

Defending Against Next Generation through Network/Endpoint Collaboration and Interaction

Spiros Antonatos*, Michael Locasto⁺, Stelios Sidirolou⁺

Angelos D. Keromytis⁺ and Evangelos Markatos*

* *Foundation for Research and
Technology Hellas (FORTH-ICS)*
Greece

⁺ *Department of Computer Science*
Columbia University
USA

Abstract

The nature of cyberattacks has changed dramatically over the past few years. Encryption, polymorphism, metamorphism of attack vectors and targeted attacks based on hitlists have rendered traditional defense mechanisms unable to react on such threats. To overcome the current limitations of detection mechanisms, we argue that large-scale collaboration among end-hosts is a viable solution. In this paper we discuss two host-based approaches: application communities, a collaboration of members who run the same applications with different instrumentation level trying to identify previously unknown attacks, and Honey@home, an architecture that enables the gathering of information from distributed unused IP address space.

1 Problem Statement

Over the past few years we have seen the use of *Internet worms*, *i.e.*, malicious self-replicating programs, as a mechanism to rapidly invade and compromise large numbers of remote computers [SPW02]. Although the first worms released on the Internet were large-scale, easy-to-spot massive security incidents [MSB02, MPS+03, SM04, BCJ+05b], also

known as *flash worms* [SMPW04], it is currently envisioned (and we see already see signs, in the wild) that future worms will be increasingly difficult to detect, and will be known as *stealth worms*. This may be partly because the motives of early worm developers are thought to have been centered around self-gratification brought by the achievement of compromising large numbers of remote computers, while the motives of recent worm and malware developers have progressed to more mundane (and sinister) financial and political gains. Therefore, although recent attackers still want to be able to control a large number of compromised computers, they prefer to compromise these computers as quietly as possible, over a longer period of time, so as to impede detection by current defense mechanisms. To achieve stealthy behavior, these attackers have started using, or at least have the capacity to use, a wide variety of mechanisms that will make their worms more difficult to detect. Such mechanisms might include:

·**Encryption:** Attackers may communicate with the potential victim using a secure (encrypted) connection, making it difficult for network-based Intrusion Detection Systems [Roe99, XCA +06] to spot their attempted attack.

·**Metamorphism:** The body of worms usually contains some initial code that will be executed when the worm invades the victim computer. Metamorphism obfuscates this code by adding various instructions to it, and/or by substituting blocks of instructions with equivalent blocks of other instructions [SFOI]. In this way, two "copies" of the worm would appear to be completely different from each other, confusing worm detection systems that depend on all copies of a worm being practically identical [SEVS04, KK04, AAM05].

·**Polymorphism:** Polymorphic approaches obfuscate the worm's body by encoding it and prepending a decoder. When propagating, the worm mutates its body so that two "copies" of the worm would look completely different from each other (modulo the body of the encoder) [Szo05, DUMU03, K201]. Much like metamorphic approaches, polymorphic systems confuse worm detection systems.

·**Hit Lists:** The first versions of recent worms selected their victims pseudo-randomly, *i.e.*, by generating a random IP address in the range *0.0.0.0* to *255.255.255.255*. It has been proposed however, that worms may be more effective if they first create a hit-list of all vulnerable computers and then attack only computers in that hit-list [SPW02,

AAMA05]. This hit-list may even be filtered to exclude honeypots¹¹. Armed with a hit-list, a worm is able to compromise a number of vulnerable computers, while generating the minimum amount of traffic possible, limiting the effectiveness of defense mechanisms that detect visible traffic anomalies.

·**Hybrid Worms:** Traditionally, worms have exploited vulnerabilities in applications and services open to Internet traffic. However, as more computers are located behind firewalls and NATs, they are theoretically protected from such types of attacks. Unfortunately, worm developers may exploit several different invasion paths including, infected email attachments, infected files shared through peer-to-peer (P2P) networks, and infected files accessed through locally shared disks [KE03].

·**Defense Mapping:** Many of the proposed (and deployed) techniques for detecting and countering new attacks use honeypots as early-warning systems [Spi03, DQG+04, YBP04, CBMM04, BCJ+05a, RMT05, MVSOI]. However, recent work has shown that attackers can exploit certain features and aspects of honeypot behavior to identify and avoid such detectors [BFV05, SII05, RMT06]. Combined with hit-lists, this can render worms (especially slow-spreading ones) and other automated attacks virtually undetectable.

·**Client-side Attacks:** In the past few years (2005-2006) we have seen an increase in the use of zero-day attacks aimed at client software (especially browsers, but also various types of document viewers such as Microsoft Word, Excel and PowerPoint, and Adobe Acrobat). Other than stand-alone, host-based intrusion detection/prevention mechanisms (such as virus scanners), very little has been done in hardening vulnerable client systems.

1.1 Impact of failing to solve the problem

Compromised computers can be used to cause harm to third parties or even to cause harm to their traditional owners.

·**Attacks to third parties:** Recent worm writers organize

¹¹A honeypot is a computer waiting to be attacked. Once attacked, the honeypot records as much information as possible so that the administrators will be able to characterize the attack and possibly generate a signature for it.

compromised computers into botnets, *i.e.*, armies of hosts that are primarily used for malicious acts, including launching of Denial of Service (DoS) attacks, blackmailing, sending of SPAM mail, click fraud, theft of intellectual property, and even identity theft. One would envision that botnets in the future could be used for political warfare purposes as well.

Attacks to the owners of compromised computers: A compromised computer can be used to steal private data and facilitate identity theft. Unfortunately, once ordinary users start to realize the dangers of a compromised computer, they will probably get increasingly less inclined to trust their computers for financial transactions or private communications. This will probably impede the adoption of an information society and may eventually reduce its overall spread and impact.

2 Research Directions

Over the last five years significant research has been conducted in the area of detection and containment of cyber-attacks. Indeed, we believe that we have currently reached the point where it is possible to readily detect one particular class of worms: rapidly spreading and massively parallel flash worms. However, it is unclear we have the technical knowledge or the deployed mechanisms in order to detect and contain *stealth attacks*. Using a combination of the techniques described earlier, such attacks can become invisible (or at least very difficult to detect) to network-based defenses.

Our view is that such attacks can only be detected via large-scale collaboration among end-hosts: by exchanging and correlating relevant information, it is possible to identify stealthy attacks, and to take appropriate measures to defend against them, or at least quarantine those nodes that appear to have been compromised. Specifically, we believe that it is increasingly important to include home and small business computers in the attack-detection process. These computers are increasingly becoming the primary targets of most attackers. Therefore, including them in the worm- (or, more generally, attack-) detection process will increase the chances of attack detection. Exemplifying a large range of access patterns and a large range of applications, these computers typically tend to have more representative configurations than the traditional honeypots currently being used in worm detection. Furthermore, ordinary computers being used by their regular owners are more difficult to be categorized as

honeypots and avoided by future attacks. The inclusion, however, of home computers in the detection process, should (1) guarantee the safety of the end computer and (2) the minimum possible intrusion in the ordinary use of the computer. Towards this direction, we propose two systems: Honey@home and Application Communities. We give a high-level description of both systems in the next two sections, both as concrete examples of collaborative defense mechanisms and to motivate further work in this direction.

On the other hand, we are not completely discounting network-based defenses: rather, we believe that such defenses must be integrated with end-host defenses. In the past, network and end-host security were viewed as two distinct areas that were meant to complement each other but kept separate. While this allowed for a clean separation between the respective security mechanisms, it also meant that the potential of both was stunted. Furthermore, by keeping them isolated, it was (and is) impossible to exploit scale for defensive purposes. Exploiting scale is something that attackers have learned to do well, as evidenced by such phenomena as distributed denial of service attacks, self-propagating worms, and botnets.

The industry is beginning to follow such an approach, albeit in a fragmented, *ad hoc* fashion. For example, several enterprises exchange alert and IDS logs through sites such as DShield.org; anti-virus vendors with extensive presence on the desktop are correlating information about application behavior from thousands of hosts; network security and monitoring companies perform similar correlation using network traces and distributed black-holes (honeypots). To the extent that such approaches are being explored, they seem largely confined to the realm of information gathering. This also largely seems to be the situation with the US Department of Defense and the various intelligence agencies. For example, DARPA is currently funding the Application Communities effort, which seeks to leverage large software monocultures to distribute the task of attack monitoring - again, an approach confined to the end-host. Previous work (notably in the DARPA OASIS program) looked into the space of reactive security, but only considered small-scale environments. Arguably, we need to extend the reach of our collaboration-based mechanisms to counter such pervasive threats as DDoS and botnets.

Thus, we argue that it is important to transition into a network architecture design where networks and end-hosts, in various combinations, can elect to collaborate and coordinate their actions and reactions to better protect themselves (and, by implication, the network at large). There are several research issues arising in such an environment, including:

- What problems are best addressed through a collaborative approach;
- New mechanisms at all levels of the network architecture (routers, protocols, end-hosts, processes, hardware) that are "collaboration friendly";
- Metrics that quantify the security of collaborative approaches over non-collaborative approaches ;
- Who to trust, and to what extent;
- How to prevent attacks that exploit such mechanisms, including insider threats;
- Command-and-control vs. loose-coupling mechanism composition.

Furthermore, in an era of distributed software services (what is fashionably called "Web 2.0"), no single application, node, or network has enough information to detect and counter high-level semantic attacks, or even some of the more conventional web-based malware (*e.g.*, cross-site scripting attacks). Large-scale distributed systems require large-scale distributed defenses. This is particularly true within specific application domains (such as health care and industrial SCADA control), where large-scale collaborative (but independent) defenses will allow better control to critical information and resources.

3 Honey@home

Traditional honeypot architectures are based on monitoring unused IP addresses located at specific institutes and organizations [CBMM04]. This unused IP address space, also called "dark space", is easy to identify and thus be blacklisted by attackers [BFV05]. Furthermore, all honeypot technologies rely on the size of the dark space in order to be effective; the more dark space is used, the faster and more accurate the results obtained. To overcome these two problems, Honey@home [AAM07] empowers ordinary users and organizations, institutes and enterprises, who are not familiar with honeypot technologies, to contribute their dark space to a network of affined honeypots. Many public bodies, universities and even home users do not use all the address space they possess. They also do not have the expertise to setup and maintain a honeypot to monitor that unused

space. Honey@home fills that gap by installing a virtual honeypot to the machine(s) of unfamiliar users. Several other “@home” approaches, like Seti@home and Folding@home, have shown that users can contribute significantly towards a common goal.

Honey@home is designed to be used by people unfamiliar with honeypot technologies. From the user perspective, no configuration is needed. Honey@home is a cross-platform tool that requires minimal resources and can run unsupervised at the background, just like modern messengers. Its basic functionality is to claim an unused IP address through the DHCP server of the local network it is installed on and forward all the traffic going to that address to a centralized farm of honeypots. The centralized farm runs multiple services/applications, and processes all the traffic received from Honey@home clients. Central honeypots will provide answers to the received traffic and send them back to the Honey@home clients. From their side, Honey@home clients will send the responses from honeypots back to the originators of the attack. The attacker is under the impression that she communicates with the address claimed by Honey@home client, but in reality she communicates with a central honeypot that gathers, analyzes, and responds to her attacks and probes. More advanced users can manually declare their dark space and contribute more than one unused IP address. The centralized farm is implemented by a number of Argos [PSB06] honeypots that are able to catch previously unknown attack vectors.

Honey@home enables the creation of an infrastructure where the monitored dark space is distributed over the network and can become arbitrarily large, depending on the number of Honey@home clients. Although the idea of forwarding traffic destined for an unused IP address to a centralized farm of honeypots may sound simple, there are several challenges behind the Honey@home approach. First, participating clients should be undetectable. If an attacker can easily determine whether an address is monitored by Honey@home, clients can be blacklisted and not contribute to the overall infrastructure. (Note, however, that this could be turned into a defensive advantage by acting as a deterrent.) Second, central honeypots must be hidden so that they cannot be remotely exploited or otherwise attacked. Finally, the installation of mock clients that will overload the central honeypots with nonsense traffic must be prevented. Honey@home tries to deal with these challenges by employing various techniques, like anonymization networks [Tor03] to hide honeypots and a registration process to prevent massive automatic installation of fake clients.

4 Application Communities

An Application Community (AC) is a collection of congruent instances of the same application running autonomously on end-hosts distributed across a wide-area network, whose members cooperate in identifying previously unknown flaws/attacks [AC06]. By exchanging information, the AC members may be able to prevent the failure from manifesting in the future. Although individual members may be susceptible to new failures, the AC should eventually converge into a state of immunity against a particular fault, adding a dimension of learning and adaptation to the system. An AC may be considered a “virtual honeypot” composed of many machines/applications that are in actual use (*i.e.*, they are not passive, non-guided entities as traditional honeypots are); AC members contribute a share of their resources (such as CPU cycles) towards the processing done by this virtual honeypot. By using real applications and systems as detectors, Application Communities can identify targeted attacks, attacks that exploit specific state, or attacks that require user action (*e.g.*, for client applications such as web browsers). The size of the AC, in terms of number of participating nodes, impacts coverage (in detecting faults) and fairness (in distributing the monitoring task) [ASA05]. An AC is composed of three main mechanisms, for monitoring, communication, and defense, respectively.

The purpose of the **monitoring mechanism** is the detection of previously unknown (“zero day”) software failures. There exists a plethora of work in this area, namely, using the compiler to insert run-time safety checks, “sandboxing”, anomaly detection, and content-based filtering [2]. While shortcomings may be attributed to each of the approaches, when they are considered within the scope of an AC a different set of considerations need to be examined. Specifically, the significance of the security *vs.* performance tradeoff is de-escalated with respect to the ability to efficiently employ the mechanism in a distributed fashion. The advantage of utilizing an AC is that the use of a fairly invasive mechanism (in terms of performance) may be acceptable, since the associated cost can be distributed to the participating members. By employing a more invasive instrumentation technique, the likelihood of detecting subversion and identifying the source of the vulnerability is increased. The monitoring mechanism in our prototype is an instruction-level emulator that can be selectively invoked for arbitrary segments of code, allowing us to mix emulated and non-emulated execution inside the same execution context [STEM05], although other mechanisms can be used instead (or in addition) [SGK05].

Once a failure is detected by a member’s monitoring component, the relevant information is distributed across the AC. Specifically, the purpose of the **communication component** is the dissemination of information

pertaining to the discovery of new failures and the distribution of the monitoring work load within the AC. The choice of the communication model to be employed by an AC is subject to the characteristics of the collaborating community, such as size and flexibility. The immediate trade-off associated with the communication model is the overhead in messages versus the latency of the information in the AC. In the simplest case, a centralized approach is arguably the most efficient communication mechanism. However, there are a number of scalability and trust issues associated with this approach. If there is a fixed number of collaborating nodes, a secure structured overlay network can be employed, with exemption from the problems associated with voluminous joins and leaves. If nodes enter and leave the AC at will, a decentralized approach may be more appropriate. Efficient dissemination of messages is outside the scope of this paper, but has been the topic of much research in the networking community.

The **immunizing component** of our architecture is responsible for protecting the AC against future instances of a specific failure. Ideally, upon receiving notification of a failure observed by another AC member, individual members independently confirm the validity of the reported weakness and create their own fix in a decentralized manner. At that point, each member in the AC decides autonomously what fix to apply in order to inoculate itself. As independent verification of an attack report may be impossible in some situations, a member's action may depend on predefined trust metrics (*e.g.*, trusted verifications servers). Depending on the level of trust among users, alternative mechanisms may be employed for the adoption of universal fixes and verification of attack reports. In the case of systems where there is minimal trust among members a voting system can be employed at the cost of an increased communication overhead. Finally, given that a fix could be universally adopted by the AC, special care must be placed in minimizing the performance implications of the immunization.

The inoculating approach that can be employed by the AC is contingent on the nature of the detection mechanism and the subsequent information provided on the specific failure. The type of protection can range from statistical blocking, behavioral or structural transformation. For example, IP address and content filtering, code randomization [KKP03], adaptive defenses [SGK05], and emulation [STEM05] may be used for the protection of the AC members.

5 Conclusions

We have argued that the Internet-borne cyber-attacks of the future require collaborative solutions that encompass (and perhaps focus) on end-

hosts, rather than depend on network-based defenses. We have briefly described two such research thrusts, Honey@home [AAM07] and Application Communities [AC06]. Although there are many research challenges (and opportunities) ahead, we believe that large-scale collaborative defenses hold the key to a future secure Internet.

Acknowledgments: This material is partially based on research sponsored by the Air Force Research Laboratory under agreement number FA8750-06-2-0221 and by NSF Grant 06-27473, with additional support from Google and New York State. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. There opinions expressed herein do not reflect those of the NSF or the U.S. Government.

References

- [AAM05] P. Akritidis, K. G. Anagnostakis, and E. P. Markatos. *Efficient content-based worm detection*. In *Proceedings of the 40th IEEE International Conference on Communications (ICC)*, 2005.
- [AAM07] S. Antonatos, K. G. Anagnostakis and E. P. Markatos. *Honey@home: A New Approach to Large-Scale Threat Monitoring*. To appear in the *Proceedings of the 5th ACM Workshop on Recurring Malcode (WORM)*, November 2007.
- [ASA05] K. G. Anagnostakis, S. Sidiroglou, P. Akritidis, K. Xinidis, E. Markatos, and A. D. Keromytis. *Detecting Targeted Attacks Using Shadow Honeypots*. In *Proceedings of the 14th USENIX Security Symposium*, pages 129-144, August 2005.
- [AAMA05] S. Antonatos, P. Akritidis, E. P. Markatos, and K. G. Anagnostakis. *Defending against hit list worms using network address space randomization*. In *Proceedings of the ACM Workshop on Rapid Malcode (WORM)*, pages 30-40, November 2005.
- [AC06] M. E. Locasto, S. Sidiroglou, and A. D. Keromytis. *Software Self-Healing Using Collaborative Application Communities*. In *Proceedings of the 13th ISOC Symposium on Network and Distributed Systems Security (SNDSS)*, pages 95-106,

February 2006.

- [BCJ+05a] M. Bailey, E. Cooke, F. Jahanian, J. Nazario, and D. Watson. *The Internet Motion Sensor: A Distributed Blackhole Monitoring System*. In *Proceedings of the 12th ISOC Symposium on Network and Distributed Systems Security (SNDSS)*, pages 167-179, February 2005.
- [BCJ+05b] M. Bailey, E. Cooke, F. Jahanian, D. Watson, and J. Nazario. *The Blaster worm: Then and now*. In *IEEE Security & Privacy Magazine*, 3(4):26-31, 2005.
- [BFV05] J. Bethencourt, J. Franklin, and M. Vernon. *Mapping Internet Sensors With Probe Response Attacks*. In *Proceedings of the 14th USENIX Security Symposium*, pages 193-208, August 2005.
- [CBMM04] E. Cooke, M. Bailey, Z. M. Mao, and D. McPherson. *Toward Understanding Distributed Blackhole Placement*. In *Proceedings of the ACM Workshop on Rapid Malcode (WORM)*, pages 54-64, October 2004.
- [DQG+04] D. Dagon, X. Qin, G. Gu, W. Lee, J. Grizzard, J. Levine, and H. Owen. *HoneyStat: Local Worm Detection Using Honeypots*. In *Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection (RAID)*, pages 39-58, October 2004.
- [DUMU03] T. Detristan, T. Ulenspiegel, Y. Malcom, and M. Underduk. *Polymorphic shellcode engine using spectrum analysis*. In *Phrack*, 11(61), August 2003.
- [K201] K2. *ADMmutate*. <http://www.ktwo.cal/ADMmutate-0.8.4.tar.gz>.
- [KE03] D. M. Kienzle and M. C. Elder. *Recent worms: A survey and trends*. In *Proceedings of the ACM Workshop on Rapid Malcode (WORM)*, pages 1-10, 2003.
- [KK04] H. Kim and B. Karp. *Autograph: Toward automated, distributed worm signature detection*. In *Proceedings of the 13th USENIX Security Symposium*, pages 271-286, August 2004.
- [KKP03] G. S. Kc, A. D. Keromytis, and V. Prevelakis. *Countering*

Code-Injection Attacks With Instruction-Set Randomization.
In *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS)*, pages 272–280, October 2003.

- [LAAA06] V. T. Lam, S. Antonatos, P. Akritidis, and K. G. Anagnostakis. *Puppetnets: Misusing Web Browsers as a Distributed Attack Infrastructure.* In *Proceedings of the 13th ACM Conference on Computers and Communications Security (CCS)*, November 2006.
- [MD88] P. Mockapetris and K. J. Dunlap. *Development of the Domain Name System.* In *Proceedings of the ACM SIGCOMM Symposium on Communications architectures and protocols*, pages 123-133, 1988.
- [MPS+03] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. *Inside the Slammer worm.* In *IEEE Security & Privacy Magazine*, 1(4):33-39, 2003.
- [MSB02] D. Moore, C. Shannon, and J. Brown. *Code-Red: A case study on the spread and victims of an Internet worm.* In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurement (IMW)*, pages 273-284, 2002.
- [MVSOI] D. Moore, G. Stalker, and S. Savage. *Inferring Internet Denial-of-Service Activity.* In *Proceedings of the 10th USENIX Security Symposium*, pages 9-22, August 2001.
- [PSB06] G. Portokalidis, A. Slowinska, and H. Bos. *Argos: an Emulator for Fingerprinting Zero-Day Attacks.* In *Proceedings of ACM SIGOPS Eurosys*, April 2006.
- [RMT05] M. A. Rajab, F. Monrose, and A. Terzis. *On the Effectiveness of Distributed Worm Monitoring.* In *Proceedings of the 14th USENIX Security Symposium*, pages 225-237, August 2005.
- [RMT06] M. A. Rajab, F. Monrose, and A. Terzis. *Fast and Evasive Attacks: Highlighting the Challenges Ahead.* In *Proceedings of the 9th International Symposium on Recent Advances in Intrusion Detection (RAID)*, pages 206-225, September 2006.
- [Roe99] Martin Roesch. *Snort: Lightweight intrusion detection for networks.* In *Proceedings of USENIX LISA*, pages 229-238, 1999.

- [SEVS04] S. Singh, C. Estan, G. Varghese, and S. Savage. *Automated Worm Fingerprinting*. In *Proceedings of OSDI*, pages 45-60, 2004.
- [SFO1] P. Szor and P. Ferrie. *Hunting for metamorphic*. In *Proceedings of the Virus Bulletin Conference*, pages 123-144, September 2001.
- [SGK05] Stelios Sidiroglou, Giannis Giovanidis, and Angelos D. Keromytis. *A Dynamic Mechanism for Recovering from Buffer Overflow Attacks*. In *Proceedings of the 8th Information Security Conference (ISC)*, pages 1-15, September 2005.
- [SII05] Y. Shinoda, K. Ikai, and M. Itoh. *Vulnerabilities of Passive Internet Threat Monitors*. In *Proceedings of the 14th USENIX Security Symposium*, pages 209-224, August 2005.
- [SM04] C. Shannon and D. Moore. *The spread of the Witty worm*. In *IEEE Security & Privacy Magazine*, 2(4):46-50, 2004.
- [SMPW04] S. Staniford, D. Moore, V. Paxson, and N. Weaver. *The Top Speed of Flash Worms*. In *Proceedings of the ACM Workshop on Rapid Malcode (WORM)*, pages 33-42, November 2004.
- [Spi03] L. Spitzner. *Honeypots: Tracking Hackers*. Addison-Wesley, 2003.
- [SPW02] S. Staniford, V. Paxson, and N. Weaver. *How to Own the Internet in your spare time*. In *Proceedings of the 11th USENIX Security Symposium*, August 2002.
- [STEM05] S. Sidiroglou, M. E. Locasto, S. W. Boyd, and A. D. Keromytis. *Building Ra Reactive Immune System for Software Service*. In *Proceedings of the USENIX Annual Technical Conference*, pages 149-161, April 2005.
- [Szo05] Peter Szor. *The Art of Computer Virus Research and Defense*. Addison-Wesley Professional, February 2005.
- [Tor03] R. Dingledine, N. Matthewson, and P. Syverson. *Tor: The Second-Generation Onion Router*. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.
- [VE06] R. Vaughn and G. Evron. *DNS Amplification Attacks*

(Preliminary Release). <http://www.isotf.org/news/DNS-Amplification-Attacks.pdf>.

- [XCA +06] K. Xinidis, I. Charitakis, S. Antonatos, K. G. Anagnostakis, and E. P. Markatos. *An active splitter architecture for intrusion detection and prevention*. In *IEEE Transactions on Dependable Secure Computing*, 3(1):31-44, 2006.
- [YBP04] V. Yegneswaran, P. Barford, and D. Plonka. *On the Design and Use of Internet Sinks for Network Abuse Monitoring*. In *Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection (RAID)*, pages 146-165, October 2004.