine or inappropriate network activity has to contend with NAT issues. Other likely issues also include traffic tunneling and encryption. Inasmuch as these issues were not factored into the experiments conducted, then, there is possibility that using the utility in such scenario may not be effective or functional.

By targeting injected RST packets at offending host, the result of the conducted experiments showed that it was possible to terminate and/or disrupt established TCP/UDP connections. When compared with isolated tests conducted by issuing the '*KILL'* command on a Unix/Linux system, significant differences were observed.

The '*KILL'* command is executed with the process ID number of target application supplied as argument. The command sends a special, high-priority SIGTERM signal,

which instantaneously terminates all main and child processes associated with the target application. All windows associated with the target application are instantaneously shutdown.

On the contrary, RST packets terminate only a session of the application using the targeted TCP flow. For example, the specific Internet browser window used for P2P activity or the window of the telnet client connected to SERVER-PC were the only sessions affected by RST packets. Atimes, these windows sessions simply hang-up and/or stops responding as a resultant effect of injected RST packets. Further checks were necessary to verify that the session/connection was indeed terminated. However, other instances or sessions of the browser or Telnet client remained unaffected.

5 Conclusions and Recommendations

The research established that instantaneous termination of undesirable connections is effective; even though it could not always achievable due to lag conditions, packet delays, network congestion, false positives and non-identification of some application traffic.

It is recognized that the capability being developed can be considered a double-edged sword. The utility can be used to disrupt legitimate user activities on the network if used by unauthorized mischievous persons. Consequently, access right to utilize the program is highly restricted to **ONLY** the network administrator with root privilege.

It is worth investigating the effect a switched network topology will have on the outcome of the experiments. Introducing a Layer-2 network switch into the experimental topology will ensure each host resides in its own collision domain, thereby isolating each hosts from collision within its switch port. In an enterprise setting, such topology will reduce network congestion, and *could possibly* ensure RST packets reach their intended destination faster. Virtual LANs (VLAN) can also be configured on the switch to enhance security and further prevent

The resource revocation utility was not only developed on a Unix/Linux system, but also aimed at NOC management console based on Unix/Linux Operating Systems. The wealth of powerful intrinsic commands offered by Unix/Linux systems informed this decision. Attempts to modify and/or develop similar functionality for a NOC management console hosted on other Operating System platforms can be undertaken as future research.

6 References

Agbai, O. C. (2007) ‘Implementing A Visual Network Management Console’, MSc Thesis, Plymouth University.

Bali, R. (2005) ‘Implementing Network Operation Center Management Console: NetMaCon’, MSc Thesis, Plymouth University.

Bhagwan, R., Savage, S. and Voelker, G. (2003) 'Understanding Availability', International Workshop on Peer-To-Peer Systems, IPTPS, Available at http://iptps03.cs.berkeley.edu/final-papers/availability.pdf [Accessed 5th August 2010]

Gkantsidis, C., Mihail, M. and Saberi, A. (2004) 'Random Walks in Peer-to-Peer Networks', Information Communication Conference, INFOCOMM, Available at http://www.ieee-infocom.org/2004/Papers/03_4.PDF [Accessed 15th July 2010]

Harry V. B., Donna B. (2001) 'Network Operations Center Hardware and Software Design', Patent No.: US 6,259,911 B1

Kuai X., Zhi-Li Z., and Supratik B. (2008), 'Internet Traffic Behavior Profiling for Network Security Monitoring' IEEE/ACM Transactions on Networking, (16)6: 1241-1252

Leibowitz, N., Bergman, A., Ben-Shaul, R. and Shavit A. (2002) 'Are File Swapping Networks Cacheable: Characterizing P2P Traffic', 7th International Workshop on Web Caching and Content Distribution, IWCW, 2002. Available at http://2002.iwcw.org/ [Accessed 15th July 2010]

Mohyuddin, A.(2006) 'Implementing a Network Operation Center Management Console', MSc Thesis, Plymouth University.

Packeteer (2009) http://www.packeteer.com [Accessed 15th July 2010]

Sen, S. and Wang, J. (2002) 'Analyzing Peer-to-Peer Traffic Across Large Networks', Proceedings of ACM SIGCOMM Internet Measurement Workshop, IMW, 2002. Available at http://conferences.sigcomm.org/imc/2002/imw2002-papers/167.ps.gz [Accessed 11th June 2010]

Sen, S., Spatscheck, O. and Wang, D. (2004) 'Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signatures' World Wide Web (WWW) Conference, Available at http://www.iw3c2.org/WWW2004/docs/1p512.pdf [Accessed 27th August 2010]

Schoen, S. (2007) 'Comcast and BitTorrent', Available at http://www.fcc.gov/eb/ Orders/2007/DA-07-4005A1.html [Accessed 15th July 2010