2014

Evaluating the security vulenerabilities of the IP6to4 tunnelling mechanism

Brian Cusack
Auckland University of Technology

Raymond Lutui
Auckland University of Technology

DOI: 10.4225/75/57b65dad343d1

12th Australian Information Security Management Conference. Held on the 1-3 December, 2014 at Edith Cowan University, Joondalup Campus, Perth, Western Australia.

EVALUATING THE SECURITY VULENERABILITIES OF THE IP6to4 TUNNELLING MECHANISM

Brian Cusack, Raymond Lutui
Auckland University of Technology, Auckland, New Zealand
brian.cusack@aut.ac.nz, riutui@aut.ac.nz

Abstract

The two versions of Internet Protocol (IP) rely on mechanisms that will convert one protocol to the other and vice versa. Version 4 is still prevalent in the Internet backbone and version 6 in most private networks. In this research we focus on the automatic tunnelling mechanism that provides the encapsulation at one end of the transition tunnel and the de-encapsulation at the other end dependant on the direction of transition. In our research we asked: How secure is the automatic tunnelling mechanism? It is a simple question but important given the number of times transition may occur in any communication and the potential for vulnerabilities. To test the capability of the software instance we launched attacks on the inside and the outside of the tunnel; recorded performance variations and noted opportunities for information sniffing. In all instances the results show weaknesses that can be exploited and the potential for an outsider to not only launch for example DoS attacks but to also disrupt the information being managed in the tunnel. How secure is the automatic tunnelling mechanism?

Keywords
IPv4, IPv6, Transition, Tunnelling, Mechanisms, Network, Security

INTRODUCTION

TCP/IP is the standard protocol suite that plays an important role in facilitating computer communication around the world. The Internet Protocol version 4 (IPv4) is the core and traditional standard. However IPv4 has a limited address capability and IPv6 was introduced principally to increase the number of addresses available for the massive growth in Internet and device usage (Bi, Wu & Leng, 2007). Consequently mechanisms had to be developed so that communications could proceed seamlessly in either protocol and between each. The Network Working Group through a series of Requests For Comment (RFCs) has developed the standards for bi-directional tunnelling so that reciprocity between the two versions is achieved. When the demands for IP addresses surpassed the conjectural limit of its current architecture about two decades ago, the Internet Engineering Task Force (IETF) in the Work Group started developing a solution for the shortage of IP addresses and came up with IPv6. It has major improvements such as its security features, 340 trillion, trillion, trillion unique IPv6 addresses, plug and play configurations, mobility support and Quality of Service (QoS) capabilities. Due to the recent growth and the complexity of the Internet, migrating from IPv4 to a pure IPv6 environment is going to be a long and very challenging process (Bi, Wu & Leng, 2007; Xin & Xi, 2010). To help during the transition period, the IETF developed several mechanisms so that IPv4 and IPv6 are able to co-exist (Narayan & Tauch, 2010a). There are three methods of transition mechanisms known as: dual stacks, tunnelling, and translation. Dual stacks works by maintaining the two protocol stacks. This allows them to operate in parallel. This can be implemented on both end systems and all network devices which allow them to carry both IPv4 and IPv6 packet types. The protocol translation mechanism on the other hand employs two different techniques known as *"stateful"* and *"stateless"*. In a stateful setup, both end systems and the network devices are used for the environment and the translator maintains a recording of the two types of IP addresses (Narayan & Tauch, 2010b). The Stateless technique on the other hand, the translator is able to process each packet individually without having to refer to previous translation records (Narayan & Tauch, 2010b). The third mechanism is knows as tunnelling. This technique works by allowing IPv6 traffic to travel over the existing IPv4 infrastructure (Julianne & Mile, 2013). In this setup, both the end systems and network devices will be the tunnel's endpoints in which they will perform the encapsulation or decapsulation process (Narayan & Tauch, 2010a; Guardini, Durand & Lento, 2001).

In this research our focus is on the automatic tunnelling mechanism. There are four categories of relationship in the field of study: Router to Router; Host to Router; Host to Host and Router to Host. There are also eight tunnelling mechanisms employed in networking and they are: IPv6 configured tunnel; Automatic tunnel with IPv4-compatible IPv6 address; 6over4 tunnel; 6to4 tunnel; ISATAP; DSTM; Tunnel broker and Toredo (Mun & Lee, 2005). The primary focus in this paper is the performance of the automatic tunnelling mechanism under attack as it carries most normal traffic loadings. Part of the performance appraisal is to assess the tunnel behaviour under attack and to locate security vulnerabilities in the 6to4 tunnelling mechanism. Our question is; How secure is the automatic tunnelling mechanism? The paper is organised to briefly review the related

literature, to design a research methodology, to report the findings and to discuss the implications for network security evaluation.

BACKGROUND LITERATURE

The set of transition mechanisms that allow IPv6 to coexist with IPv4 are described as a "transition tool box" (Xin & Xi, 2010). The two Internet protocols will have to co-exist for an undefined period of time and as a consequence different techniques and architectures are available. The two prominent solutions are dual stack and tunnelling (Narayan & Tauch, 2010a). It is also very important to manage this process effectively so as to maintain network services seamlessly. With regards to managing the co-existence of the two Internet Protocols, Hsieh & Shang-Juh (2005) suggested that, it is in two parts. First, in a co-existing environment, the transition method to be employed should be determined. Secondly, appropriate management facilities should be developed and applied. Management facilities for a co-existence environment will include topology discovery, detection of IP misusing and traffic monitoring. The dual stack approach requires the full support for both protocols whereas tunnelling is a cost effective way of encapsulating IPv6 packets to rout them over the IPv4 protocol and then in the reverse direction to de-encapsulate for routing over IPv4. Not surprisingly tunnelling is often the preferred approach and the opportunity to use the solution is built into most proprietary software.

IPv6/IPv4 hosts and routers can tunnel IPv6 datagrams over regions of the IPv4 routing topology by encapsulating them within IPv4 packets in a variety of ways (Cowley, 2007):

> Router-to-Router. IPv6/IPv4 routers interconnected by an IPv4 infrastructure can tunnel IPv6 packets between themselves. In this case, the tunnel spans one segment of the end-to-end path that the IPv6 packet takes.
>
> Host-to-Router. IPv6/IPv4 hosts can tunnel IPv6 packets to an intermediary IPv6/IPv4 router that is reachable via an IPv4 infrastructure. This type of tunnel spans the first segment of the packet's end-to-end path.
>
> Host-to-Host. IPv6/IPv4 hosts that are interconnected by an IPv4 infrastructure can tunnel IPv6 packets between themselves. In this case, the tunnel spans the entire end-to-end path that the packet takes.
>
> Router-to-Host. IPv6/IPv4 routers can tunnel IPv6 packets to their final destination IPv6/IPv4 host. This tunnel spans only the last segment of the end-to-end path.

The underlying mechanisms for tunnelling are (Cowley, 2007):

> The entry node of the tunnel (the encapsulator) creates an encapsulating IPv4 header and transmits the encapsulated packet.
>
> The exit node of the tunnel (the decapsulator) receives the encapsulated packet, reassembles the packet if needed, removes the IPv4 header, and processes the received IPv6 packet.
>
> The encapsulator may need to maintain soft-state information for each tunnel recording such parameters as the MTU of the tunnel in order to process IPv6 packets forwarded into the tunnel.

Other matters such as managing the header sizes (fragmentation) and so on have to be compensated by the tunnelling mechanism (Cowley, 2013a). Security management on the other hand, has a special set of issues relating to tunnelling (Cowley, 2013b). Source address spoofing for example is a vulnerability for automatic tunnelling when the ingress filters are circumvented. In this case, without specific ingress filtering checks in the decapsulator it would be possible for an attacker to inject a packet with:

- Outer IPv4 source: real IPv4 address of attacker
- Outer IPv4 destination: IPv4 address of decapsulator
- Inner IPv6 source: Alice, which is either the decapsulator or a node close to it
- Inner IPv6 destination: End user

Even if all IPv4 routers between the attacker and the decapsulator implement IPv4 ingress filtering, and all IPv6 routers between the decapsulator and the end user implement IPv6 ingress filtering, the spoofed packets will not be filtered out. As a result, the end user will receive a packet that looks like it was sent from a legitimate source even though the sender is unknown. The solution is to have the decapsulator accept only encapsulated packets

from the explicitly configured source address at the other end of the tunnel. While this does not provide a complete protection in the case ingress filtering has not been deployed, it does provide a significant increase in security. The threat model for a tunnel is different than another part of a network because of the simplicity in design and the emphasis on communication effectiveness (Cowley, 2007).

The primary aim of this paper is to evaluate the security vulnerabilities of the 6to4 tunnelling mechanism. There have been numerous studies done on the performance side of the network management model regarding the tunnelling mechanisms. As a result, this paper concerns the security issues in one important network mechanism. In the RFC 4942, Savola, Krishnan & Davies (2007) recommended that, extra care should be taken when deploying automatic tunnelling mechanism such as 6to4. The concern here is that there is no pre-configured relationship between end points. As a result, the receiving end will have to accept and decapsulate all packets. In the RFC 3964, Savola and Patel (2004) also added that, such mechanism's characteristic is vulnerable to threats such as Denial of Service (DoS) and IP Spoofing. The 6to4 router has no way of knowing whether the receiving traffic are from legitimate 6to4 relays or not. Tunnels are more vulnerable to a breach than physical links, as an attacker anywhere in the Internet can send an IPv6-in-IPv4 packet to the tunnel decapsulator, causing injection of an encapsulted IPv6 packet to the configured tunnel interface unless the decapsulation checks are able to discard packets injected in some manner. There are four major concerns with 6to4 mechanism argued in (ITPSS, 2008) and they are:

- There is no way for the 6to4 routers to identify the receiving packets to verify whether they are from legitimate relays.
- Potentially incomplete or wrongly configured 6to4 router or relay security checks.
- 6to4 architecture used to participate in DoS or reflected DoS, makes another attack harder to trace.
- 6to4 relays being subject to administrative abuse.

ITPSS (2008) also added that, even in an environment where 6to4 is properly implemented, it is still vulnerable to threats such as DoS attacks, Reflection DoS attacks and also Service Theft. This is when a malicious node, site or an operator uses the service without authorisation (Gelogo & Lee, 2011). These issues are not to be taken lightly and some reports such as Bilski (2011) predicted that the transition phase from IPv4 to IPv6 will continue for decades. Gallaher and Rowe (2005) from the National Institute of Standards & Technology (NIST) support the extended implementation time frame by citing the estimated costs inhibitors.

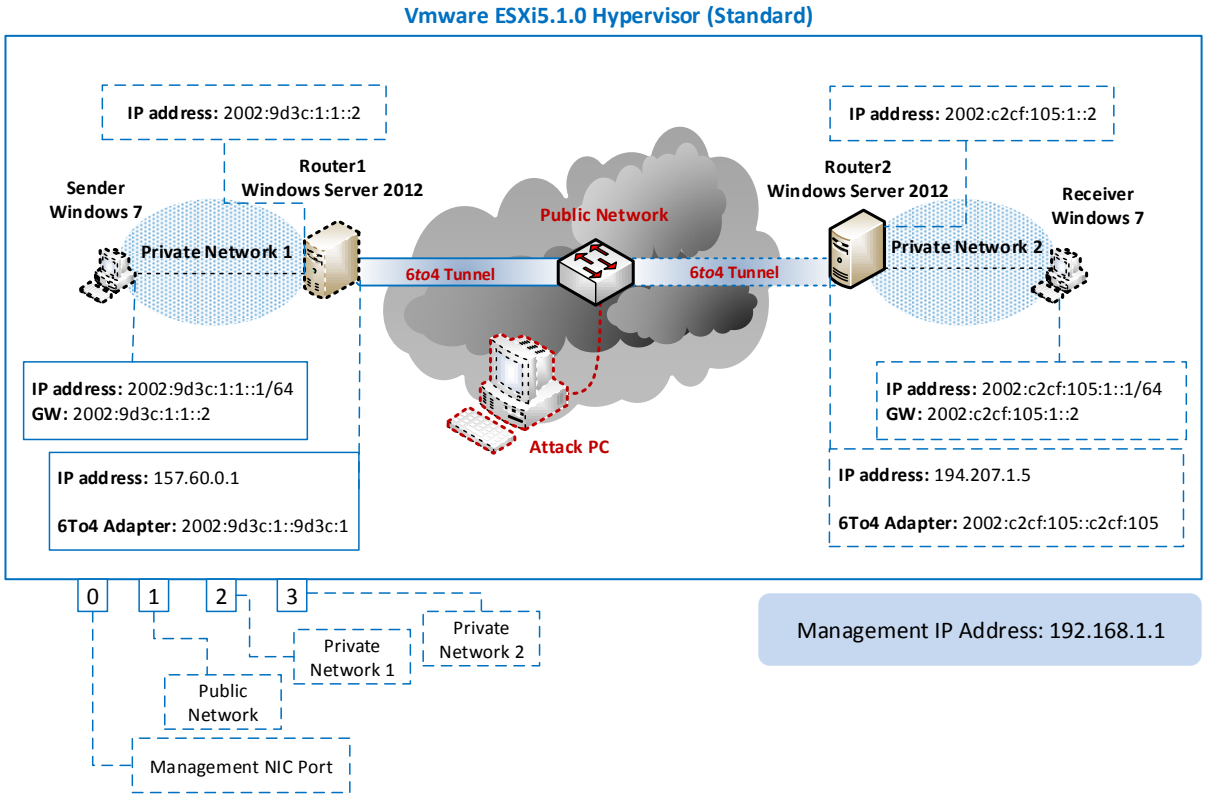

Figure 2: Test bed network diagram.

METHOD

To analyse the vulnerabilities of the 6to4 tunnelling mechanism, a test bed was setup as shown in Figure 2. We also used software tools such as the Distributed Internet Traffic Generator (D-ITG) version 2.8.1-r1023 for Windows to generate the traffic around the test network. Wireshark 64bit version 1.10.3 was used to sniff the network for information vulnerabilities around the 6to4 tunnel.

The test bed network diagram showed two servers running on Windows Server 2008 r2. These servers were configured as routers, also the host of the 6to4 mechanisms. Figure 2 also showed three client PCs, one being the sender, the other was the receiver while the third one was used as the attacker's PC. In our setup, the two Local Area Network (LAN) environments were pure IPv6 networks while the public network was an IPv4 network. In order for us to analyse the vulnerabilities of the 6to4 tunnel and also to verify our theory when introducing some attacks, the raw performance of the network was tested as a benchmark to measure attack variations from (Narayan, et al., 2010). In order to capture the raw performance of the network without further improvements, the operating systems involved were installed without further configurations (Hadiya, Save & Geetu, 2013). The *Private1* switch was configured as a Fast Ethernet switch in order to throttle the speed to 100Mb/s and the Maximum Transmission Unit (MTU) was set to 1500 Bytes (Narayan & Tauch, 2010c). In the experiments, eleven packet sizes were used ranging from 64 Bytes to 1518 Bytes. Each packet size was run 50 times and 180000 packets were generated every second for 60 seconds on each run. Every experiment was conducted on both the two transmission protocols and that is Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). A batch script was used to simplify and minimise the configuration time of the 6to4 tunnel on both routers and are shown in figures 3 and 4.

```
# Router 1
# Set packet forwarding
        netsh int ipv6 set int "Private" forwarding=enabled
# Enable 6to4
        netsh int ipv6 set int "Private" forwarding=enabled
# Set packet forwarding on 6to4 endpoint interface
        netsh int ipv6 6to4 set state enabled
# Add IPv6 address to the Private network interface
        netsh int ipv6 set int "6to4 Tunnelling Pseudo-Interface" forwarding=enabled
        netsh int ipv6 add address "Private" 2002:9d3c:101:1::2
# Configure static routing
        netsh int ipv6 add route 2002:9d3c:101:1::/64 "Private"
        netsh int ipv6 set int "Private" forwarding=enabled advertise=enabled
        netsh int ipv6 add route 2002:9d3c:1:472a::/64 "Private"
```

Figure 3. Batch script for Router 1

```
# 6to4 Router2
# Set packet forwarding
        netsh int ipv6 set int "Private" forwarding=enabled
# Enable 6to4
        netsh int ipv6 6to4 set state enabled
# Set packet forwarding on 6to4 endpoint interface
        netsh int ipv6 set int "6to4 Tunnelling Pseudo-Interface" forwarding=enabled
# Add IPv6 address to the Private network interface
        netsh int ipv6 add address "Private" 2002:c2cf:105:1::2
# Configure static routing
        netsh int ipv6 add route 2002:c2cf:105:1::/64 "Private"
        netsh int ipv6 set int "Private" forwarding=enabled advertise=enabled
        netsh int ipv6 add route 2002:c2cf:101:472a::/64 "Private"
```

Figure 4. Batch script for Router 2:

The D-ITG tools were used on the Sender PC to generate the packets for every packet size. To automate the 50 runs, a script was written with the following commands.

-a 2002:c2cf:105:1::1 -rp 10001 -C 180000 -c 64 -t 60000 -T TCP

-a option defines the destination address, **-rp** defines the port number to receive the traffic from that particular run. **–C** defines the number of packets to send every second, **-c** defines the size of the packet to test, **-t** defines the duration of the run and **–T** defines the protocol being tested. The D-ITG tool has to be run on both the *Sender* and the *Receiver* for it to work. Following command was used to start generating traffic. On the Receiver – **ITGRecv** and **ITGLog** were both started. On the Sender, the following command was used.

ITGSend <name of the script> –x <name of the recv log file>

After each run, the **ITGDec** tool was used to decode the Receiver's log files. On the decoded files, six metrics were taken for analysis and they are; Throughput, Maximum Delay, Jitter, Total time of each run, Total Bytes received and Packet Loss rate on User Datagram Protocol (UDP).

To capture the behaviour of the abnormal traffics flow rates, the **Attack** machine attaching to the **Public Network** switch was used to introduce a Denial of Service (DoS) attack one of the 6to4 routers while the normal traffics is sent from the Sender PC to the Receiver. The **Ping of Death** (PoD) commands was used to introduce a Denial of Service (DoS) attack from the public network which is the IPv4 network. After that, the **Attack** machine was also attached to the **Private1** switch and introduces the same DoS attack but this time was directly inside the 6to4 tunnel which is the IPv6 network. The **PoD** command used:

IPv4 – Ping 157.60.0.1 –l 65500 –n 10000000 -t

IPv6 – Ping 2002:c2cf:105::c2cf:105 –l 14776 –n 10000000 –t

In addition Wireshark was used to monitor spoofing and related attack varations.

FINDINGS

The test-bed analysis was divided into two parts, firstly the use of the packet analyser (Wireshark) software tool to capture information from the network especially information on the 6to4 tunnel. Secondly, we measured how the network performs when there is a DoS attack on the tunnel from both inside and out. The DoS attack was introduced from the public network; this was achieved by attacking the IPv4 address. The attack was introduced from inside the private network directly on the IPv6 tunnel address. The following sections shows the results of our two test-bed findings.

The Packet Analyser Results

Network Packet Analyser – Wireshark 1.11, this software was employed to sniff and capture network packets then analyse them to determine the vulnerabilities of the 6to4 tunnelling mechanism. The information captured by the packet analyser is shown in *Figure 5*.

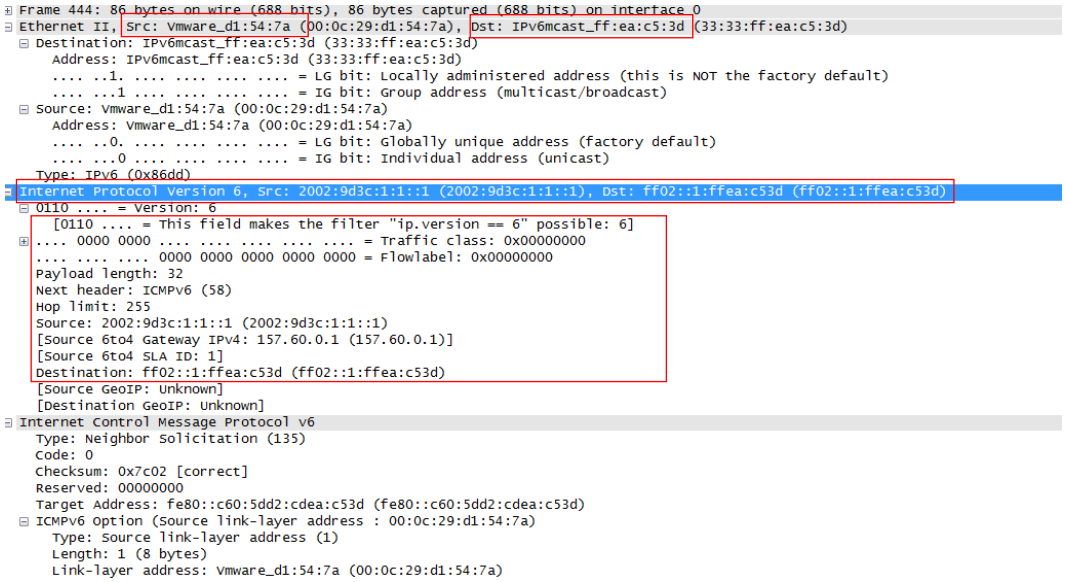

Figure 5. Packet analyser report when running from the public network.

An attacker can easily tell that this particular server is on a virtual environment, VMware in particular. Under the Internet protocol information, even sniffing from the public network, it can be seen that the network employed the Internet protocol version 6 (IPv6). Source IP address for all traffic inside the Private network can be seen as well. However, the destination address captured was the multicast address only not the actual destination address of the Receiver. All of the information can be readily accessed and exploited for spoofing and man-in-the-middle attacks. Not only that but the packet analyser also captured information such as the payload length, the hop limits and it can also see that this server is a host for a particular 6to4 tunnelling mechanism disclosing a vector for specific attacks. Wireshark also captured the IPv4 address that hosts the 6to4 tunnel on the gateway out from the private network that can be a vector for any external attacker. The only noted defence was that the packet analyser when running from the public network was not able to capture the actual tunnel address.

The 6to4 mechanism involved in this research is the automatic tunnel. This is called an automatic tunnel because the user does not configure the tunnel address IPv4 because the end point will be encapsulated in the IPv6 destination address. The main concern with the automatic tunnel address mechanism is that it can be guessed (Krishnan, Thaler & Hoagland, 2011). This is feasible because in the IPv6 address range, the 2002::/16 range has been reserved for tunnelling addresses. An attacker can know the first /16 prefix, next task is to create the /48 prefix but this is easily converted from the IPv4 address that the packet analyser gives. The format is showed in Figure 6 below.

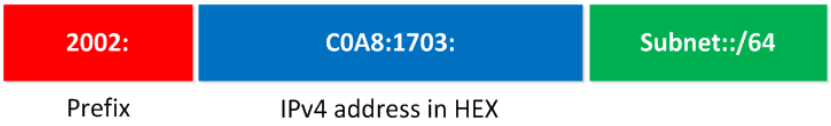

Figure 6. Format of 6to4 tunnel address. (Molenaar, 2013)

There are tools available on the Internet to convert IPv4 addresses to its hexadecimal values however, to manually calculate it, the attacker first needs to convert the IPv4 addresses to binary. The IPv4 address used in this study on the first 6to4 host is 157.60.0.1.

The result shown in Figure 7 was captured when the packet analyser was run inside the source's private network.

```
Frame 444: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface 0
    Interface id: 0
    Encapsulation type: Ethernet (1)
    Arrival Time: Jan 21, 2014 18:56:14.511996000 New Zealand Daylight Time
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1390283774.511996000 seconds
    [Time delta from previous captured frame: 4.693499000 seconds]
    [Time delta from previous displayed frame: 4.693499000 seconds]
    [Time since reference or first frame: 2108.417319000 seconds]
    Frame Number: 444
    Frame Length: 86 bytes (688 bits)
    Capture Length: 86 bytes (688 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ipv6:icmpv6]
    [Coloring Rule Name: ICMP]
    [Coloring Rule String: icmp || icmpv6]
Ethernet II, Src: Vmware_d1:54:7a (00:0c:29:d1:54:7a), Dst: IPv6mcast_ff:ea:c5:3d (33:33:ff:ea:c5:3d)
  Destination: IPv6mcast_ff:ea:c5:3d (33:33:ff:ea:c5:3d)
      Address: IPv6mcast_ff:ea:c5:3d (33:33:ff:ea:c5:3d)
      .... ..1. .... .... .... .... = LG bit: Locally administered address (this is NOT the factory default)
      .... ...1 .... .... .... .... = IG bit: Group address (multicast/broadcast)
  Source: Vmware_d1:54:7a (00:0c:29:d1:54:7a)
      Address: Vmware_d1:54:7a (00:0c:29:d1:54:7a)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv6 (0x86dd)
Internet Protocol Version 6, Src: 2002:9d3c:1:1::1 (2002:9d3c:1:1::1), Dst: ff02::1:ffea:c53d (ff02::1:ffea:c53d)
  0110 .... = Version: 6
      [0110 .... = This field makes the filter "ip.version == 6" possible: 6]
  .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
      .... 0000 00.. .... .... .... .... .... = Differentiated Services Field: Default (0x00000000)
      .... .... ..0. .... .... .... .... .... = ECN-Capable Transport (ECT): Not set
      .... .... ...0 .... .... .... .... .... = ECN-CE: Not set
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 32
```

Figure 7. Packet analyser report when running from the private network.

The packet analyser can extract more information about the private network such as Ethernet port number that the hypervisor used for managing the servers. An attacker can also tell the country that this network is in, the date and time of the activities. The packet analyser still cannot capture the actual IP address of the 6to4 tunnel.

However, for a skilled hacker it really does not matter since the tunnel address can be calculate once you can see the IPv4 address of the host. Krishnan, et al., (2011) also noted that, in some tunnelling protocols it is also possible to guess the tunnel address once an arbitrary client IP address is discovered. For instance, protocols such as 6to4 or Toredo, they are using a well-defined address ranges. This is because of the fact that these addresses are structured which allows an attacker to be able to algorithmically derive the address from known values. As a result, it reduces the search space for an attacker. It is recommended that when implementing tunnel address, a random value from an unused range on the endpoint will reduce the address scanning risks.

The DoS Test Results

Following in *Figure 8* shows the results of how the network performs under *Normal* traffic flows and also how the network performs when the DoS attacks were introduce on the IPv4 address (65500 attack) and also on the IPv6 address (inside attack).

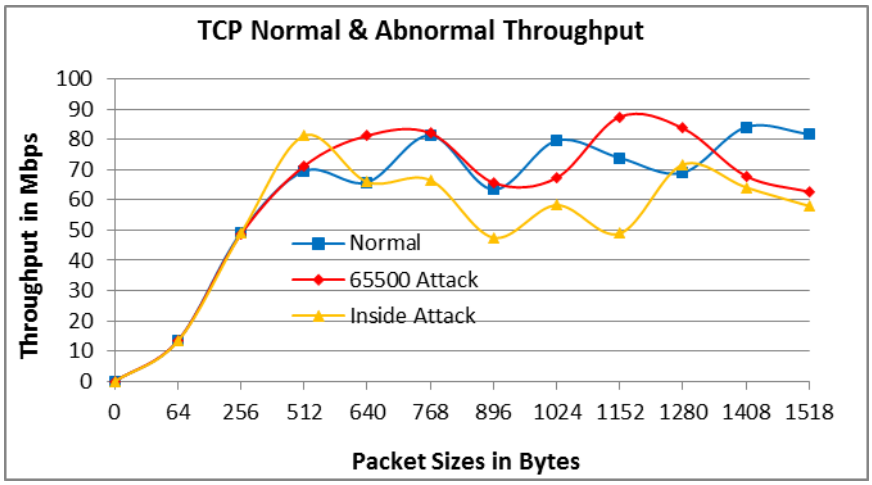

Figure 8. TCP throughput of Normal and Abnormal data.

Following in *Figure 9* shows the amount of delay in the network under normal traffic flows and how it compares to when the network is under attacks.

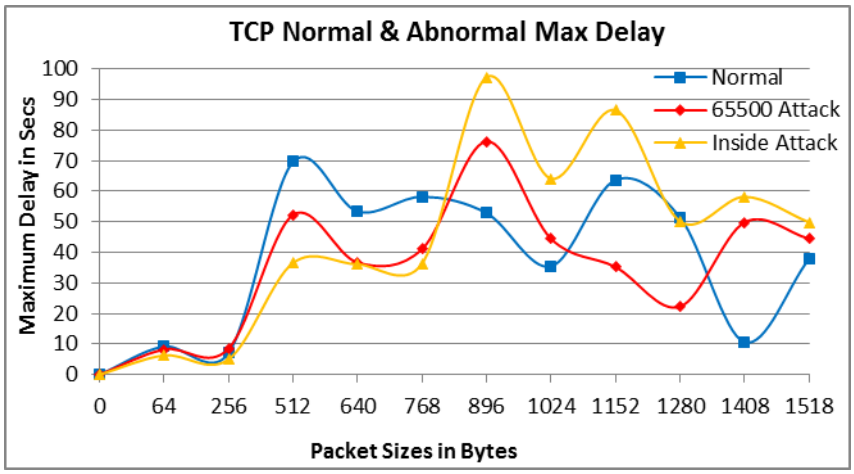

Figure 9. TCP maximum delay of Normal and Abnormal data.

Following in *Figure 10* shows the Total time results. This is the time taken for one test to complete. In this test, a 60 seconds time limit was defined for each test. This was done so that we can see the difference under normal traffic flows and when the DoS attacks were introduced.

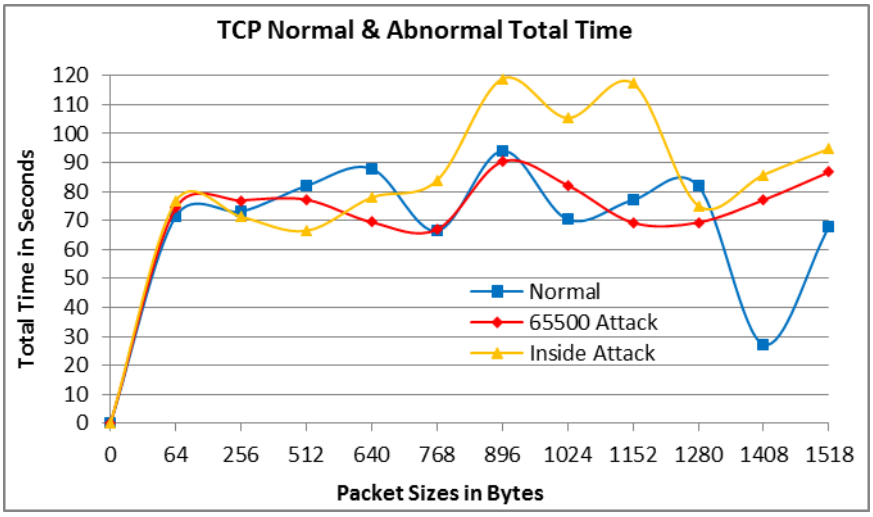

Figure 10. TCP total time of Normal and Abnormal data.

The *Figure 10* shows the results of the data gathered from our tests based on the total Bytes received on the receivers end.

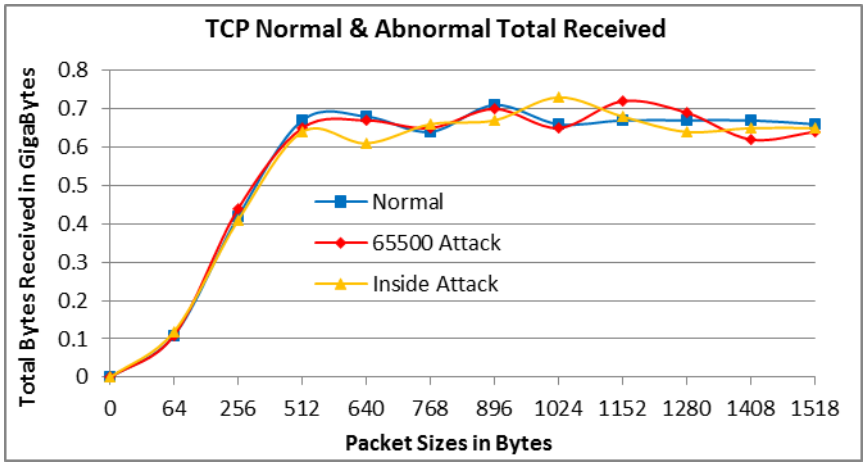

Figure 11. TCP total bytes received data of Normal and Abnormal.

Overall figures 8, 9, 10 and 11 have sufficient predictable variation in the attacks to be useful for building signature databases and determining rapid event response.

DISCUSSION

These tests have shown that an automatic tunnelling mechanism has vulnerabilities for the information in transition and also provides signatures while under attack that can be used to build a signature database for rapid responses to events. The tests themselves may be criticised as self-evident or constructed to deliver the outcomes. The extent of testing may be further commented that it is insufficient to cover a complete range of attacks and all the possible security flaws. We accept these concerns as being within the declared limitations of the research. The tests were selected to be feasible and to demonstrate rather than to prove the existence of vulnerabilities. As such the sniffing has confirmed the problems identified in the literature and others that are associated with information disruption and unintended use are present in the mechanism. The variation analysis of testing from inside and outside the tunnel indicates that the variations are sufficient in three instances to pursue the approach as a detection indicator with a potential system design. The implications of the results are consequently three fold in that they suggest concern regarding the security of the mechanism, the potential for security alerts and the necessity for further research.

The goals of network management are to keep the number of problems in the network to a minimum and to prevent those problems from spreading and causing further disruptions. Configuration management deals with monitoring and controlling the network's normal operations while the Fault management concerns with the abnormal behaviour in the network. The research suggests that awareness has to be built into network

management and security of the potential vulnerabilities with the IPv4 / IPv6 transition mechanisms so that planning is in place to assess the risk and to have the relevant controls. Tactically the faults must be detected, logged, isolated and dealt with as a continuous operation. Fault management processes should be alert to detect exceptional communications from both internal and external sources that may be capable of monitoring traffic in the transition mechanisms and functional to steal information and to disrupt channels. The performance of a network must be monitored, measured and maintained while security mechanisms should in place to make sure that access to resources in the network are accessible to authenticated users only. The results from the packet analyser tool proved that the 6to4 tunnel can be exploited and our test bed evaluation confirms that there are some irregularities in the behaviour in terms of data transmission around the 6to4 tunnel. As a result, the 6to4 automatic tunnelling mechanism can be exploited in various types of attacks such as, DoS, DDoS, spoofing and the man in the middle attack just to name a few. These mechanisms require inclusion in any network security risk assessment and potential problem analysis.

CONCLUSION

The research shows that an IPv4/Ipv6 automatic tunnelling mechanism has a number of security vulnerabilities that may be exploited to disrupt the legitimate use of the communicated information. Similarly tests show that the automatic tunnelling mechanism can be used to detect incoming attacks. The data for DoS attacks inside and from outside the tunnel both showed variations from the expected system benchmark data. Consideration of pre-configuring a network between end to end points (including across transition mechanisms) can greatly enhance the protection of information in a system and it is our recommendation that such precautions are taken in network security. Further research is to be done testing the other seven transition mechanisms and to extend the possible range of attacks in testing to record the signatures.

REFERENCES

Bi, J., Wu, J., & Leng, X. (2007). IPv4/IPv6 transition technologies and univer6 architecture. *International Journal of Computer Science and Network Security, 7*(1), 232-242.

Bilski, T. (2011). From IPv4 to IPv6–data security in the transition phase. *Proceedings of the Seventh ICNS 2011 International Conference on Networking and Services* (pp. 66-72). Venice: ThinkMind.

Cowley, J. (2007). Network Management. In *Communications and Networking* (pp. 153-167): London: Springer. doi:10.1007/978-1-84628-645-2_9

Cowley, J. (2013a). Network Management. In *Communications and Networking* (pp. 169-185): London: Springer. doi:10.1007/978-1-4471-4357-4_9

Cowley, J. (2013b). Network Security. In *Communications and Networking* (pp. 137-168): London: Springer. doi:10.1007/978-1-4471-4357-4_8

Gallaher, M., & Rowe, B. (2005). IPv6 Economic Impact Assessment (Final Report). *National Institute of Standards and Teehnology, US Department of Commerce*.

Gelogo, Y. E., & Lee, S. (2011). Internet migration and underlying security issues. *International Journal of Database Theory and Application, 4*(1), 49-54.

Guardini, I., Durand, A., & Lento, D. (2001). IPv6 Tunnel Broker. *Internet Engineering Task Force* RFC 3053.

Hadiya, D., Save, R., & Geetu, G. (2013). Network performance evaluation of 6to4 and configured tunnel transition mechanisms: An empirical test-bed analysis. *Proceedings of the 2013 6th Conference on Emerging Trends in Engineering and Technology (ICETET)* (pp. 56-60). Nagpur: IEEE. doi:10.1109/ICETET.2013.14

Hsieh, I. P., & Shang-Juh, K. (2005). Managing the co-existing network of IPv6 and IPv4 under various transition mechanisms. *Proceedings of the third International Conference on Information Technology and Applications* (pp. 765-771) IEEE. doi:10.1109/ICITA.2005.175

ITPSS. (2008). IPv6-to-IPv4 Transition & Security Issues. *Information Technology Protective Security Services*.

Julianne S. Sansa-Otim, & Mile, A. (2013). IPv4 to IPv6 transition strategies for enterprise networks in developing countries. In *e-Infrastructure and e-Services for Developing Countries* (pp. 94-104). Berlin Heidelberg: Springer.

Krishnan, S., Thaler, D., & Hoagland, J. (2011). Security Concerns with IP Tunnelling. *Internet Engineering Task Force RFC 6169*.

Molenaar, R. (2013). *How to configure IPv6 automatic 6to4 tunnelling*. Retrieved July 7, 2014, from http://networklessons.com/ipv6/how-to-configure-ipv6-automatic-6to4-tunnelling/

Mun, Y., & Lee, H. K. (2005). Interconnection between IPv4 and IPv6. In *Understanding IPv6* (pp. 115-149). US: Springer.

Narayan, S., Lutui, P. R., Vijayakumar, K., & Sodhi, S. (2010). Performance analysis of networks with IPv4 and IPv6. *Proceedings of the 2010 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)* (pp. 1-4). Coimbatore: IEEE. doi:10.1109/ICCIC.2010.5705805

Narayan, S., & Tauch, S. (2010a). IPv4-v6 configured tunnel and 6to4 transition mechanisms network performance evaluation on Linux operating systems. *Proceedings of the 2nd International Conference on Signal Processing Systems* (ICSPS) (pp. V2-113-V2-117). Dalian: IEEE.

Narayan, S., & Tauch, S. (2010b). Network performance evaluation of IPv4-v6 configured tunnel and 6to4 transition mechanisms on windows server operating systems. *Proceedings of the ICCDA 2010 International Conference on Computer Design and Applications.* (pp. V5-435-V5-440). Qinhuangdao: IEEE. doi:10.1109/ICCDA.2010.5540939

Narayan, S., & Tauch, S. (2010c). IPv4-v6 transition mechanisms network performance evaluation on operating systems. *Proceedings of the 2010 3rd International Conference on Computer Science and Information Technology (ICCSIT)* (pp. 664-668). Chengdu: IEEE. doi:10.1109/ICCSIT.2010.5564141

Savola, P., Krishnan, S., & Davies, E. B. (2007). IPv6 Transition/Co-existence Security Considerations. *Internet Engineering Task Force RFC 4942*.

Savola, P., & Patel, C. (2004). Security considerations for 6to4. *Internet Engineering Task Force RFC 3964*.

Waddington, D. G., & Chang, F. (2002). Realizing the transition to IPv6. *Communications Magazine, IEEE, 40*(6), 138-147.

Xin, M., & Xi, C. (2010). Research on IPv6 transition evolvement and security architecture of smart distribution grid data communication system. *Proceedings of the 2010 China International Conference on the Electricity Distribution (CICED)* (pp. 1 - 5). Nanjing: IEEE.