REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* DEC 2014	**2. REPORT TYPE** JOURNAL ARTICLE (Post Print)	**3. DATES COVERED** *(From - To)* JAN 2011 – DEC 2011

4. TITLE AND SUBTITLE Protection Without Detection: A Threat Mitigation Technique	**5a. CONTRACT NUMBER** IN-HOUSE
	5b. GRANT NUMBER N/A
	5c. PROGRAM ELEMENT NUMBER 62788F
6. AUTHOR(S) Joshua White, Joseph R. McCoy, Paul Ratazzi	**5d. PROJECT NUMBER** GAIH
	5e. TASK NUMBER CY
	5f. WORK UNIT NUMBER BR
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/Information Directorate Rome Research Site/RIGA 525 Brooks Road Rome NY 13441-4505	**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/Information Directorate Rome Research Site/RIGA 525 Brooks Road Rome NY 13441-4505	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/RI
	11. SPONSORING/MONITORING AGENCY REPORT NUMBER AFRL-RI-RS-TP-2014-055

12. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA Case Number: 88ABW-2012-2407
DATE CLEARED: 23 APR 2012

13. SUPPLEMENTARY NOTES
© 2012 SPIE. Defense Transformation and Net-Centric Systems 2012, edited by Raja Suresh, Proc. of SPIE Vol. 8405, 84050B. This work is copyrighted. One or more of the authors is a U.S. Government employee working within the scope of their Government job; therefore, the U.S. Government is joint owner of the work and has the right to copy, distribute, and use the work. All other rights are reserved by the copyright owner.

14. ABSTRACT
Networking systems and individual applications have traditionally been defended using signature-based tools that protect the perimeter, many times to the detriment of service, performance, and information flow. These tools require knowledge of both the system on which they run and the attack they are preventing. As such, by their very definition, they only account for what is known to be malicious and ignore the unknown. The unknown, or zero day threat, can occur when defenses have yet to be immunized via a signature or other identifier of the threat. In environments where execution of the mission is paramount, the networks and applications must perform their function of information delivery without endangering the enterprise or losing the salient information, even when facing zero day threats. In this paper we, describe a new defensive strategy that provides a means to more deliberately balance the oft mutually exclusive aspects of protection and availability. We call this new strategy Protection without Detection, since it focuses on network protection without sacrificing information availability.

15. SUBJECT TERMS
network security, zero-day threat

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** **PAUL RATAZZI**
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	UU	8	**19b. TELEPHONE NUMBER** (*Include area code)* N/A

Protection Without Detection: A Threat Mitigation Technique

Joshua White[a], Joseph R. McCoy[a], Paul Ratazzi[b]
[a]Everis Inc., 110 RR. Street, Frankfort NY 13357
[b]Air Force Research Laboratory, 525 Brooks Road, Rome NY 13441

ABSTRACT

Networking systems and individual applications have traditionally been defended using signature-based tools that protect the perimeter, many times to the detriment of service, performance, and information flow. These tools require knowledge of both the system on which they run and the attack they are preventing. As such, by their very definition, they only account for what is known to be malicious and ignore the unknown. The unknown, or zero day threat, can occur when defenses have yet to be immunized via a signature or other identifier of the threat. In environments where execution of the mission is paramount, the networks and applications must perform their function of information delivery without endangering the enterprise or losing the salient information, even when facing zero day threats. In this paper we, describe a new defensive strategy that provides a means to more deliberately balance the oft mutually exclusive aspects of protection and availability. We call this new strategy Protection without Detection, since it focuses on network protection without sacrificing information availability. The current instantiation analyzes the data stream in real time as it passes through an in-line device. Critical files are recognized, and mission-specific trusted templates are applied as they are forwarded to their destination. The end result is a system which eliminates the opportunity for propagation of malicious or unnecessary payloads via the various containers that are inherent in the definition of standard file types. In some cases, this method sacrifices features or functionality that is typically inherent in these files. However, with the flexibility of the template approach, inclusion or exclusion of these features becomes a deliberate choice of the mission owners, based on their needs and amount of acceptable risk. The paper concludes with a discussion of future extensions and applications.

Keywords: Protection, Detection, Threats, Avoidance, Mitigation, Mission Assurance

1.INTRODUCTION

The layered approach to networking allows for a heterogeneous combination of software and equipment to operate seamlessly, as if it had been designed to work together from the start. Data delivery mechanisms in high speed networks generally operate in a way that is transparent to the actual application, purpose or mission that is being supported. As such, defenses employed at the networking and transport layers also operate independently of the mission they are protecting. Today's best practices for intrusion detection system / intrusion prevention system (IDS/IPS) deployments focus on preventing attacks rather than ensuring the mission. Since these systems exist to detect traffic containing known malicious or disallowed content and prevent it from being delivered, the mission-critical data associated with flagged content is usually partially or entirely blocked, resulting in degradation or denial of service. Furthermore policies and threat information that drive the nature of these deployments are usually created by information assurance personnel without much regard to the true information needs of the mission being supported. All of this is contrary to the concept of mission assurance, which requires that mission critical data and processing be available, even when systems are under attack and/or compromised. Hence, if we are to put mission assurance as the ultimate goal when operating a supporting information system, we must rethink how threats and attacks are addressed.

Approved for Public Release; Distribution Unlimited: 88ABW-2012-2407, 23-APR-2012

Defense Transformation and Net-Centric Systems 2012, edited by Raja Suresh, Proc. of SPIE
Vol. 8405, 84050B · © 2012 SPIE · CCC code: 0277-786X/12/$18 · doi: 10.1117/12.918272

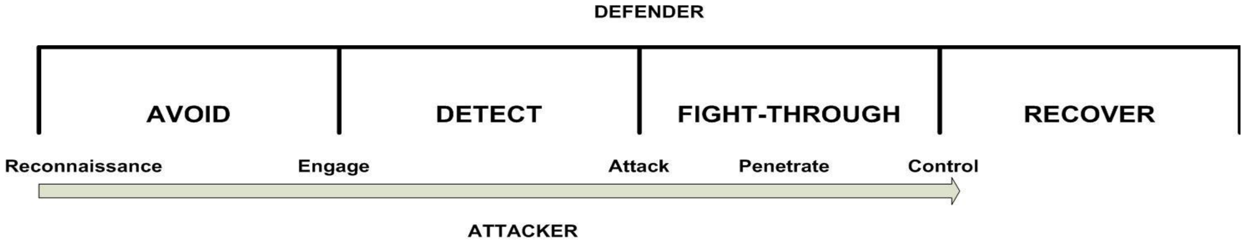

Figure 1: Phases of defense aligned with the life cycle of attacks.

1.1 BACKGROUND

For decades, computer and network security architectures have depended on the premise that the inability to achieve perfect security in the system design phase can be compensated by a strategy of detect-and-respond in the system operation phase. Unfortunately, even recent history provides ample evidence that this premise is flawed. [1] Systems are constantly subjected to zero day attacks that exploit conditions not addressed by the original system specifications, the run-time IDS/IPS, or both. Furthermore, many attacks that were foreseen by the designers and/or deployers of these systems are still successful in interrupting normal system operations for some period of time, due to response latency or inadequacy. When these types of attacks occur in a mission-critical system that supports operations with tolerances shorter than detection and response times, mission failure is likely to occur. In Department of Defense (DoD) or other critical environments, this is unacceptable.

Does this mean that every mission-critical information technology (IT) system must be designed from the bottom up, using rigorous systems engineering processes and formal methods in order to mathematically prove its infallibility? If cost and other practical constraints are eliminated, this approach would likely achieve some measure of success. However, in today's cost-conscious, technology-driven environment, this approach would likely yield outdated and expensive systems better suited to a monolithic, symmetrical cold war-era adversary.

In contrast, the IT systems used today to support the diverse missions of all types of organizations are constantly-evolving, heterogeneous compositions of many different technologies, old and new, built from a variety of many independent sources. This includes those that support the mission-essential functions of major weapon systems and portions of critical infrastructure. Although at first, this chaotic state itself may appear to be the problem, we propose that it is simply the nature of modern cyber-enabled systems and thus becomes the motivator and key design consideration for a new defensive strategy.

1.2 A NEW STRATEGY: PROTECTION *WITHOUT* DETECTION

The upper portion of figure 1 depicts the four phases of defense. The first phase, *avoid*, is typically the concern of designers that build systems to various recognized information assurance (IA) principles and tenets. Threat avoidance is also the primary goal of system integrators that install and configure specific customer sites using standard checklists, configurations, and current threat intelligence. During this phase, many attack vectors are eliminated or blocked, a security audit is conducted, additional measures are deployed to address findings, and the typical result is an initial certification or authority to operate on a particular network, and the system becomes operational. While these up-front processes are helpful to an extent, the system is now operational with the assumption that undiscovered flaws may and likely do exist. Since current defensive strategy makes no further use of the avoid phase, defense against attacks that exploit these undiscovered flaws becomes entirely dependent on the next phase, *detect*. Considering the alignment of this phase of defense with the attack life cycle, shown at the bottom of figure 1, this means any further defensive actions are necessarily reactive and based solely on adversary triggers (i.e., engage, attack). As discussed earlier, the track record of attack detection is questionable at best, and largely ineffective against the most dangerous zero day attacks.

This high-risk situation arises because of a defensive strategy that fails to leverage avoidance techniques during system operations.

Our work in developing the protection without detection strategy is an attempt to redefine standard network defense strategy by extending the threat avoidance phase into the operational portion of a system's lifecycle. In doing so, we are able to change the focus of operational security mechanisms from attack detection and information flow blocking to threat elimination and information flow preservation.

At a high level, protection without detection begins by identifying the information flows on the network, their purpose, and their requirements for the underlying communication system. For example, a daily intelligence report might consist of a text document with several images attached. Such a document might be created and transmitted as a PDF file. Under nominal conditions, PDFs might be a great choice for sharing such information. However, PDF is a very rich format that has much more capability than is needed for transmitting simple text along with a set of images. In this example, the minimum functionality necessary for accomplishing the mission probably does not depend on extended features such as embedded hyperlinks, scripts, macros, etc. Many times, these features represent potential attack channels to the adversary. A format that allows these extra features, even though they are not needed or normally used, adds significant unnecessary risk because of the additional opportunities of attack through one or more of these channels.

In a detection-centric system, the system would scan each PDF for known vulnerabilities and attack signatures. Upon detection, the security system could then take some sort of preemptive action, such as deleting the file and warning the sender and/or recipient. If detection fails, the file would most likely be delivered with the potential result being a degradation or disruption of any missions dependent on the affected system(s). Even in the case of a successful detection, the mission would likely still suffer, since the security system would most likely prevent the file and its needed information content from being delivered.

In our protection without detection approach, each and every file is transformed according to a mission-specific template. In the previous example, the template would be constructed with an understanding of the information transfer process and knowledge that only the plain text information of the document was relevant. As long as the content is preserved, the particular file format or delivery mechanism is unimportant. The PDF file might then be converted into a plain text file and several separate GIF images. These GIF images might then be converted further into JPG files. These additional conversions serve to increase the likelihood that format-specific attack vectors have been eliminated. Every file goes through this mission-specific transformation process, thus protecting the destination without having to rely on detection of specific undesired content within the file.

2. PREVIOUS WORK

Current file reconstruction techniques, such as tcpxtract[1] and foremost[2], are limited in both file recognition support and speed. These systems do not support kernel space acceleration and do not offer support for re-injection back into the network stream [6]. These limitations are primarily caused by the high layer user space operation of these applications. Typical tools such as these rely on the standard libpcap as the method for capture, identification and reconstruction of packets.

A number of methods have been developed to design perimeter protection systems where security is defined by a series of policies or rules. These systems often result in an amassment of unmanageable criteria, causing administration to become a cumbersome task. Our system uses a combination of both policies and rules with additional data fields. We have automated some of the rule creation tasks using policies that define global categories. For example, this means that

[1] tcpxtract: http://tcpxtract.sourceforge.net/

[2] Foremost: http://foremost.sourceforge.net/

many specific rules of the form *"from source IP on Port X deny Protocol Y if criteria n1 is met"* may be simplified and consolidated using common terms as follows: *"deny all connections with .js in the stream"*.

Many throughput-intensive network security applications rely on multi-core CPUs to accomplish packet inspection and manipulation at line speeds. Multi-core systems offer a number of benefits to security applications which are usually developed as either multi-threaded or parallelized algorithms [12, 13, 14, 15]. In order to lower the complexity of running multiple instances of our core system while trying to de-conflict memory space and concatenate log files, we have chosen a multi-threading design approach.

3. APPROACH

The technical approach discussed herein combines protocol and meta-data analysis technique with network stream manipulation techniques for rendering potentially harmful files inert. This process is based on the disarmament of active fields within file transfers and not thorough the detection of actual malicious content. At its core, this technique identifies files within a network flow and selectively reconstructs them with the template-specified active content removed and then forwards them on to their intended destination.

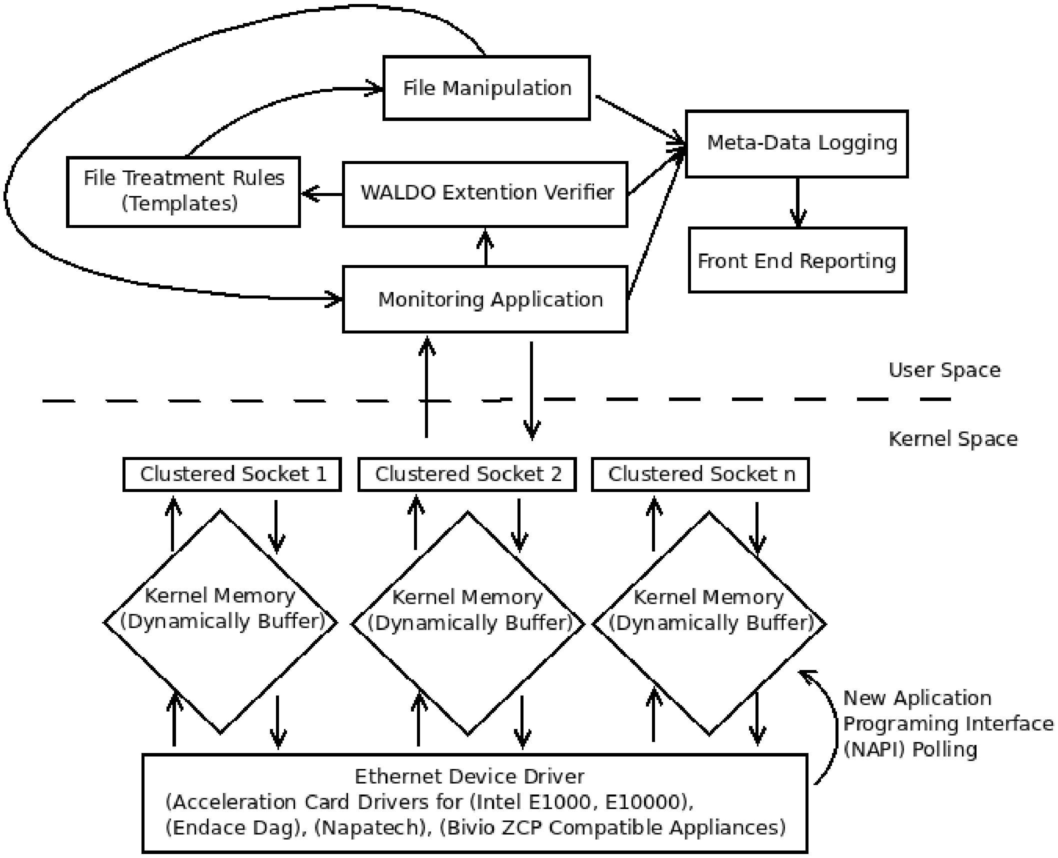
File Manipulation
Meta-Data Logging
File Treatment Rules (Templates)
WALDO Extention Verifier
Front End Reporting
Monitoring Application
User Space
Kernel Space
Clustered Socket 1
Clustered Socket 2
Clustered Socket n
Kernel Memory (Dynamically Buffer)
Kernel Memory (Dynamically Buffer)
Kernel Memory (Dynamically Buffer)
New Aplication Programing Interface (NAPI) Polling
Ethernet Device Driver
(Acceleration Card Drivers for (Intel E1000, E10000),
(Endace Dag), (Napatech), (Bivio ZCP Compatible Appliances)

Figure 2: Protection Without Detection General Information Flow

3.1 PROTECTION ENGINE

The protection engine uses a kernel memory buffer to facilitate the extraction, processing, and re-injection of files. The control and analysis portion of this system resides within Linux user space. These portions of the system interact with kernel level buffers that hold all the packets associated with a network flow. Any file to be analyzed is copied out of the buffer into user space memory, where the byte-level manipulation occurs according to the applicable template that specifies how that file type is to be handled.

After the user space file is re-written, its byte structure is compared to the copy within kernel memory. Wherever the compared bytes differ, they are changed in kernel memory match the manipulated copy. Finally, the buffer is flushed and the packets are allowed to continue on to their destination. This method places a high emphasis on processing *all* files of selected types, thus eliminating the computational time needed for analyzing files for intrusion detection purposes. Because the method strives to clean the file *structure* rather than specific types of known malicious content, it can protect against some undiscovered attacks as well. This capability against zero day threats is a direct result of the system's ability to remove or mask all unnecessary elements within a file type as defined by the trusted template, regardless of whether the particular content is malicious, used, unused, or otherwise.

3.2 PROCESS

Figure 2 details the information flow within the protection without detection engine. As shown, the engine is split into two major areas: the kernel space modules and user space applications. As a packet enters the system, a specific accelerated device driver is chosen depending on the physical network interface being used. Packets are split up depending on the specific session that they are associated with and placed into separate memory buffers. The number of buffers is limited only by the amount of memory that the system has available at the time. When enough packets are available to identify a specific file format, the monitoring application is alerted and extraction begins.

File extraction itself is an additional interaction module that can be used with the standard HTTP-Parser included in most IDS/IPS. Once the monitoring daemon has extracted the complete file, it passes it to the WALDO Extension Verifier which checks the file's legitimacy. For instance, WALDO can determine if a file with a .zip extension is really a Zip file and not a VBscript file (.vb) that has been renamed. Once the actual filetype is determined, it is checked against the file treatment rules. The associated rule(s) and file(s) are then passed off to the file manipulation daemon. Once the manipulation has been completed, a byte comparison is done with the memory buffer and the packets are released to the outgoing network interface.

At multiple points during this process, including file manipulation, extension verification, and extraction/re-injection, the daemons report to a meta-data logging system which creates properly formatted syslog messages as well as standard Barnyard format log messages, and passes them off to the front-end reporting engine.

3.3 EXAMPLE: RENDERING PDF FILES INERT

PDF files are of particular interest because of their demonstrated potential for being carriers of malicious content [9]. In the case of PDF files, the system may be setup to disarm all such document types traveling through the system by extracting them from the stream and processing them per the above procedure. Specifically, any segments of code that reside within an executable container within the PDF markup language are simply removed. This includes code within the /AA, /OpenAction, /ObjStm, /JS, /Launch, RichMedia, JBIG2Decode, JBIG2Decode"compression" and /JavaScript objects. In this example, the roles these documents have for the sample mission do not require them to possess these features, since they are not necessary for the PDF to function as a readable text document. By removing any code within these areas of a PDF, the core content within a document is kept intact while any potentially active threat is removed.

3.4 DELIMITATIONS

In this work we have built our software using commodity hardware and our operating system environment is Linux-based. Linux was chosen because of the functionality that direct kernel modules allow. Although our implementation could be modified to work in a Win32 environment using libraries such as WinPcap, which allow system calls from applications to talk directly to network devices, we find that this solves only the needs of analysis and not reinjection [2]. Furthermore, our system relies heavily on management of kernel memory space in a way that is not currently feasible using the Win32 kernel [10].

The current system uses an implementation of Bloom filters to process each byte of data in each packet payload. This is done before the bit-split string matching process [3, 6]. This method dramatically increases the performance of string matching within the system, but could be faster with the addition of the Aho-Corasick string matching algorithm [7].

4. RESULTS

We have described a new strategy for network defense that emphasizes threat avoidance during the operational phase of an IT system, and explained the inner workings of an implementation of several techniques that embody this strategy. Although still in the development phase, we have completed the core system, demonstrated its functionality, and are now working to quantify its effectiveness. Our starting goal for the project was to achieve inline file manipulation at 1 Gbps half-duplex line speeds. Initial tests using packet captures replayed at line speed indicate that our system meets and exceeds this goal by at least a factor of 10. On a standard 24-core test system, processing performance reaches its limit at 97,000 packets per second, or approximately 10 Gbps. Approximately 60% of the packets processed contain file transfers that our system correctly identified and extracted. Of those, approximately 10% could be manipulated at line rate. These included all PDF files, multiple image formats, office formats, and some streaming media formats such as Flash.

The first-generation system demonstrated a maximum file reconstruction size of 4 MB. It was determined that this limitation was due to the way in which packets were in the memory buffer. After implementing a lockless memory buffer [11], the system is now able to handle single files up to 97MB in size.

5. CONCLUSIONS & FUTURE WORK

In this paper we presented the concept of protection without detection, a revised strategy for more effective network defense. Although still in its initial stages, refinement of the strategy and implementation as a network device has resulted in several positive outcomes. We demonstrated the system’s ability to both extract and manipulate files at line speed. Extraction can be used to feed other applications such as antivirus tools or archival services. Inline manipulation of files based on a hybrid policy-rule approach enables the device to protect a network from malicious payloads by blocking their ingress routes, while still providing a reasonable guarantee that mission-critical data will be delivered, even while under the threat of zero day attacks. In future iterations of this work, we expect to implement our simplified/unified policy-rule hybrid method in conjunction with machine learning techniques in order to automate the process of finding network characteristics to build new policies [4]. We are also evaluating its potential with regard to normalizing network protocols, such as Domain Name Service (DNS), as a way to eliminate the potential for new patterns of protocol misuse.

6. ACKNOWLEDGMENT

Partial funding for this work was received under Air Force Research Laboratory contract number FA8750-10-C-0213. Approved for Public Release; Distribution Unlimited: 88ABW-2012-2407, 23-APR-2012

7. REFERENCES

[1] Gary McGraw, "How Bad is Intrusion Detection?" IT Architect, October 2005 (accessed March 29, 2012), http://www.cigital.com/papers/download/0510sec.ids.pdf.

[2] Fulvio Risso, Loris Degioanni, "An Architecture for High Performance Network analysis," Computers and Communications, IEEE Symposium on, p. 0686, Sixth IEEE Symposium on Computers and Communications (ISCC'01) 2001.

[3] Kun Huang, Dafang Zhang, "A Byte-Filtered String Matching Algorithm for Fast Deep Packet Inspection," Young Computer Scientists, International Conference for, pp. 2073-2078, 2008 The 9th International Conference for Young Computer Scientists, 2008.

[4] Peter Teufl, Udo Payer, Michael Amling, Martin Godec, Stefan Ruff, Gerhard Scheikl, Gerno Walzl, "InFeCT – Network Traffic Classification," International Conference on Networking, pp. 439-444, Seventh International Conference on Networking (ICN 2008), 2008.

[5] Yuebin Bai, Hidetsun Kobayash, "New String Matching Technology for Network Security," Advanced Information Networking and Applications, International Conference on, p. 198, 17th International Conference on Advanced Information Networking and Applications (AINA'03), 2003.

[6] Joshua Broadway, Benjamin Turnbull, Jill Slay, "Improving the Analysis of Lawfully Intercepted Network Packet Data Captured for Forensic Analysis," Availability , Reliability and Security, International Conference on, pp. 1361-1368, 2008 Third International Conference on Availability, Reliability and Security, 2008.

[7] Derek Pao, Wei Lin, and Bin Liu. 2010. A memory-efficient pipelined implementation of the aho-corasick string-matching algorithm. *ACM Trans. Archit. Code Optim.* 7, 2, Article 10 (October 2010), 27 pages.

[8] Fang Yuan, Bo Liu, and Ge Yu. 2005. A study on information extraction from PDF files. In *Proceedings of the 4th international conference on Advances in Machine Learning and Cybernetics* (ICMLC'05), Daniel S. Yeung, Zhi-Qiang Liu, Xi-Zhao Wang, and Hong Yan (Eds.). Springer-Verlag, Berlin, Heidelberg, 258-267.

[9] Didier Stevens. 2011. Malicious PDF Documents Explained. *IEEE Security and Privacy* 9, 1 (January 2011).

[10] Francesco Fusco and Luca Deri. 2010. High speed network traffic analysis with commodity multi-core systems. In *Proceedings of the 10th annual conference on Internet measurement* (IMC '10). ACM, New York, NY USA.

[11] Robert W. Wisniewski and Bryan Rosenburg. 2003. Efficient, Unified, and Scalable Performance Monitoring for Multiprocessor Operating Systems. In *Proceedings of the 2003 ACM/IEEE conference on Supercomputing* (SC '03). ACM, New York, NY, USA.

[12] Yang Xiang and Wanlei Zhou. 2008. Using Multi-Core Processors to Support Network Security Applications. In *Proceedings of the 2008 12th IEEE International Workshop on Future Trends of Distributed Computing Systems* (FTDCS '08). IEEE Computer Society, Washington, DC, USA, 213-218.

[13] Yaxuan Qi, Zongwei Zhou, Baohua Yang, Fei He, Yibo Xue, and Jun Li. 2008. Towards effective network algorithms on multi-core network processors. In *Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems* (ANCS '08). ACM, New York, NY, USA.

[14] Junchang Wang, Haipeng Cheng, Bei Hua, and Xinan Tang. 2009. Practice of parallelizing network applications on multi-core architectures. In *Proceedings of the 23rd international conference on Supercomputing* (ICS '09). ACM, New York, NY, USA, 204-213.

[15] Benjamin Wun, Patrick Crowley, and Arun Raghunth. 2009. Parallelization of Snort on a multi-core platform. In *Proceedings of the 5th ACM/IEEE Symposium on Architectures for Networking and Communications Systems* (ANCS '09). ACM, New York, NY, USA, 173-174.