# Feasibility of Eliminating IDPS Devices from a Web Server Farm

Sujatha Sivabalan, P. J. Radcliffe
*(Corresponding author: Sujatha Sivabalan)*

The Department of Electrical and Computer Engineering& RMIT University
Swanston Street, Melbourne, Australia
(Email: s3365148@student.rmit.edu.au)

*(Received June 28, 2017; revised Aug. 27 & Sept. 25, 2017)*

## Abstract

Current web security systems need Intrusion Detection and Prevention Systems (IDPS), web proxies and firewalls to protect the websites from malicious network traffic. All these functions come at a cost for a web farm and add to power costs. Our previous work has concluded that the web server detection of application layer DDoS attacks is far more power efficient than an equivalent IDPS. This paper shows that all remaining IDPS functionality can be split between the firewall and the web server allowing the removal of the traditional IDPS and so substantially reducing the CPU load and total electrical power bill of a web farm.

*Keywords: Application Layer (App-layer); Intrusion Detection and Prevention System (IDPS); Web Server*

## 1 Introduction

The Web plays a vital role in modern life but web servers are under constant attack. Common attacks include buffer overflow based attacks, cross-site scripting, code injection, brute force attacks, and DoS attacks [11, 13]. A web security system is usually built around web proxies, firewalls and IDPS. The IDPS play a significant role in blocking attacks but they can be overwhelmed by high traffic levels and consume electrical power. In a modest system such as a host-based web server, the network traffic is received, analysed, and transmitted by these security devices before getting to the web server. Consider the above scenario; the packet needs to traverse the full TCP/IP stack three times before the web server. Consider a web server farm in an enterprise that operates many web servers with a large number of security devices. These security devices consume significant power due to this repetitive packet.

A possible solution is to remove any devices if the functionality can build into other existing devices. Former research work [14] analysed the power consumption of traditional IDPS as a separate device in a host based web server and compared this to a Two Dimensional Web page Daemon (TDWD) which implemented IDPS functionality within the web server in a host based system. The experimental results show that the traditional IDPS consumed significantly more power than the TDWD. Based on this novel foundation, this paper examines the possibility of detecting all types of attacks by distributing the IDPS functionality between a web server and other devices thus eliminating the IDPS box or IDPS software such as Bro or Snort. If this proves feasible, it will reduce equipment costs and the electricity usage in a web server farm.

This paper is organized as follows: Section 2 describes the different types of attacks handled by an IDPS and explores attack detection strategies for web servers and firewalls. Section 3 explains a novel security system that eliminates the IDPS. Section 4 considers practical implementation. Section 5 suggests some exciting future work based on the new architecture. Finally Section 6 provides a conclusion.

## 2 Literature Search

This first part of this section will examine IDPS functionality and discuss the types of attacks such a system can detect and block. The second part will then show that many authors have suggested how individual IDPS functions can be implemented either within a firewall or web server.

### 2.1 IDPS Functionality

An IDPS aims to detect and alert the system when suspicious traffic occurs and blocks the offending traffic [18]. IDPS detection methodologies on major attacks are discussed below.

Phishing [12] is a common problem in an email, where the embedded hyperlink in an apparently legitimate email redirects to the fake website which aims to steal user secrets. Common aims of this attack includes financial gain, identity hiding mainly in the purchase of goods and for fame and notoriety. Phishing attack is possible through the websites blogs and on commercial websites. IDPS apply signature-based detection for phishing attacks. The author Khonji *et al.* [8] describes the Collaborative Intrusion Detection System (CIDS) where many number of IDS share phishing related data and each IDS will maintain a list of infected IP addresses, pattern matches to mitigate phishing attacks.

Several major websites including eBay, Google, and McAfee have been the targets of cross-site scripting [4], SQL injection exploits, or content based sniffing [6, 17]. An attacker injects malicious script in the web application thereby causing unintended script execution by the victims browsers. Once this attack is successful, the attacker can perform exploits such as account hijacking, cookie poisoning, DoS and web content manipulation. IDPS detection on these attacks are based on the signature rules such as pattern matching, whitelist techniques which compare inputs with the known good inputs, and model-based approaches to analyse the user behaviour [18].

Brute force attacks are an illegal attempt to websites by repeatedly trying username and password. Major victims are email and banking user. IDPS detect these types of attacks by pattern matching [18]. According to a review by Hydara *et al.* [6] this functionality can be provided by a web server.

Cookie poisoning is a fraudulent act on cookie data after accessing a website. This is a common attack on web applications, for example in an online shopping. An attacker can poison the cookie by neglecting the shipping fee or postal price using tools as Paros proxy [9] that results in financial loss to the owners. IDPS detect these types of attacks by state transition analysis or by model-based approaches.

Network layer DoS (Net-layer DoS) attacks include SYN attack, ICMP attack, and UDP attack [5]. These attacks aim to flood the server and make it unavailable to the legitimate requests. IDPS will detect and block flood attacks by using state transition analysis.

Well known commercial IDPS products such as those from Cisco IDS, Snort or Bro can detect these types of attacks with the help of attack signature matching in central databases. IDPS plays a major role in detecting and blocking attacks. Most inline IDPS sensors offer firewall capabilities to mitigate the suspicious network activity.

### 2.2 Detection Techniques by Web Server And Firewall

This subsection examines how web servers and firewalls [2] can take over responsibilities of an IDPS. There is a good body of research showing that web servers provide effective detection against individual forms of application layer (app-layer) attacks. For example WebIDS [7] from IBM Tivoli Risk Manager analyses the Web servers access log files to detect Web server attacks. Apache server modules such as mod_security, mod _evasive, and mod_rewrite [10] can be configured to defend against applayer attacks. Mod_security is as web application firewall designed for blocking applayer attacks. Mod_evasive is an Apache module used to detect DDoS attacks, the detection is based on number of single page access per unit time. Anton *et al.* [1] deployed a web server to detect cookie poisoning and SQL injection and stated that server side detection is more powerful for cross-site scripting than a firewall. Their approach based on checking the payload content such as response headers, < Meta tag > and the number of bytes. Web servers can filter for phishing attacks [8] with the help of blacklisting and whitelisting IP.

Web servers are unsuitable for Net-layer DoS attacks such as SYN, ICMP and UDP flood attacks. Haining *et al.* [5] showed that an advanced firewall is capable of resisting these types of flooding attacks. Gallagher [3] stated that instead of using an IDPS, firewalls could be configured to block Net-layer attacks and also Domain Name Service (DNS) and Network Time Protocol (NTP) reflection attacks.

Commercial web application scanners include AppScan, WebInspect, Hailstorm, Acunetix WVS. Open source web application scanners, include Paros and Pantera. These scanners examine the log files from the web server to detect problems. This is essentially an IDS function but not a real time thus making them less useful for directly blocking unwanted traffic.

The literature has outlined the functionality of IDPS and has shown that all individual app-layer IDPS functions can be moved into the web server, and that individual net-layer IDPS functions can be implemented in a firewall. There is no overarching reasoning as to why the IDPS is still required. There is no commentary on the possibility or advantages of completely eliminating an IDPS.

## 3 Novel Architecture

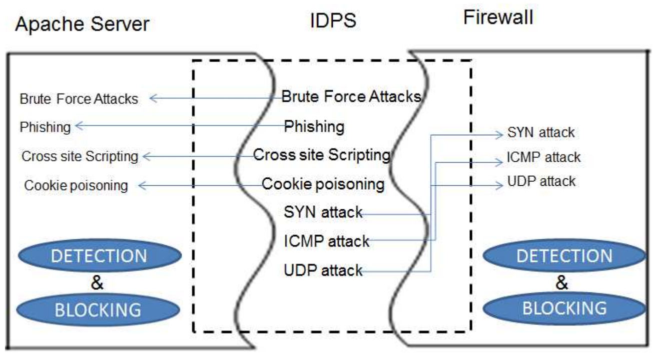

Figure 1: Attacks handled by web server and firewall

The literature search showed that many authors have implemented individual IDPS functions on the firewall or

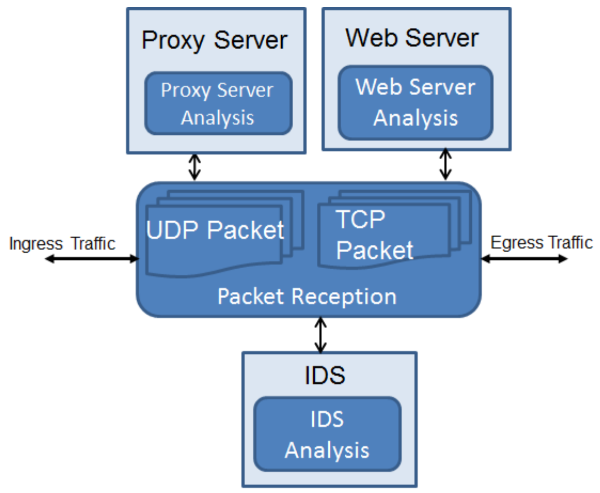

Figure 2: Novel architecture

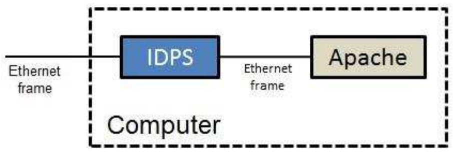

Figure 3: Traditional network design

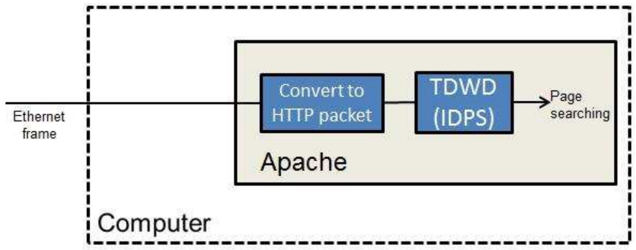

Figure 4: Novel design

web server. This section will show that the total IDPS function can successfully split between a web server and a firewall thus making it possible to eliminate the IDPS. Furthermore, it will show that this results in real power and CPU savings even in a cloud environment.

### 3.1 Eliminating IDPS

Figure 1 illustrates the attacks handled by IDPS can be splitted between a web server and a firewall. The Apache web server has powerful modules such as mod_status, mod_rewrite, mod_evasive, mod_clamav, and mod_proxy that can be customised using HTTP variables for effective detection of attacks.

The detection techniques used by web server modules are listed in Table 1. The Apache web server is capable of detecting all types of attacks except network layer attacks, which the firewall can handle. The first column lists the common types of attacks. The second column describes the user activities on each attack. The IDPS detection and blocking methods on each one is summarized in the third column.

The last column explores the web server techniques in handling those attacks. All attacks can be handled by the web server or firewall thus it is possible to eliminate the IDPS. In situations requiring very high security the redundancy of having an IDPS may be considered worthwhile but there are no attacks which and IDPS can handled that cannot be handled by the web server and firewall together.

### 3.2 Power Saving Opportunity

Figure 1 showed that much of the IDPS functionality can absorbed into the web server and so it is feasible that the security modules can work at the HTTP, TCP, or UDP level. This proposed a novel architecture is shown in Figure 2.

The system receives packets to the TCP, UDP, or HTTP level and these formed packets are shared between the applications thus the packets need not to travel up and down TCP/IP stack, from Ethernet frame to TCP/UDP and from TCP/UDP back down to Ethernet frame. Avoiding multiple packet reception and transmission on each devices results in far less CPU computing time and the use of far less electrical power. The next section examines the size of this electrical power saving.

## 4 Practical Implementation

The basic architecture of a traditional IDPS and web server is shown in Figure 3 where the ingress and egress traffic for each network device is at the level of the Ethernet frame. The IDPS used in our work was Snort, and another called Bro. Section 3.2 showed that this was computational inefficient and proposed a generic solution in Figure 2. Figure 4 shows how this was implement as a way to eliminate Bro or Snort when working with Apache. This novel design has been implemented in our previous work [14, 15, 16] and further extended in this paper. Our TDWD is built into the Apache web server and uses a two-dimensional linklist (client IP and time) with time based garbage collection. The following rules have been implemented in TDWD:

- DDoS attack: excessive similar page accesses per second, accessing pages at random, and repeatedly accessing the same page. Blocking rules can be evaluated by examining the linklist of accesses for each user IP.
- Brute Force attack: repeatedly accessing the administration page.
- Blocking: Blacklisting based on user IP.

Table 1: Attack methods handled by web server

| Attacks | User Behavior | IDPS | Web server detection |
|---|---|---|---|
| Brute force attacks | Many failed login for a single account from same IP address or different IP addresses | • IDS signatures<br>• Pattern matching<br>• Block blacklist IP's<br>• Account lockout<br>• CAPTCHA | • Blacklist IP address using "HTTP USERAGENT"<br>• Redirect CAPTCHA webpage |
| Phishing | Illegal URL and URI and Forged mail headers. | • Block blacklist IP's<br>• Block illegal request<br>• IDS signatures | • Blacklist IP address using "HTTP USERAGENT"<br>• Block through Request using "THE_REQUEST" or "REQUEST_URI"<br>• Block through the referrer using "HTTP_REFERER"<br>• Get and Post scanning for blockable text eg bad web site. |
| Cross site scripting | Malicious script execution in the HTTP request. | Anomaly based detection | • Blacklist IP address using "HTTP USERAGENT"<br>• Block through Request using "THE_REQUEST" or "REQUEST_URI"<br>• Block through the referrer using "HTTP_REFERER"<br>• Block through Query String.<br>• Scan post and get for any script, reject if any found. |
| Cookie poisoning | Change in content of Cookie data. | • Block blacklist IP's<br>• Block illegal request<br>• IDS signatures | • Blacklist IP address using "HTTP USERAGENT"<br>• Block through Request using "THE_REQUEST" or "REQUEST_URI"<br>• Block through the referrer using "HTTP_REFERER"<br>• Block through "HTTP_Cookie" |

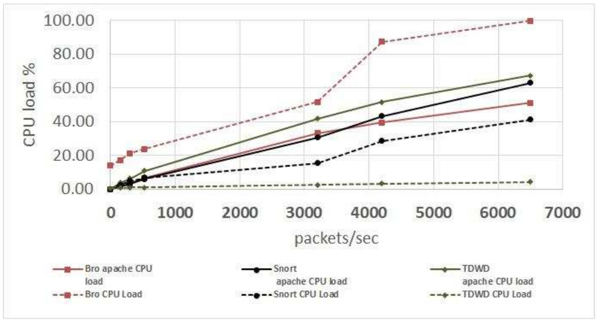

Figure 5: Bro, Snort and TDWD a comparison

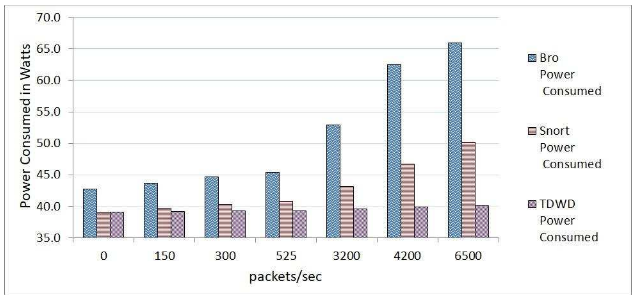

Figure 6: Power consumption of Bro, Snort and TDWD

In order to analyse the power savings of the TDWD approach three experimental setups were devised; the Bro IDPS running with Apache, the Snort IDPS running with Apache, and TDWD which integrates into Apache. In order to achieve a realistic comparison each IDPS devices was run by itself on the same machine with Apache active and servicing packets. The CPU usage is shown in Figure 5 where the dotted lines are the CPU load of the individual IDPS devices, and the solid lines show any variation in the Apache CPU load given the IDPS being used. Our previous research [16] shows that the CPU usage is linearly related to the power consumption. Using a watt meter we successfully calibrated our computer against this model which enables the translation of the CPU utilization in Figure 5 to real watts in Figure 6. This shows that at the highest traffic load TDWD consumes 40 watts, Snort 50 watts, and Bro 65 watts.

Additionally the IDPS programs were configured to receive the network traffic but do no processing. The power consumption was proportional to the packets per second. When the IDPS analysis was added, the power consumption increased very marginally. The receive process, which is dominated by Ethernet frame to TCP/UDP translation, was responsible for the majority of the power consumption in the IDPS. To confirm this, the same type of packet analysis was programmed into TDWD, which runs inside Apache and inspects the already assembled HTTP packets. This added functionality marginally increased the power consumption of Apache but this was under 10% of the power used by Snort or Bro to achieve the same functionality.

The experimental work has clearly shown that IDPS functionality can be moved into a web server and that further more there is a significant saving in electrical power. This work points to a general methodology whereby network devices share data at the highest possible level (HTTP, TCP, or UDP) and do not waste CPU time and power in unnecessary conversions between Ethernet frames and higher levels. This approach could be useful in a wide variety of network designs.

## 5 Future Work

The TDWD is novel and powerful tool as the link list of user page requests with time stamping allows complex DDoS detection rules to be implemented as well as more basic rules such as blacklisting. There is considerable scope to develop rules well beyond the simple rules we have implemented. There is also scope for machine learning to analyse the link list to detect anomalies caused by attacks. The TDWD structure is suitable for other research projects, as it will work in a real webserver or in a cloud network. The focus of this work has been the reduction in electrical power usage by allowing networking devices to share data at the highest possible level, HTTP, TCP, or UDP. By reducing CPU utilization not only is power reduced, the speed of network operations should also be significantly improved. The ability to speed up network devices in this way and the actual time savings achievable are an interesting research area. Not only can networks be made greener, they can be made faster.

## 6 Conclusion

This paper has examined the literature and has concluded that an IDPS device can have all its functionality moved into the firewall or web server thus the IDPS may be removed. An IDPS may be kept for reasons of security redundancy but it is not required as a function. The IDPS functions that are moved into the web server can be implemented at the UDP, TCP, or HTTP level thus eliminating the repeated conversions between Ethernet frame and TCP/IP. This results in a useful saving of CPU capacity and electrical power. In proving that there was a saving of electrical power, a Two Dimensional Web page Daemon (TDWD) was developed to hold user access requests so they could be analysed for attacks, particularly DDoS attacks. This structure has proved to be very useful and may be the basis for developing novel intelligent attack detection and blocking algorithms.

## References

[1] A. Barua, H. Shahriar, and M. Zulkernine, "Server Side Detection of Content Sniffing Attacks," in *IEEE 22nd International Symposium on in Software Reliability Engineering (ISSRE'11)*, pp. 20-29, 2011.

[2] A. Blyth, "An architecture for an XML enabled firewall," *International Journal of Network Security*, vol. 8, pp. 31-36, 2009.

[3] S. Gallagher, *Biggest DDoS Ever Aimed at Cloudflare Content Delivery Network*, May 24, 2016. (`http://arstechnica.com/security/2014/02/ biggest-ddos-ever-aimed-at-cloudflares-content-delivery-network/`)

[4] S. Goswami, N. Hoque, D. K. Bhattacharyya, and J. Kalita, "An unsupervised method for detection of XSS attack," *International Journal of Network Security*, vol. 19, pp. 761-775, 2017.

[5] W. Haining, Z. Danlu, and K. G. Shin, "Detecting SYN flooding attacks," in *IEEE Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'02)*, pp. 1530-1539, 2002.

[6] I. Hydara, A. B. M. Sultan, H. Zulzalil, and N. Admodisastro, "Current state of research on cross-site scripting (XSS): A systematic literature review," *Information and Software Technology*, vol. 58, pp. 170-186, 2015.

[7] IBM, *Tivoli Risk Manager Analyses*, Jan. 8, 2018. (`https://publib.boulder.ibm.com/tividd/td/TRM/GC32\-1323\-00/en\_US/HTML/ad`)

[8] M. Khonji, Y. Iraqi, and A. Jones, "Phishing detection: A literature survey," *Communications Surveys & Tutorials*, vol. 15, pp. 2091-2121, 2013.

[9] J. Long, A. W. Bayles, J. C. Foster, C. Hurley, M. Petruzzi, N. Rathaus, et al., *Chapter 4 - Web Server & amp: Web Application Testing*, Penetration Tester's Open Source Toolkit, ed Burlington: Syngress, pp. 189-276, 2005.

[10] I. Muscat, *How To Mitigate Slow HTTP DoS Attacks in Apache HTTP Server*, May 5, 2015. (`https://www.acunetix.com/blog/articles/slow-http-dos-attacks-mitigate-apache-http-server/`)

[11] A. A. Orunsolu, A. S. Sodiya, A. T. Akinwale, B. I. Olajuwon, "An anti-phishing kit scheme for secure web transactions," *International Journal of Electronics and Information Engineering*, vol. 6, no. 2, pp. 72–86, 2017.

[12] A. San Martino and X. Perramon, "Phishing secrets: History, effects, countermeasures," *International Journal of Network Security*, vol. 11, pp. 163-171, 2010.

[13] J. J. Sheu, "Distinguishing medical web pages from pornographic ones: An efficient pornography websites filtering method," *International Journal of Network Security*, vol. 19, no. 5, pp. 839-850, 2017.

[14] S. Sivabalan and P. J. Radcliffe, "Real time calibration of DDoS blocking rules for web servers," *Computer Communication & Collaboration*, vol. 4, pp. 42-50, 2016.

[15] S. Sivabalan and P. J. Radcliffe, "A novel framework to detect and block DDoS attack at the application layer," in *IEEE TENCON Spring Conference*, pp. 578-582, 2013.

[16] S. Sivabalan and P. J. Radcliffe, "Power Efficient Secure Web Servers," *International Journal of Network Security*, vol. 20, no. 2, pp. 304-312, 2018.

[17] M. Stampar, "Inferential SQL injection attacks," *International Journal of Network Security*, vol. 18, pp. 316-325, 2016.

[18] T. Verwoerd and R. Hunt, "Intrusion detection techniques and approaches," *Computer Communications*, vol. 25, pp. 1356-1365, 2002.

## Biography

**Sujatha Sivabalan** received a B.Eng from Bharathidasan University(India)in 2004 and M.Eng from Anna University (India) in 2007.Currently doing Ph.D. in School of Electrical and Computer Engineering, RMIT University, Australia. Her research interest includes network security, DDoS attack detection and blocking.

**P. J. Radcliffe** received a B.Eng from Melbourne University (Australia) in 1978 and worked for Ericsson Australia R&D for 7 years followed by consulting to various companies. He joined Royal Melbourne Institute of Technology (RMIT) and was awarded and M.Eng. in 1993 and a PhD in 2007. Main research interests include network protocols, Linux, and embedded systems. He received a national teaching award in 2011 and in 2012 received the RMIT early career researcher award.