Mediated Overlay Services (MOSES): Network Security as a Composable Service

Stelios Sidiroglou Angelos Stavrou Angelos D. Keromytis
Department of Computer Science
Columbia University, New York, NY
{*stelios,angel,angelos*}@cs.columbia.edu

Abstract

In recent years, organizations have been shifting focus to their core business competencies, and reducing total cost of ownership (TCO) associated with training and management of their IT infrastructure. In the same motif, organizations are establishing security and survivability frameworks as an integral part of their business strategy so as to provide an acceptable quality-of-service for their clients and employees. However, the current paradigm of outsourced managed security service providers (MSSPs) is often difficult to transition to, offers little control to the organization, does not allow "best of breed" composition, and risks vendor lock-in due to the complexity of migrating to a different MSSP.

We present MOSES (Mediated Overlay Services), an architecture for composing network security services such as anti-spam, antivirus, automated vulnerability detection and mitigation, and filtering. MOSES is roughly modeled on the web services framework. In addition to ease-of-deployment, MOSES allows for economies of scale and a reduction to the total cost of ownership. In this paper, we discuss our motivation and high-level view of such an architecture. We highlight the advantages, illuminate potential drawbacks, and discuss a broad research agenda toward realizing this vision.

1 Introduction

As organizations are increasingly focusing on their core competencies and business processes, they have been looking for ways to outsource their IT operations and management. As a result, a new class of application service providers has emerged, offering a diverse set of outsourcing services: from help-desk support, to server-farm management and hosting, to IP telephony, to providing web-based application suite, these providers leverage cheap and fast connectivity to achieve economies of scale combined with specialization to critical but relatively narrow aspects of an enterprise's IT needs. With the exception of a handful of large application service providers (such as Google and IBM), the market for service providers remains highly fragmented, with individual firms striving to provide very specialized, "best of breed" software and services in their respective niches.

In the same motif, organizations are establishing security and survivability frameworks as an integral part of their business strategy so as to provide an acceptable quality-of-service for their clients and employees. However, the increasing dependence on the Internet as a means of both gluing together the outsourced IT components and for interacting with customers means that there is a tension between openness and connectivity on the one hand, and the security needs of the organization on the other. Specifically, a key function of the IT management and support stuff nowadays revolves around handling malicious behavior from both inside and outside of the enterprise network in the form of spam, virus and worms, denial of service (DoS) attacks, insider malfeasance, and so on. As a result, a number of Managed Security Service Providers (MSSPs) have appeared, striving to offer "one-stop shop" security services. However, this approach can leave an enterprise vulnerable, since very few MSSPs are big enough to have highly specialized expertise in all the possible threat domains. Unfortunately, the all-in-one nature of current MSSPs, combined with the sensitivity of the provided service, act as a deterrent to changing providers, leading to vendor lock-in.

We argue that a better model for outsourced security would be to emulate the "best of breed" approach largely followed by web-based application service providers. That is, security services should be provided as modules that can be composed in a flexible manner to meet the needs of an organization. Such modules may be under the direct control of the organization (*e.g.,* in the form of

stand-alone devices or software that runs on the organization's servers), may be offered as a remotely managed appliance, or as a software service running off-premises. This approach allows organizations to select exactly the type of services they need, using the best provider for each service. By standardizing the way such components interact with each other, service replacement should be a much easier task. For small MSSPs, such an architecture is more attractive since it becomes easier for customers to employ their services *vs.* using a large integrated MSSP. Even large MSSPs can benefit, by replacing their offerings in areas in which they do not have deep expertise with outside partners that do, thus providing a more attractive overall service.

We propose *MOSES*, an overlay-based architecture that facilitates the deployment and composition of a plethora of security services. MOSES uses a flow-based model of enterprise data, and allows services to act as filters on those flows. The elements performing the filtering can reside entirely on the enterprise's premises, or be entirely outsourced, or a combination of the two. Is it also possible to selectively split data flows, as well as (selectively) join different flows. This architecture provides fertile ground for a new modus operandi of security services. Examples of these services are transparent network-wide filtering, attack inference, service availability (resilience), worm and virus detection, and large-scale behavior analysis (users, traffic, *etc.*). What is of importance here is that our architecture enables new types of services, by allowing for a natural aggregation of traffic and routing decisions above the network layer. From an economic perspective, one can quickly realize that economies of scale can be easily achieved by each SSP, thus procuring the ability to provide services at a reduced cost. From an architectural viewpoint, the use of a secure overlay like SOS [7] facilitates a proactive approach to dealing with survivability and service availability by using a amalgamation of secure overlay tunnels, policy-based routing, and filtering.

Ultimately, the goal of MOSES is to provide a base architecture that facilitates the outsourcing of services in general by fostering an open, flexible services framework. However, this approach also introduces several risks and disadvantages relative to an "in-house" approach. First, SSPs may fail to perform their duties (or even act maliciously, or get compromised). Second, the use of an overlay may impact overall system performance, since the data may traverse the wide-area network several times as it flows between SSPs and the enterprise. Finally, unless the whole enterprise is virtualized and outsourced (with computation and storage decoupled from the organization), MOSES will not be able to deal with insider risks and attacks.

The rest of this paper describes and motivates MOSES's architecture and design principles in detail, including a description of sample proposed services and scenarios, as well as ways of mitigating the above-mentioned risks.

2 MOSES System Architecture

Current approaches to securing an enterprise typically require acquiring and managing logically distinct components, which an organization needs to compile into service entities. For example, enterprises routinely employ different products, often from different vendors, for their firewall, VPN, intrusion detection, intrusion prevention, patch management, spam, anti-virus, web security, backup, insider misbehavior detection, and other needs. It is not uncommon to simultaneously use several different products that address the same threats, *e.g.,* worm detectors or spam blockers. While this approach might be adequate for large enterprises with numerous IT staff, it renders the cost of ownership prohibitively expensive for smaller organizations. The problem is further exacerbated when one examines the cost beyond the initial cost of the infrastructure, in particular, the cost of maintenance and training. Thus, it would seem that securing an enterprise is also amenable to outsourcing, within some constraints.

2.1 MOSES Overview

The goal of MOSES is to provide a distributed, secure and robust mechanism to transparently connect service providers with their respective customers. For this purpose, MOSES provides a level of indirection between its customers' infrastructure, security service providers (SSPs), and the Internet. It employs a distributed policy-based overlay network of access points that acts as a service mediator between the different sites that MOSES connects, as shown in Figure 1. Each of the entities connected to MOSES can take the role of a service provider, a service consumer, or both, depending on policy agreements. For example, an organization may wish to outsource the spam detection of its incoming mail to another site that provides a spam filtering service. MOSES will intercept all mail requests to the organization and reroute them through the spam-detection site and back to the customer in a completely transparent fashion. Several services can be chained in this way, with data flowing through the overlay before reaching the enterprise; similar processing may occur in the outgoing direction, typically

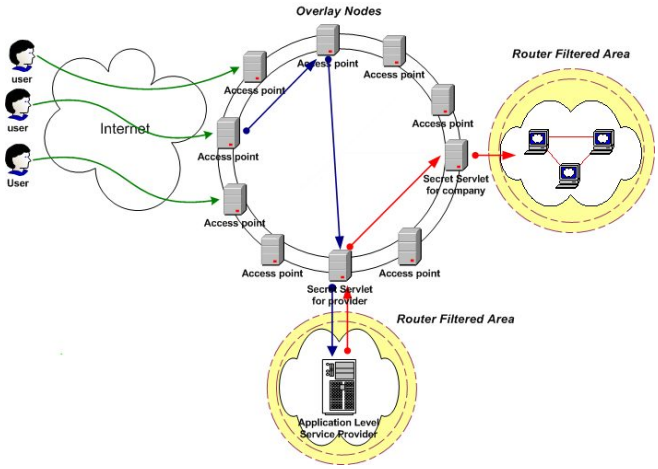

Figure 1: **Moses architectural overview: The overlay network acts as a policy-enforcement mediator, re-routing all traffic between users (or enterprise components) and the enterprise through requested services using the information stored in the routing policy database. Each site connected to MOSES can be thought of as service provider, a service consumer or both.**

for connections initiated by the enterprise. If two enterprises A and B communicate through MOSES, one can view the data flow as two pipes (A's outgoing and B's incoming processing pipeline, or vice versa) connected somewhere inside the overlay. At any step in the pipeline, data may be duplicated (and sent to multiple destinations or paths, *e.g.,* to a backup site), merged (*e.g.,* combining and pruning the streams received from two different virus detection SSPs that are used in parallel, instead of in series), or dropped (if deemed malicious). MOSES (and the SSPs) must provide enough visibility of the processing pipeline to an enterprise that there is confidence in its operation and results.

MOSES harnesses the distributed nature of overlay networks to strategically deploy overlay nodes close to the customers' intranets utilizing different ISPs. This provides the ability to opportunistically choose the best route between components independently of the underlying network topology thus achieving high availability [3, 5] and, occasionally, lower latency [13].

This high-level description raises several questions, posed and partially answered below. Our goal is not to provide a complete architecture, but to motivate research in this direction and outline a broad research vision (complete with challenges and open problems) that will stimulate further work.

How do entities discover each other? Initially, we envision a simple directory service that lists the various SSPs and the services they provide, along with the traffic types they can consume and produce. System managers can peruse the directory and select an SSP with whom to negotiate a service-level agreement. Reputation systems and auctioning may be used as part of the selection process. Eventually, we envision a system where applications can discover entities and incorporate them in their processing pipeline in a fully automated fashion, reflecting the needs of specific transactions/interactions. Additional information that may come into play during selection includes network topology (for performance reasons, as we shall see shortly), processing capacity guarantees, and others. This requires a language for specifying and analyzing MOSES-provisioned security services, akin to the Web Services Description Language (WSDL) used in Web Services.

How does routing work in MOSES? Once two parties agree on a service-level agreement, a routing policy is generated and "pushed" in the overlay. This policy describes what type of traffic the SSP should be expecting to receive (and from whom), what type of traffic it is expected to produce, and what the next hop in the overlay should be. The next hop may be the enterprise itself, or it may be another SSP in MOSES. This policy can either be stored in a protected centralized database or it can be distributed among the overlay nodes creating a distributed policy database. We believe that the latter is more scalable, secure and resilient, as shown in previous research [7, 15]. One approach to routing is to treat MOSES simply as an information layer that SSPs use to determine

what the next hop in an enterprise's pipeline; traffic then flows directly between SSPs (and the enterprise). One potential problem for the enterprise is that each SSP knows the previous and next SSP in its pipeline. Thus, a second approach to routing is for MOSES to effectively act as an anonymization and complete routing layer, passing data flows from one SSP to the next without revealing the respective identities. While this approach maintains the secrecy of the pipeline elements, it may increase the network latency due to the additional hops involved. An interesting approach is the use of the overlay to control routing decisions, while using the underlying network infrastructure for direct data communication.

How does traffic enter MOSES? Pre-established pipelines between known enterprise elements need not interact with the rest of MOSES. However, users (or other enterprises) either need to be aware of MOSES or somehow must be redirected to an entry point of the enterprise's pipeline. There are several ways to achieve this. For example, an enterprise may be accessible through DNS or IP-provisioned Anycast, which will point to a MOSES overlay node that runs an application-level proxy that will tag and route traffic to the appropriate pipeline. Another approach, especially applicable to web-based interactions, is to redirect a client's browser or similar client to that entry point. Because of our strong requirements for system availability, we have made SOS [7] the front-line element of MOSES. We give a high-level description of SOS in Section 2.2.

How does an enterprise protect its communications from malicious parties, including malevolent SSPs? Trusting a third party (an SSP, or MOSES itself) to route enterprise traffic introduces an element of risk. Currently, Internet Service Providers are trusted to to route traffic to and from external entities. Small and medium-size enterprises are usually connected to the Internet via a single ISP, who controls their external connectivity. There is also some dependency on external infrastructure (*e.g.,* DNS) but, generally, data flows directly between customers or external partners and the enterprise without intermediate processing (although email, with its store-and-forward architecture, is a notable exception). Even if the third party routes data flows correctly, there is no guarantee that the processing performed will be correct, benevolent, or complete. Finally, there is the risk of data theft, especially in the financial services domain, as private information flows through multiple intermediaries; malicious employees and system compromises are some of the obvious risks there.

Although there is no all-encompassing solution to this problem, we believe that a combination of automated active auditing, redundancy, and proper use of encryption can mitigate some of these risks. For example, MOSES allows enterprises to inject traffic at any point in their pipeline; doing so and comparing the received results allows an enterprise to monitor the quality of service received, and to identify some types of misbehavior. The use of encryption and, more interestingly, data anonymization can also help mitigate some of the risks of information disclosure to third parties. To aid in this, MOSES nodes can perform some nominal processing on the data streams the view, on behalf of the enterprise. Such processing is intended to be relatively lightweight so as not to overload the infrastructure. Interesting future work is the development of a lightweight processing infrastructure, perhaps based upon prior work in active networking [2], and of algorithms and techniques for anonymizing various types of data flows for processing by SSPs.

What are the performance implications of such traffic routing? Another potential problem is the network overhead of the re-routing operation. We claim that network latency although substantial for services that require relatively fast server response like web browsing, it is within acceptable limits; our previous work has demonstrated an overall increase in latency by a factor of 2 to 3 [8]. The use of multi-path routing and traffic spreading [13] can further reduce the overhead, although the use of multiple SSPs can mitigate these gains. An interesting direction for future work will be the reconciliation of MOSES policy-based routing with performance considerations; such concerns can also affect the selection of SSPs, since topologically clustered components will result in lower overall latency. On the other hand, a large class of interesting services like email and backup are not quite as time-sensitive.

What types of services can (and cannot) fit into this framework? We envision a variety of services as inherently MOSES-friendly, and describe a few of them in Section 3. Fundamentally, operations on data that are stateless (or almost so) should be straightforward to implement in MOSES. Services that require bidirectional interaction between components may be more difficult to efficiently model in MOSES, although we have not yet come across a good example. Determining the architectural limitations and possible extensions to MOSES is part of our future plans of work.

We believe that security and resilience is of vital importance for our system since it will be the connection infrastructure for a variety of services. Since critical policy and routing information will be stored and maintained from MOSES, we decided to use an enhanced version of

Secure Overlay Services (SOS) [7] as a component of the MOSES design. For applications that assume a more secure environment of operations, *e.g.,* within the intranet of a single organization, a simpler approach would suffice. The original design of SOS allowed for the coexistence of multiple services using the same overlay. However, composition or encapsulation of services were not supported. Additionally, SOS was limited to authentication policy enforcement, whereas MOSES tackles more sophisticated policy requirements that affect the routing of packets inside the overlay. Before we continue, we include a brief description of the fundamental aspects of SOS; especially the construction of the overlay network explaining the enhancements made to accommodate MOSES.

2.2 Secure Overlay Services for MOSES

Fundamentally, the goal of SOS (and of many indirection-based DDoS defense system) is to selectively drop or rate limit unauthorized traffic while routing the good traffic to a selected service provider. At a very basic level, SOS provides the functionality of a firewall "deep" enough in the network such that the access link to the target does not become congested[1]. This distributed firewall [6] may perform access control by using protocols such as IPsec or TLS, or by relying on authentication and authorization services from the system being protected, or by using techniques such as Graphical Turing Tests [2] [18]. Once admitted, traffic is routed to a secret location, which can be target itself [14] or a node that is allowed to contact the target (called "secret servlet" in SOS [7]), with all other traffic being filtered [7, 8], as shown in Figure 2. The secret forwarder can vary over time, and is different for each site protected by the system; part of the functionality built in SOS concerns itself with maintaining and propagating this information to other overlay nodes. Otherwise, we assume that the identity of the protected server and all indirection nodes is publicly known or easily determined by an attacker.

In the original SOS, each overlay node maintains a table that stores the identities of m other overlay nodes. The Chord algorithm [16] routes packets around the overlay "circle", progressively getting closer to the desired overlay node, with $O(m)$ overlay nodes visited. Chord provides a robust and reliable, while relatively unpredictable for an adversary means of routing packets from an over-

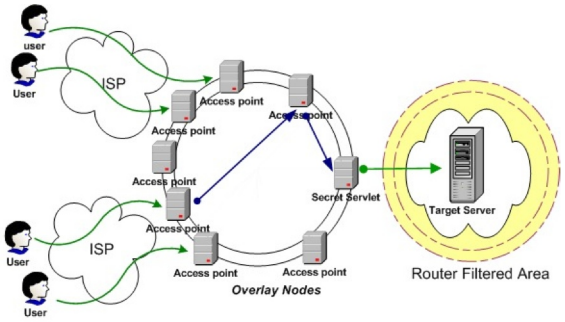

Figure 2: **SOS architecture**

lay access point to one of the nodes that knows the identify of the secret servlet for a target. Unfortunately, this also increases the communication latency, since traffic to the target must be redirected several times across the Internet. Instead, a two-level redirection approach can be used, in which traffic from any access point can be sent directly to the secret servlet, dispensing with Chord as the intra-overlay routing mechanism. This approach can reduce the latency by an order of magnitude, to about a little over twice the latency of a direct communication between a user and the target site [8]. By using traffic spreading across all overlay nodes as shown in Figure 3, it is possible to further reduce this overhead while also increasing the resiliency of the system against DoS attacks [13].

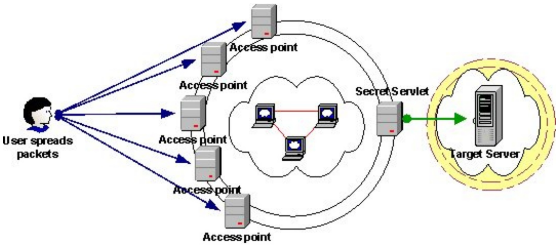

Figure 3: **Spread-spectrum SOS**

The same overall architecture is retained in MOSES, with a few changes. As in SOS, MOSES nodes implement the full SOS functionality and can thus simultaneously act as access points and secret servlets; in addition, these nodes implement MOSES routing. SSPs themselves need not be treated as part of the overlay; instead, they become a SOS "target", and all traffic to them is routed through MOSES. Thus, between each hop in the pipeline, there are two MOSES/SOS nodes: an access point and a secret servlet. (In reality, when using traffic spreading, all MOSES nodes are used as access points.) However, when two components reside within the enterprise or when the interconnection policy dictates so, traf-

[1] In terms of network topology, this typically means the first or second-level router in the hosting Internet Service Provider's Point-of-Presence (POP).

[2] Although the simple GTTs can be easily broken [9], more advanced versions are as of yet outside the reach of computer vision and semantic understanding.

fic can completely bypass MOSES and instead flow from one SSP to the next. Furthermore, when network reservations can be guaranteed [17], traffic can flow between SSPs and the overlay in a single hop, without need for the two-hop indirection. The ability to use SOS on a pay-per-use basis [12] also allows for flexible charging models between users and organizations, or even among SSPs and enterprises, especially when MOSES pipelines are automatically established and torn down on demand for short periods of time.

3 Example MOSES Services

Since the goal of designing MOSES and one of its key functions is the provision of security-related services, we now turn our attention to currently available and envisioned services for MOSES. Our goal is to give some examples of such services, rather than an exhaustive description. We omit some of the obvious services, such as firewall filtering and VPN provisioning, since such capabilities are inherent in the core functionality of MOSES.

Worm Vaccine and Email worm detection The Network Worm Vaccine Architecture [11] is an ideal example for the types of services that would be deployed with MOSES. The WormVaccine architecture presents a first-reaction mechanism that seeks to automatically patch vulnerable software. The system employs a collection of sensors that detect and capture potential worm infection vectors. These vectors are automatically tested against appropriately-instrumented sandboxed instances of a targeted application (*e.g.,* the web server) for any exploited software weakness. Based on a number of heuristics, the WormVaccine software automatically generates patches that can protect against certain classes of attacks and tests the resistance of the patched application against the infection vector. Also part of the WormVaccine architecture is an extension that deals with email worms: potential worms are forwarded to a sandboxed environment where a host-based intrusion detection system [4] identifies infection vectors based on the behavior of attachments as they are automatically opened by the system [10].

Under MOSES, applications used across the overlay would be registered with a worm/email-vaccine provider. The provider would, in turn, be responsible for patching and propagating protected instances of the application to registered users of the service using the overlay. The detection mechanism would employ distributed honeypots on the overlay [1], in addition to sensors at the organization level.

WebSOS The ability to provide guaranteed access to a web service, especially under adverse network conditions is fairly ambitious under the current Internet infrastructure. WebSOS [8] is an overlay-based architecture that provides such access to a web server that is targeted by a denial of service attack. We envision the use of an architecture like WebSOS in conjunction with network-wide policies, allowing for policy-driven behavior at the network, organization or user level. For example, when traffic levels exceed a threshold values, access to certain web services would become available only through WebSOS. Inherent to the WebSOS architecture is a distributed intrusion detection component that deals with the issue of identifying misbehaving traffic.

Spam Detection Outsourcing logically coherent services to multiple vendors is an integral part of the MOSES architecture. An ideal example for this type of service is spam detection. One can easily envision outsourcing spam detection to open and proprietary detection engines and correlating the results into a more coherent and potentially accurate set of results. Both SSPs and organizations can benefit from using a larger user base to train their respective detection engines, since they can diversify the signature base across organizations. For example, a detection engine can take advantage of early detection of spam originating at a certain locale and provide filters proactively in other locations. Furthermore, novel spam architectures such as SpamWatch [19] may be deployed to take advantage of collaborative, peer-to-peer and content similarity based detection.

Distributed Intrusion Detection Systems Although not a panacea to problems involving intrusion detection systems, an overlay-based system can provide a number of vantage points. Briefly, the framework for collaboratively collecting network-wide traffic data across the underlying heterogeneous network becomes readily available, greatly facilitating the data-mining of this information. Furthermore, with MOSES, intrusion detection systems would be able to easily access both incoming and outgoing network traffic given the desire to do so.

Non-Security related In this paper, we focus primarily on the management and deployment of security services. However, we also anticipate the deployment of a number of non-security services. Examples of such services are backup and software update. For the backup service, one can easily envision employing a location-driven overlay system to offer secure and resilient large-file transfers to geographically distinct areas providing a natural protection mechanism against natural and man-made disasters.

The software update service builds upon the security infrastructure of a secure overlay system like SOS to provide a highly robust update mechanism. MOSES provides the underlying security mechanism and the ability to manage content dissemination efficiently.

4 Conclusions

We presented MOSES, an architecture for composing network security services such as anti-spam, anti-virus, automated vulnerability detection and mitigation, and filtering. MOSES is roughly modeled on the web services framework. We gave a high-level view of the type of services MOSES would offer, its key elements, and identified areas where additional work is needed and opportunities for future research exist.

References

[1] K. Anagnostakis, S. Sidiroglou, P. Akritidis, K. Xinidis, E. Markatos, and A. D. Keromytis. Detecting Targetted Attacks Using Shadow Honeypots. In *Proceedings of the* 14^{th} *USENIX Security Symposium*, pages 129–144, August 2005.

[2] K. G. Anagnostakis, S. Ioannidis, S. Miltchev, and J. M. Smith. Practical Network Applications on a Lightweight Active Management Environment. In *Proceedings of the* 3^{rd} *International Working Conference on Active Networks*, pages 101–115, October 2001.

[3] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. N. Rao. Improving Web Availability for Clients with MONET. In *Proceedings of the* 2^{nd} *Symposium on Networked Systems Design and Implementation (NSDI)*, May 2005.

[4] F. Apap, A. Honig, S. Hershkop, E. Eskin, and S. J. Stolfo. Detecting Malicious Software by Monitoring Anomalous Windows Registry Accesses. In *Proceedings of the* 5^{th} *Symposium on Recent Advances in Intrusion Detection (RAID)*, October 2002.

[5] K. P. Gummadi, H. V. Madhyastha, S. D. Gribble, H. M. Levy, and D. Wetherall. Improving the Reliability of Internet Paths with One-hop Source Routing. In *Proceedings of the* 6^{th} *Symposium on Operating Systems Design & Implementation (OSDI)*, December 2004.

[6] S. Ioannidis, A. Keromytis, S. Bellovin, and J. Smith. Implementing a Distributed Firewall. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 190–199, November 2000.

[7] A. D. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure Overlay Services. In *Proceedings of ACM SIGCOMM*, pages 61–72, August 2002.

[8] W. G. Morein, A. Stavrou, D. L. Cook, A. D. Keromytis, V. Misra, and D. Rubenstein. Using Graphic Turing Tests to Counter Automated DDoS Attacks Against Web Servers. In *Proceedings of the* 10^{th} *ACM Conference on Computer and Communications Security (CCS)*, pages 8–19, October 2003.

[9] G. Mori and J. Malik. Recognizing Objects in Adversarial Clutter: Breaking a Visual CAPTCHA. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, June 2003.

[10] S. Sidiroglou, J. Ioannidis, A. D. Keromytis, and S. J. Stolfo. An Email Worm Vaccine Architecture. In *Proceedings of the* 1^{st} *Information Security Practice and Experience Conference (ISPEC)*, pages 97–108, April 2005.

[11] S. Sidiroglou and A. D. Keromytis. A Network Worm Vaccine Architecture. In *Proceedings of the IEEE Workshop on Enterprise Technologies: Infrastructure for Collaborative Enterprises (WETICE), Workshop on Enterprise Security*, pages 220–225, June 2003.

[12] A. Stavrou, J. Ioannidis, A. D. Keromytis, V. Misra, and D. Rubenstein. A Pay-per-Use DoS Protection Mechanism For The Web. In *Proceedings of the Applied Cryptography and Network Security (ACNS) Conference*, pages 120–134, June 2004.

[13] A. Stavrou and A. Keromytis. Countering DoS Attacks With Stateless Multipath Overlays. In *Proceedings of the* 12^{th} *ACM Conference on Computer and Communications Security (CCS)*, pages 249–259, November 2005.

[14] A. Stavrou, A. D. Keromytis, J. Nieh, V. Misra, and D. Rubenstein. MOVE: An End-to-End Solution To Network Denial of Service. In *Proceedings of the ISOC Symposium on Network and Distributed System Security (SNDSS)*, pages 81–96, February 2005.

[15] I. Stoica, D. Adkins, S. Zhuang, and S. Surana. Internet Indirection Infrastructure. In *Proceedings of ACM SIGCOMM*, pages 73–86, August 2002.

[16] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-To-Peer Lookup Service for Internet Application. In *Proceedings of ACM SIGCOMM*, August 2001.

[17] D. M. Turner, V. Prevelakis, and A. D. Keromytis. The Bandwidth Exchange Architecture. In *Proceedings of the* 10^{th} *IEEE Symposium on Computers and Communications (ISCC)*, pages 939–944, June 2005.

[18] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. CAPTCHA: Using Hard AI Problems For Security. In *Proceedings of EUROCRYPT*, May 2003.

[19] F. Zhou, L. Zhuang, B. Y. Zhao, L. Huang, A. D. Joseph, and J. Kubiatowicz. Approximate Object Location and Spam Filtering on Peer-to-peer Systems. In *Proceedings of ACM Middleware*, June 2003.