A Portable Network Vulnerability Analyser

Fergus D. R. Symon*

University of New South Wales at the Australian Defence Force Academy.

As both the defensive and offensive sides of computer security evolve, more innovative and dangerous attacks are created daily. It is critical that organisations are aware of their vulnerabilities. Security assessments are often expensive and time consuming processes. However, with automated testing, this process can be simplified. This project has developed a portable network vulnerability analyser that provides a versatile interface for an operator to undertake automated network testing. Validation of the device was accomplished through the development of a closed network test environment. Results have conclusively shown that the device was able to discover all devices on a network and take control of the data set across the network, while only minimally affecting the network's performance.

Contents

I Project Outline 2
  I.A Introduction ........................................... 2
  I.B Ethics and Legality ....................................... 2
  I.C Aim ...................................................... 3
  I.D Development Process .................................... 3

II ARP Spoofing 3

III Hardware 3
  III.A OpenBus Driven Design ................................ 3
  III.B FPGA Schematic Driven Design ......................... 5

IV Software 5
  IV.A Device Initialisation and Network Connection .................. 6
  IV.B Network Device Discovery .................................. 6
  IV.C Network Attack .......................................... 7
  IV.D Packet Inspection ........................................ 7

V Testing 8
  V.A Methodology .............................................. 8
    V.A.1 Python Traffic Generator .............................. 8
    V.A.2 Distributed Internet Traffic Generator ................. 9
  V.B Setup .................................................. 9

VI Results 9
  VI.A Test of Packet Forwarding ................................ 9
  VI.B D-ITG Results .......................................... 10

VII Conclusion 11

* Mr, School of Engineering and Information Technology, ZEIT4500
I. Project Outline

I.A. Introduction

With the ever increasing complexity of modern computer systems and networks, more advanced vulnerabilities are being discovered every day, while many of the simpler vulnerabilities are left unfixed, often due to backward compatibility requirements[10]. Discovering and testing for these vulnerabilities is often a complex process requiring special hardware that allows for more software control, specialised operating systems such as Kali Linux[6] and complex (and often undocumented) software calls[8]. Because of the devastating nature of these vulnerabilities, corporations spend large sums of money to detect, verify and remediate these weaknesses. Automated test environments that can produce repeatable tests and profile performance are difficult to set up and traditionally require expensive and difficult to use hardware[9]. This project is designed to tackle these issues, aiming to meet the requirement for a simple, yet thorough, testing of a live network and reducing the difficulty in the development of custom tests.

I.B. Ethics and Legality

This project involves the creation of hardware and software that can be used for malicious and illegal activities. These devices have been tested in closed test environments that were never connected to any external networks, including ADFA’s, any personal network, or the internet. These actions minimise the risk of external exposure to attacks and are legal methods of testing the project. Furthermore, the schematics, firmware and software are protected with a minimum of one layer of authentication and encryption, and will only be given to those that understand and have acknowledged the legal and ethical implications of the project.

Nomenclature

100BASE-TX The unshielded twisted pair copper cabling standard most commonly employed by Fast Ethernet.

ARP Address Resolution Protocol – the protocol used to translate an IP address (OSI layer 3) to a local hardware MAC address (OSI layer 2).

Fast Ethernet An OSI layer 1 physical connection standard employing unshielded twisted pair cable or optical fibre cable and a layer 2 data packet to address local physical hardware.

IEEE 802.11 The IEEE standard describing local wireless communications.

IEEE 802.3u The IEEE standard describing the cabling for Fast Ethernet.

IP Internet Protocol – the most popular network layer (OSI layer 3) protocol used by the communications throughout the internet and most current networks.

LwIP Lightweight Internet Protocol – an open source network library designed for embedded systems and provided by Altium in their TSK3000A software environment.

MAC address Media Access Control address – the 48-bit (for IEEE 802.11) local hardware address that enables computers within a subnet to directly communicate.

NTP Network Time Protocol – an extremely accurate time distribution protocol.

NVA Network Vulnerability Analyser – analyses a network’s vulnerabilities and reports back in a human-readable form; this research developed a standalone NVA.

OSI Open Systems Interconnection model – a layered model for abstracting communications over a network. Consists of seven layers; physical, data link, network, transport, session, presentation and application.

PHY Physical media layer linking device – used to encode and decode communication between a physical link (the cable) and a media access controller (MAC).
I.C. Aim

The primary aim for this project was to develop a hardware solution that allows for simple network vulnerability testing. The package is intended to be portable both in the sense of a small physical footprint as well as being simple to port software into and out of the platform. This portability allows for future students to simply test out their projects without the major effort presently required.

The primary focus is on wired networks through 100BASE-TX cabling (defined by IEEE 802.3u) which, along with 1000BASE-T (which is backwards compatible), are the most common corporate networking standards employed today[5]. This provides a simpler target for both the host and guest software development processes while providing a useful and realistic testing platform.

A test setup using cost effective hardware and software is required for the project. The purpose of this is to evaluate the various hardware solutions, to ensure correct network operation of the device and to evaluate the performance of changes that are made to the code base for each platform. This test setup was also to be used to test the operation and profile the performance of any vulnerabilities created for the platform.

I.D. Development Process

This project includes several major components: the hardware, software and the testing. A waterfall development approach was taken in order to break each component down further and ensure each major risk was considered, a solution developed, then verified prior to the next stage. Overall, this process led to success with the project. However, on one occasion, due to poor prioritisation of subcomponents a program developed to test the project had to be abandoned.

The subversion (SVN) and git source control programs were employed to track changes made to the hardware schematics and software respectively. This simplified management of backups through the use of off-site storage and goal tracking through the use of commit messages.

II. ARP Spoofing

A very simple example of an attack that is powerful and to which devices are still vulnerable is the address resolution protocol (ARP) spoof. This attacks the address resolution protocol used by devices to translate IP addresses into hardware (MAC) addresses in order for the link layer to forward a packet on. By sending ARP replies to computers on the network containing the attacking device’s MAC address and the target IP, it forces the attacked computer to update its ARP table record to the new address and subsequently send data to the attacking device rather than the correct device. The attacking device can then choose to read and modify the packet and forward it to the correct recipient, creating a man-in-the-middle attack; reply to the packet, masquerading as the intended destination; or, do nothing with the packet, creating a denial-of-service attack by blocking all network traffic.

III. Hardware

The available hardware drove the development process due to several key factors: availability, processing speed, memory size and software environment. The selection process involved identifying several options being an Altium NanoBoard 3000, Raspberry Pi or an Arduino Uno (with an Ethernet shield). A summary of the specifications of these platforms can be found in Table 1. It is clear that the Raspberry Pi is the most powerful platform in terms of raw specifications, however platforms such as the NanoBoard and Arduino allow for the development of a unique platform that has the ability, particularly in the case of the NanoBoard’s FPGA, to employ some novel techniques to carry out vulnerability analysis in lower power platforms. This, in addition to the higher overhead of the Linux operating system on the Raspberry Pi, has justified the selection of the Altium NanoBoard 3000 for the development platform of the NVA. This section shall outline the development process of this hardware and outline several areas for improvement.

III.A. OpenBus Driven Design

The hardware required for the NVA includes a processor, memory, networking hardware and communications with a host computer. For the NanoBoard, the TSK3000A soft-processor is the simplest solution since it is directly supported by Altium and is the only one that works on the Xilinx Spartan-3AN FPGA (on-board
Table 1: Summary of identified hardware platform specifications.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>TSK3000A RISC CPU on Xilinx FPGA</td>
<td>ARM1176JZF-S core</td>
<td>ATMega328P</td>
</tr>
<tr>
<td>Clock Speed</td>
<td>50MHz</td>
<td>700MHz</td>
<td>16MHz</td>
</tr>
<tr>
<td>Memory</td>
<td>1MB SRAM + 56KB FPGA Block RAM</td>
<td>256MB</td>
<td>2KB</td>
</tr>
<tr>
<td>Network</td>
<td>100BASE-TX</td>
<td>100BASE-TX</td>
<td>100BASE-TX</td>
</tr>
<tr>
<td>Communication</td>
<td>Serial/Flash Drive/SD Card</td>
<td>Flash Drive/SD Card</td>
<td>Serial/SD Card</td>
</tr>
<tr>
<td>Operating System</td>
<td>Minimal (partially POSIX-compatible)</td>
<td>Linux</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Custom OS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The NanoBoard). This processor operates a custom reduced instruction set and has a compiler included with Altium based on gcc. The processor is compliant with Altium’s wishbone connection scheme for accessing resources on a shared address bus in a Harvard architecture design. This wishbone architecture enables the TSK3000A to be connected to the 1MB of SRAM through an Altium FPGA-based SRAM controller.

The networking and host communication controllers are connected through the peripheral wishbone connection of the processor. The networking is provided through an FPGA-based Fast Ethernet media access controller (EMAC32) that translates the hardware PHY’s electronic symbols into Ethernet packets that are stored directly into the SRAM to be accessed by the TSK3000A. The host communications are provided by an Altium virtual terminal connected on the IO bus alongside the EMAC32, but could be replaced with an RS232 based physical serial connection to enable more portability with only minor schematic modifications.

Altium’s preferred method for FPGA design is through their OpenBus toolset which involves connecting the wishbone hardware from the OpenBus palette, followed by connecting the physical hardware in a top-level schematic to the generated OpenBus sheet. As seen in Figure 1a, the peripherals can be connected, allowing the OpenBus toolset to handle the wishbone connections. The PHY, SRAM, reset button and system clock can then be connected to their controllers through the top-level schematic in Figure 1b.

![OpenBus Schematic](image1.png)

![Hardware Connection Schematic](image2.png)

**Figure 1: Altium NanoBoard 3000 NVA – OpenBus schematics**

The use of the OpenBus toolset allows for two major avenues for improving network processing performance through transitioning from TSK3000A code and leveraging the FPGA’s ability to quickly process data in parallel. These are by intercepting and parsing data as it comes from the PHY or intercepting...
memory writes to the SRAM. A proof-of-concept was created, intercepting the signals coming from the PHY in an internal soft-instrument logic analyser, thereby verifying the ability to intercept and that the media independent interface (MII) standard is being used to communicate between the PHY and EMAC32 on the FPGA.

III.B. FPGA Schematic Driven Design

The OpenBus toolset approach to FPGA design enables a faster design process by abstracting the specific wishbone connections and simplifying the memory layout management required with the schematic driven approach. However, this abstraction of the wishbone connections reduces the number of accessible data lines to those going to external components that can be used in order to insert custom processing logic. Altium does provide the facility to break these connections up to their individual components, thereby requiring further memory layout design from the designer but allowing access to these memory buses. In doing this, the output of the EMAC32 can be accessed, enabling the interpretation of the packet in its reconstructed form. To test this method, the OpenBus design was transitioned to the schematic in Figure 2. However, it was found by doing this that it appears that Altium no longer supports the EMAC32 as a non-OpenBus wishbone component, therefore removing this opportunity for FPGA style processing.

![Figure 2: Altium NanoBoard 3000 NVA – FPGA sheet schematic](image)

IV. Software

The software for the network vulnerability analyser (NVA) had to be written specifically for the platform as no existing solutions for the NanoBoard/TSK3000A environment existed. Development of this software was a core ongoing task requiring the most resources when compared to other major sections of the project. Altium provides a fully-equipped micro-operating system, including the LwIP network library, network drivers for the Altium EMAC32, POSIX compatible standard libraries including file access and multithreading capabilities. The GCC compiler is provided to translate the C code to the TSK3000A instruction-set and programming occurs through Altium using the USB-to-JTAG NanoTalk FPGA on the NanoBoard.

To attack a network, a series of initial information gathering steps are first required. Both the network settings and a list of devices on a network must be gathered in order to first communicate with, then attack, the entire network. An overview of the process taken by the NVA to attack is shown in Figure 3. This section shall cover the steps taken in the software of the NVA to carry this process out and finally attack

---

*a in terms of development time and research required to achieve*
the network.

![Network Flow Diagram]

Figure 3: NVA Software Flow (red are communications with the computer, black are internal processes and green are communications with the network hardware through the LwIP library)

### IV.A. Device Initialisation and Network Connection

As with all hardware, at power on, the device must first initialise all hardware. The early steps are dealt with by Altium’s minimal operating system, prior to calling the `main()` method of the NVA software. From here, serial connections are opened and a “hello” is sent to the host. Finally, the EMAC32 is opened and instructed to begin acquiring the network settings.

The dynamic host configuration protocol (DHCP) is used to acquire network settings. DHCP operates by going through a process of discovering the DHCP server(s), receiving an offer from each of the servers, selecting and requesting an offer, and receiving an acknowledgement of the offer from the server. At a minimum, a local IP address, netmask and default gateway are provided by the DHCP server, however, often other settings such as DNS servers are sent. By taking these settings, the DHCP server and default gateway (if different to the DHCP server) can be added as one of the known hosts on the local network. From the IP address and netmask, the local network range can then be calculated with the Boolean equations (1) and (2) give the target range in order to begin network device discovery.

\[
\text{First Address} = IP \cdot \text{Netmask} + 1 \quad (1) \\
\text{Last Address} = IP + \text{Netmask} - 1 
\]

### IV.B. Network Device Discovery

With the local network address range calculated from the DHCP settings given, the hosts connected to the network must first be discovered. A ping sweep is carried out by the NVA by sending ICMP echo requests to all potential hosts. This ICMP request also creates an ARP request from the NVA, which all hosts should respond to. Therefore, if any host responds to any of these requests, it is recorded as an active host on the network. This is reported back to the host computer, including the method by which the computer was first observed.

A major disadvantage of the method used to intercept and inspect packets sent to the NVA (as outlined further in Section IV.D) is that it is unable to intercept ARP replies, which are processed prior. The
implication of this is that a polling method is required to determine whether a device responds to an ARP reply, which, when sending ARP requests to potentially thousands of IP addresses, can be a difficult process since ARP table entries are very short-lived in LwIP’s implementation of the ARP table. Therefore, the most reliable method for host discovery were replies to ICMP requests, since often ARP replies are missed. It is not uncommon to disallow ICMP replies from a computer\textsuperscript{b}, computers can therefore avoid initial detection by the NVA. However, since most computers often aren’t truly silent and emit multicast and broadcast traffic, they can still be located – the longer the NVA is able to listen to a network, the more likely it will acquire a complete map of the network’s hosts.

IV.C. Network Attack

As a host is discovered, it is recorded by the NVA. At the same time, in order to correctly receive packets destined for the additional address, the LwIP stack must be prepared by creating several structures and passing it to the stack. Initially, this was a major hurdle since it initially appeared that LwIP did not provided the capability to accept data from multiple IP addresses from a single network interface (the EMAC32). It was later discovered that, through a minor modification of the LwIP source code, this functionality could be achieved, enabling data destined for other hosts on a network to be processed by the NVA.

As described in Section II, the process of attacking the network involved forging ARP replies to requests that were never sent. This is done to each computer discovered on the network, convincing them that the NVA MAC address is the owner of each other IP address on the network. This process is immediately carried out by the NVA after the ping sweep, thereby taking control of all data sent across the network.

IV.D. Packet Inspection

It is critical for the NVA software to inspect most incoming packets in order to detect new devices on the network as well as locate packets that need to be forwarded to other devices. It was discovered that early in LwIP’s packet processing it checks a chain of raw packet servers for matches to process. These raw servers could be created to receive any IP address and any port, the only detail needed was which transport protocol it handled (TCP, UDP or ICMP). Therefore, three raw packet servers were inserted to the top of the chain that pointed to the custom packet inspector routine. This allowed the software to check the packets and pass them back to LwIP and allowed it to handle the packets in its own manner. The major advantage to this approach was that it did not require any modification to the LwIP source code. However, the major disadvantage to this method was that the network layer becomes decapsulated from the packet prior to the inspector, removing the inspector’s ability to determine the source MAC address of the sending device, only their IP address. Overall, this method was selected as there had been a significant research effort in discovering and the inability to access the network header could be worked around through use of the ARP table\textsuperscript{c}.

By intercepting packets intended for any destination, the raw packet inspector is able to pick up broadcast traffic and locate new devices on a network without polling. This is of great benefit to target discovery, since most modern operating systems will emit a small amount of broadcast traffic to the local network. Typically, since most home switches do not support IGMP snooping, multicasted packets sent on these smaller networks can also be used to enable target discovery\textsuperscript{[5]}. A potential extension to the software of the NVA would be to register the device on all potential (or a subset of the most common) multicast groups in order to receive this traffic on corporate networks where IGMP snooping is common\textsuperscript{[5]}.

After recording the source of the packet, the destination of the packet can then be determined. If the packet is destined for the NVA, it is passed back into the LwIP stack in order to allow it to process through the remainder of the stack, ultimately reaching the intended service (or to be dropped). If the packet is destined for another device, a process of forwarding is undertaken. In order to determine the correct destination’s MAC address, a local ARP table is looked up, if no MAC address has been recorded, the LwIP ARP table is polled. Finally if no match is found, an ARP request is sent and the ARP table is polled again after a delay. In most cases this process will terminate at the local ARP table, however, the process may take time if no address has been observed prior creating a large initial delay in forwarding the first packet to a new destination. With the aim being to locate the destination as fast as possible, the local ARP table is organised in a most recently used priority queue-like structure. When the destination has been located,

\textsuperscript{b}as a good security policy, this should be done
\textsuperscript{c}although this was not always reliable and resulted in a minor time penalty
the packet is modified to contain the NVA’s MAC address as the source and the correct destination’s MAC address as the packet’s destination.

V. Testing

V.A. Methodology

Real network traffic is hard to predict; often it will come as irregular bursts of data, other times it will be a large flow of data in a single direction, otherwise regular predictable flows of data, or variations to these arrangements. As such, it is difficult to test a device’s performance on a network, since one that is tuned to perform well in a particular environment may perform poorly in another[2]. In order to determine a device’s network performance, the following metrics can be measured (preferably under several network conditions):

- Latency – the time it takes for a packet to traverse a network, can be in a single direction (one-way delay) or include the return trip time (round-trip delay)
- Throughput – the number of bytes transferred across the network per second
- Error rate – the number of packets incorrectly transferred or dropped during transmission per second

Research was done that identified two potential network traffic generators: **iperf** and the distributed internet traffic generator (D-ITG). The features of each traffic generator have been summarised in Table 2. While **iperf** is more popular and provides better support, D-ITG includes features such as delay measurement and several traffic profiles that make it more suitable for use. However, a major disadvantage to using D-ITG was that it measured a one-way delay which, while more accurate, means that a common accurate time source for each of the computers must be created since each packet is tagged with the local time of each computer. There was an additional high level of difficulty in coordinating testing across a network using D-ITG, due to the difficult process of setting up servers on particular computers and starting clients on others in a coordinated effort. These difficulties led to the creation of the custom traffic generator which aimed to solve these disadvantages while still providing the advantages of each.

<table>
<thead>
<tr>
<th>Logging</th>
<th>iperf</th>
<th>D-ITG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Platforms</td>
<td>Linux &amp; Windows</td>
<td>Linux &amp; Windows</td>
</tr>
<tr>
<td>Network Protocols</td>
<td>IP</td>
<td>IP &amp; ICMP</td>
</tr>
<tr>
<td>Transport Protocols</td>
<td>TCP &amp; UDP</td>
<td>TCP, UDP, SCTP &amp; DCCP</td>
</tr>
<tr>
<td>Application Protocols</td>
<td>None</td>
<td>DNS, Telnet &amp; VoIP</td>
</tr>
<tr>
<td>Multicast</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Traffic Profiles</td>
<td>Constant</td>
<td>Constant &amp; several statistical distributions</td>
</tr>
<tr>
<td>Metrics</td>
<td>Throughput, Jitter &amp; Packet Loss</td>
<td>Throughput, Delay, Jitter &amp; Packet Loss</td>
</tr>
</tbody>
</table>

V.A.1. Python Traffic Generator

The primary goal for the custom traffic generator was to improve the coordination of the devices on the network and to centralise data collection on to one master computer. This was done by using a master device to distribute commands to slaves through a multicast network. The master controlled the number of servers on the network, the host operating each server, the transport protocol and bandwidth each server was using, as well as maintaining the list of registered clients. Each client would connect to every server on the network, transmit a random sequence of data and receive the sequence back from the server. From this, the bandwidth, two-way latency and error rate can be measured which would be reported back to the master and recorded.

This concept of operation was validated with a Python script. Python was selected because it enables fast prototyping and is capable of running on both Windows and Linux machines\(^d\). An **ncurses** based

\(^d\)a requirement for the test network

School of Engineering and Information Technology, UNSW Canberra at ADFA
user interface was selected since it provided a standard interface that could be operated from a screenless computer. Unfortunately, the selection of Python as the development language limited the maximum throughput of approximately 40Mb/s on a Fast Ethernet network that can achieve 96Mb/s. This limitation was discovered late in the development process due to the high priority placed on the user interface and no testing of Python’s throughput was done prior to being written into the traffic generator. The option of porting the code over to a compiled language such as C or C++ was considered, however, the incompatibility of the two languages was determined to be too significant that a porting effort in the remaining time would not be feasible and the project was dropped for the distributed internet traffic generator.

V.A.2. Distributed Internet Traffic Generator

The distributed internet traffic generator was selected as the successor to the custom python traffic generator because of the advantages summarised in Table 2, particularly its ability to record logs and measure traffic latency. The disadvantage of the accurate timekeeping requirement across the network was solved by setting up a global positioning system (GPS) module over a serial link to synchronise time on a host computer. A network time protocol (NTP) server was then used to distribute the time across the remaining computers on the network. The GPS timekeeping solution was selected to ensure isolation of the test network from any external networks, in addition to the inability to synchronise NTP from the ADFA network.

V.B. Setup

The environment used to test the NVA’s performance consisted of a test server and two test clients, one using Linux and the other Windows. The test server was designed to host and collect the D-ITG data as well as provide the time across the network. A Fast Ethernet switch was used which provided a maximum throughput of approximately 96Mb/s and a latency of less than 400µs. An overview of this setup can be found in Figure 4.

![Diagram of test environment](image)

Figure 4: The hardware environment used to test the NVA

VI. Results

VI.A. Test of Packet Forwarding

In order to verify the correct operation of the NVA, both the success of the attack as well as the correct operation of the packet forwarding mechanism needed to be verified. This was done using the `arp` program to verify the takeover of each network address and the `ping` program to send ICMP requests and replies between two devices on the test network. The Wireshark packet analysis program was operating on both devices to inspect the contents of each packet. The Wireshark outputs shown in Figure 5 demonstrate the correct modification of the ICMP packets, verifying the correct operation of the network.

A side-effect of this verification method was that some initial latency results could be taken using the round-trip latency measurement of `ping`. The initial results showed the network-baseline to be at a mean

---

9 of 11

School of Engineering and Information Technology, UNSW Canberra at ADFA

---

9 of 11

School of Engineering and Information Technology, UNSW Canberra at ADFA
latency of 0.392ms with a standard deviation of 0.026ms and no packet loss. When attacked, the mean latency increased to 3.050ms with a standard deviation of 0.082ms and no packet loss. These are promising results since the device is operating with a much lower clock rate than most other network devices and the network latency introduced is similar to that on a multi-subnetted network or that of a wireless network connection.

VI.B. D-ITG Results

Following the verification of the NVA’s operation, the distributed internet traffic generator was used to further measure the performance. A constant network data profile was used to measure the maximum constant throughput and the traffic effect the addition of the NVA had. The results in Figure 6 show that the unaffected network had a maximum throughput of 94Mb/s while the attacked network dropped down to 3Mb/s, a stark difference. In addition to this severe drop in the throughput, the increased latency and jitter seen prior can be confirmed with these results.

Figure 6: Attacked network throughput, latency and jitter from D-ITG with a constant data profile.
VII. Conclusion

This project created a hardware solution able to completely take control of a network’s data. Novel methods were used in the development of a hardware platform never used to undertake these attacks, and in software to achieve attacks with minimal effect to a network on the lower powered hardware. Significant effort was placed into developing an approach to accurately test the performance of the device on simulated “real network” conditions. While major issues appeared throughout the project, all significant aims were achieved to an outstanding level.

Throughout the project, several areas for improvement through future work were identified. These include the development of improved network discovery techniques such a DHCP packet inspection to discover new devices as they connect to a network. Additional attacks such as a DHCP spoof would provide improved opportunities to further control a network. Attack avenues, such as through the packet inspection and processing module, provide potential for more unique approaches to vulnerability analysis by pivoting off the existing attacks. A potential source for impressive packet processing speed improvements exist in transitioning modules from the soft-processor into FPGA based logic by intercepting packets as they enter from the network and passing them through a pre-processor.

References


School of Engineering and Information Technology, UNSW Canberra at ADFA