A Novel DDoS Attack Defending Framework with Minimized Bilateral Damages

[†]Yu Chen[*], [‡]Wei-Shinn Ku, [‡]Kazuya Sakai, [†]C. DeCruze

[†]Dept. of Electrical & Computer Engineering, SUNY - Binghamton, Binghamton, NY 13902
[‡]Dept. of Computer Science & Software Engineering, Auburn University, Auburn, AL 36849

Abstract **- Distributed Denial of Service (DDoS) attacks are one of the most damaging threats against Internet based applications. Many of the DDoS defense mechanisms may unintentionally deny a certain portion of legitimate user accesses by mistaking them as attackers or may simply not block enough traffic to adequately protect the victim. Other better performing systems have not yet to reach adoption because of designs that require a substantial investment into the Internet infrastructure before offering much effectiveness. This paper proposes Heimdall, a novel traffic verification based framework to protect legitimate traffic from bilateral damages. Based on a proof-of-work technique and application of distributed hash ID, aside from protecting established connections, our system can validate new initial request for communication and open valid channels between users and the protected server. Through intensive simulation experiments on the ns-2 network simulator, we verified that Heimdall scheme can effectively protect legitimate communications and filter out malicious flows with very high accuracy.**

Keywords: Network security, DDoS Attacks, Traffic verification, Proof-of-work system.

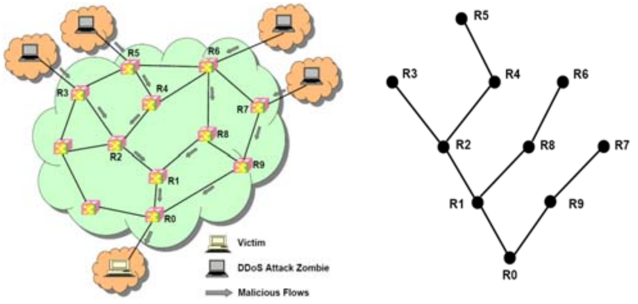
1. Introduction

Distributed Denial of Service (DDoS) attacks have become one of the major threats to Internet based services and electronic transactions [9]. While many research efforts have been suggested, widespread adoption of any DDoS defense has yet to happen. Fighting against DDoS attacks effectively on the Internet has been a pressing task which involves two critical issues: (1) accurately identifying the machines participating in the forwarding of malicious flows and (2) effectively cutting the malicious flows at those machines while minimizes the bilateral damages on legitimate traffic flows [3].

Previously, we have proposed a distributed change-point detection technique to detect DDoS attacks at an early stage [3]. A *Change-Aggregation Tree* (CAT) is constructed and nodes in a CAT are routers that participate in forwarding the malicious flows. Such routers are called *Attack Transit Routers* (ATRs). The links in the CAT indicate the path along which malicious attacking traffic goes through towards the victim machine. Essentially, the CAT presents the spatiotemporal propagation pattern of the malicious flows inside the network.

Figure 1 illustrates the CAT construction principle in distributed DDoS attack detection. Figure 1(a) shows a flooding attack launched from 4 zombies. The ATRs detect abnormal surge of traffic at their I/O ports. The victim is attached to the end router R0. All the attack flows home towards the end router. Figure 1(b) presents a CAT tree rooted at the end router. The CAT presents a traffic-flow tree pattern rooted at the router connected to the edge network, where the victim is attached. Hence, once a CAT is constructed, a DDoS attack is detected and ATRs are identified. The next task is to filter out malicious flows while minimize the impact on the performance of legitimate applications.

Note that CAT based detection scheme can be extended into multiple ISP networks or *Autonomous Systems* (ASs). The ATRs in a CAT can belong to multiple domains and those domains are not necessarily physically connected with each other directly. Due to the limited space, interested readers can find more detailed description of CAT in [3].



(a) (b)

Figure 1. Illustration of the CAT principle.

Unfortunately, accurate segregation of malicious flows from legitimate flows is very difficult. Many of the suggested DDoS defense schemes may unintentionally deny a certain portion, to a greater or lesser percent, of legitimate users' access by mistaking them as attackers or may simply not block enough attacking traffic to adequately protect the victim. Other better performing systems have not yet to reach adoption because of designs that require a substantial investment into the Internet infrastructure.

In this paper, we propose a novel DDoS attack countermeasure framework called *Heimdall*, which offers significant defense without significant infrastructure

[*] Manuscript submitted on Oct. 01, 2009 to *the 7th IEEE Consumer Communication and Networking Conference - Security for CE Communications* (CCNC'10), Las Vegas, NV, USA, Jan. 9 – 12, 2010. Corresponding author: Yu Chen, SUNY – Binghamton, Binghamton, NY 13902. E-mail: ychen@binghamton.edu, Tel.: (607) 777-6133.

changes. Taking advantage of existing proof-of-work technique and application of distributed hash ID, aside from protecting established connections, our system can validate new initial requests for communication and open valid channels between users and the protected server. Through intensive simulation experiments using the ns-2 network simulator, we verified that Heimdall can effectively protect legitimate communications and filter out malicious flows with very high accuracy.

The rest of the paper is organized as follows: Section 2 provides a brief review of related work. Section 3 discusses the rationale and architecture of Heimdall. Section 4 presents the experiment results and the system performance evaluation. Section 5 discusses several design issues and concludes this paper.

2. Related Work

A considerable number of DDoS attack defense and response mechanisms have been suggested [2], [9]. This section provides a brief overview of DDoS defense schemes that are closely related to our work.

From the perspective of system deploy locations, a DDoS countermeasure can be allocated at the victim network, the source network, or/and distributed in the network. Defenses located at the victim side help to alleviate the excessive traffic. Thus, they provide better protection per dollar for a victim. The distributed network-wide defense mechanisms rely on changes in the current Internet. Such changes would need to happen on all parts of the Internet's topology. ISPs would have responsibility for source network defenses. The providers would then need to guarantee no malicious traffic generated from their users can exit the ISP network and enter the surrounding networks. Alternatively, traffic that does exit has some manner of simple identification that allows for simple filtering of malicious traffic [2].

Hop-Count filtering [7] proposed to detect suspicious packets by comparing the hops between the source and the victim. This avoids the assumption that a relationship exists between the TTL field in legitimate TCP/IP packets and the number of hops a packet travels across. Traffic level measurements [1] focus on the way legitimate traffic acts when face communication difficulties. In this case, when the victim detects an aggregate, it compares the traffic patterns before the aggregate with the patterns after the aggregate to differentiate legitimate and illegitimate traffic. A broad range of anti-DDoS techniques comes under the term Packet Filtering [12]. In the case of an attack, or when the pool of the victim's available resources is close to empty, the server will give preference to trusted users over untrusted ones. Trust forms by monitoring traffic behavior patterns to separate typical usage from malicious attacks.

Communication between routers allows for aggregate management before it may congest Internet pathways. In Pushback defenses [5], [6], the detection of aggregates triggers routers to identify malicious flows and implement traffic controls. This router sends a message to connected aggregate carrying routers and each router continues the strategy to push back congestion. StackPi [13] works to build a map of the route packets take as they traverse across networks to identify the source of malicious traffic in place of non reliable source IPs. The IP Identification field of the IP packet contains hashes of the packet's path. As the field fills, newer entries may overwrite older ones, but the amount of concatenation, and the range on routers which perform such markings are adjustable depending on the implementation of StackPi. WebSOS [5] offers a great deal of protection with a great deal of complexity. Direct communication with the victim cannot happen. The only communication happens through perimeter nodes. Each incoming connection must also have passed verification from the protected server. The perimeter of secure nodes have the responsibility of ensuring malicious traffic stays out of the secure network and randomizing the paths taken once the data gets inside.

Since DDoS attacks used spoofed IP addresses to avoid detection, the attackers fail to establish a valid connection and prevent legitimate users from making one. Portcullis [11] uses a proof-of-work system to prioritize capability request packets. Potential users must spend CPU time to solve puzzles and in doing so, DDoS advantage of superior available bandwidth lessens, allowing legitimate users to establish connections. While the intermediate network defenses do encompass the victim network and the source network, D-WARD [10] falls entirely in the domain of the source network: ISPs. In the event of aggregates, D-WARD will dramatically reduce available bandwidth or drop outgoing packets entirely. The exterior routers can monitor the valid IPs that exists behind them, preventing forged IP addresses from leaving the AS.

Among the previously reported works, Portcullis and Pushback defenses most closely resemble Heimdall. To ensure the validity of puzzles, Portcullis uses unique identifiers distributed from DNS servers. While Portcullis shows a great deal of success with “limited” deployment, its success still relies on the cooperation of the Internet's governing bodies to allow for the distribution of puzzle keys. In contrast, Heimdall utilizes the CAT tree obtained during the detection of DDoS attacks [3] to track the path of malicious traffic flows. Work reported below focuses on the principle and framework of the countermeasure, the specific design and implementation of puzzle generation/solving functions are beyond the scope of this paper.

3. Design of Heimdall Architecture

This section presents the function blocks of our Heimdall architecture first, then the detailed design and principles are introduced in section 3.2.

3.1 Heimdall System Architecture

As shown in Figure 2 below, Heimdall architecture consists of three distinct function units: a puzzle/identifier generator, a puzzle solution verifier, and a puzzle resolver. They are allocated at the victim, the intermediary Heimdall routers, and the user machine respectively.

The puzzle/identifier generator is deployed at the user machines that adopt the Heimdall system. When a DDoS attack is detected, a CAT is constructed and the victim is recognized. Then the generator produces a puzzle and a unique puzzle identifier (UPI). The victim machine sends them to Heimdall routers in the CAT tree. And the generator will keep producing new UPIs and sending them to Heimdall routers periodically.

Puzzle verifiers are deployed in the intermediary Heimdall routers. When an initiation of new connection to the victim is received, the router will not forward this request to the victim immediately. Instead, the verifier generates a new ID for the client and sends it along with the puzzle and UPI to the client. Then, the router validates the client's responses. Only if the client solved the puzzle correctly the router will forward the connection initialization request to the victim. Otherwise, any packet sent by the client to the victim will be dropped.

At the client side, there is a puzzle resolver. This enables the client to receive packets from the router, which contains a puzzle, a UPI, and an ID. Upon receiving such a packet, the resolver will figure out a puzzle solution and transmit the result to the appropriate router.

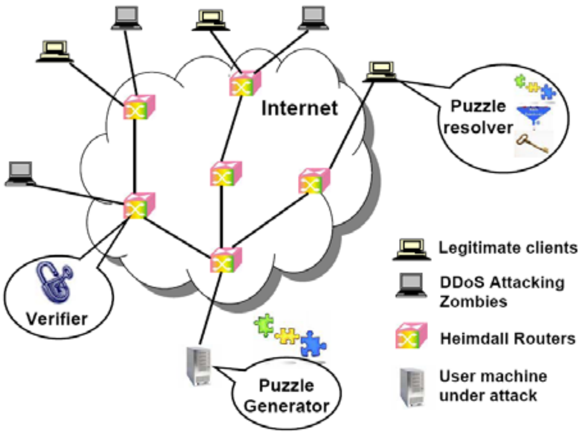


Figure 2. Deployment of Heimdall function units.

3.2 Design Rationale

With the three main components as illustrated in Figure 2, the Heimdall system works by utilizing a proof-of-work technique similar to Portcullis [11]. However, Heimdall system focuses primarily on validating the initial request for communication, the opening of a valid channel between clients and the protected server. Certain existing systems allow established connections to prioritize traffic before it reaches the server. However, these systems lack a way for new users to establish valid connections during a DDoS attack as users whose requests appear in an aggregate may initially appear indistinguishable from the malicious traffic. The design goal of Heimdall is to guarantee that all legitimate users can establish connections with the protected victim even during a DDoS attack.

Using the CAT constructed during the DDoS attack detection [3], as the root of the CAT, the victim server will communicate with the ATRs and ATRs communicate with peers during the verification process as described in [3]. For ARTs that have Heimdall puzzle verifier embedded, the key function they take can be broken down into two parts: puzzle identification and puzzle verification.

Puzzle identification provides an approach by which ATRs can verify that the clients have solved the assigned puzzles, rather than simply computing different puzzles beforehand. Only upon detection of an attack, the victim will generate and send puzzles and UPIs to ATRs. The UPI value will be updated periodically, so that each UPI remains valid for some fixed period of time.

Through puzzle verification the verifiers recognize and validate legitimate users if the replied UPI and puzzle solution match. The puzzles need to meet certain criteria. It should allow users to geometrically increase their effort, take the ID and UPI as inputs so multiple users cannot share work and is cryptographically secure.

For clients to perform proof-of-work, they also need to complete two tasks. Firstly the clients have to figure out the correct solution of the puzzle in fixed time constraint. Then, a hash function is executed with the input including the puzzle solution, the UPI and the ID issued by the router.

An example puzzle is similar to the one used in Portcullis [11], where the user performs a hash function $H(x, r, ID) = p$. Here, x is the solution to the puzzle, r is the UPI and ID is the assigned client ID. The UPI prevents attackers from using the same solution to launch replay attacks as the server would release its own identifier. This also removes the need of source IP address as part of the puzzle parameters. Each machine starting to establish a connection must include a specific ID given by the router. When duplicate IDs are detected, all received packets except the first one will be dropped to preclude collaborative work on the puzzles.

Figure 3 illustrates the workflow of the Heimdall system. Once a DDoS attack has been detected, a CAT tree will be created as shown in Figure 1 and the victim is identified. Then the ATRs will receive puzzles and UPIs periodically from the victim machine. When a client initiates a new connection with the victim, the Heimdall router will generate a new ID for the client and send it along with a puzzle to the user. Once solved the puzzle, the user will calculate a hash function $H(x, r, ID)$ with the inputs including the puzzle solution, the UPI, and the client ID.

When the router receives the hash function output from the client, it will check whether it matches the correct result. Only users which are able to pass the verification are allowed to connect to the victim.

In case when an ISP fully adopts Heimdall, it is ideal to make all of the edge routers support Heimdall to best protect its clients. The edge routers may themselves become targets of a coordinated DDoS attack. If the path to such a router does not include any other hardened routers it may not be able to cope with the attack. However, the attackers have very little ability to control the routes their traffic takes and would not be able to ensure such attack paths. If there are multiple ISPs adopt Heimdall, the spread of Heimdall routers would make such attacks targeted on individual routers even more difficult.

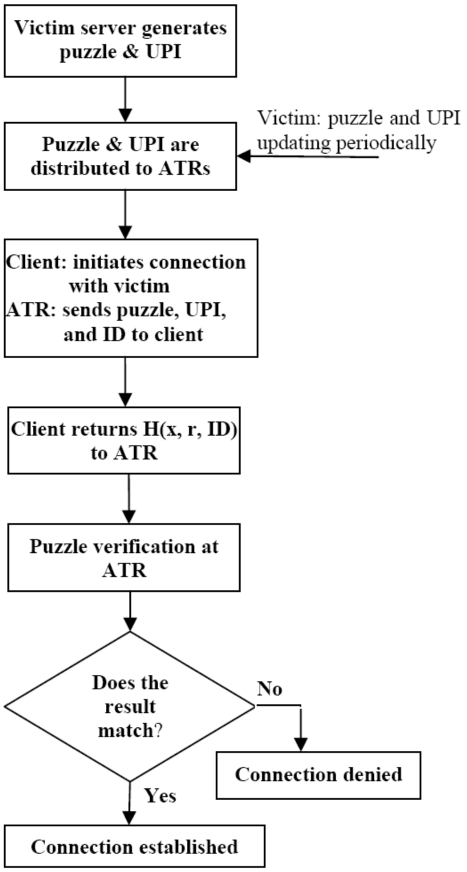

Victim server generates puzzle & UPI
Puzzle & UPI are distributed to ATRs
Victim: puzzle and UPI updating periodically
Client: initiates connection with victim
ATR: sends puzzle, UPI, and ID to client
Client returns H(x, r, ID) to ATR
Puzzle verification at ATR
Does the result match?
No
Connection denied
Yes
Connection established

Figure 3. The workflow of Heimdall.

4. Experiment and Performance Analysis

In this section, we give the definition of the performance evaluation metrics first, and then the simulation experiment configurations are discussed. In subsection 4.3, the numerical simulation results and some discussions regarding the effectiveness of the Heimdall system are presented.

4.1 Performance Evaluation Metrics

To evaluate the performance of our Heimdall system, we focus on the percentage of new legitimate connections established versus the delay users need to tolerate. We studied the effectiveness of our system with different values of parameters including the percent of existing legacy routers, the amount of malicious traffic, and the number of Heimdall routers. Below are the metrics we adopted in our simulation experiments:

- *Successful connection rate*: the percentage that legitimate users can connect to the server;
- *Connection delay (sec.)*: the time required by the three-hand shake;

4.2 Experiment Setup

The network topology used in our ns-2 simulation experiments is similar to Figure 1. The act of puzzle solving is simulated by setting a fixed delay time that the users wait before sending a result. As shown in Figure 1, once the CAT tree has been constructed we know which routers are ATRs. Heimdall routers are randomly placed in the network. The victim server is connected to R0, while the other ASs contain malicious DDoS attack zombies and legitimate users who desire to access certain services provided by the server. As each user tries to reach the protected server (victim) their traffic must pass through a number of intermediary Heimdall routers. The total amount of malicious traffic from zombies rages from 20 KB/Sec to 200 KB/Sec. During DDoS attacks, 1000 new TCP requests are generated.

4.3 Experiment Results and Performance Evaluation

Figure 4 compares the percentages of successful newly initiated connections under variant DDoS attacks. As the amount of malicious traffic becomes higher than 120 KB/Sec, it is more and more difficult for users set up new connections with the server. The successful connection rate decreases quickly. As shown in Fig. 4, without the help of Heimdall routers, when the amount of malicious traffic is 200 KB/Sec, less than 40% of the newly initiated connections can be set up successfully. In contrast, when three routers, R4, R6, and R9, support Heimdall scheme, all the new connections are set up successfully under variant amount of malicious traffic.

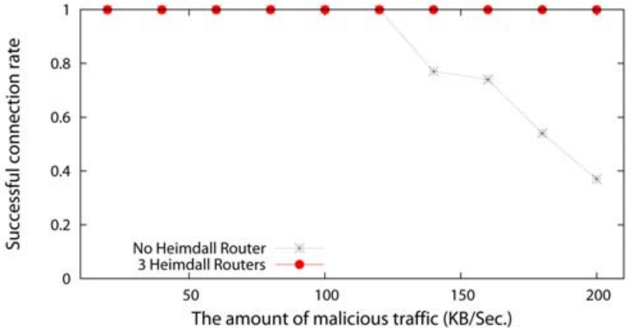


Figure 4. Successful connection rate.

Figure 5 presents the average delays those success legitimate connections have to experience under different attacking scenarios. Again, it is obvious that the users need to wait longer time when the DDoS attack traffic rate increases. Particularly, the average delay increases drastically when the attacking rate is higher than 140 KB/Sec. Our simulation results also show that the Heimdall routers effectively decrease the delay the users suffer.

In addition, compare with the average connection delay when there is no DDoS attack, the puzzle solving operation does introduce some extra delay. Compare to the much longer delay under attacks, it is acceptable for legitimate users to spend a little bit time to solve the puzzles.

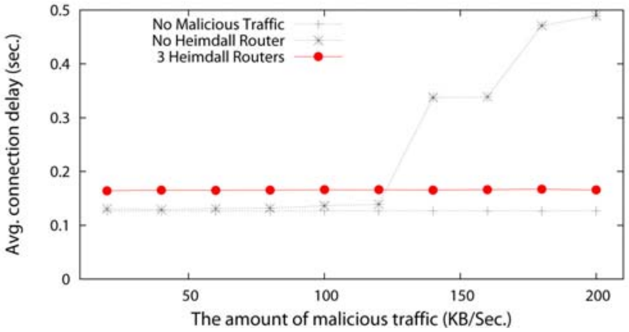


Figure 5. Average connection delay.

Figure 6 illustrates the results of the impact of the number of Heimdall routers in the network. The successful connection rate increases in proportion to the number of Heimdall routers. When more than 3 Heimdall routers exist, all of the new connections are established successfully. Connection delay decreases toward the number of Heimdall routers is 3, and then it slightly increases. This indicates that placing too many Heimdall routers introduces the control overhead to establish a new connection. It is interesting to see that the moderate number of Heimdall routers is preferred to increase the successful connection rate in keeping with lower delay.

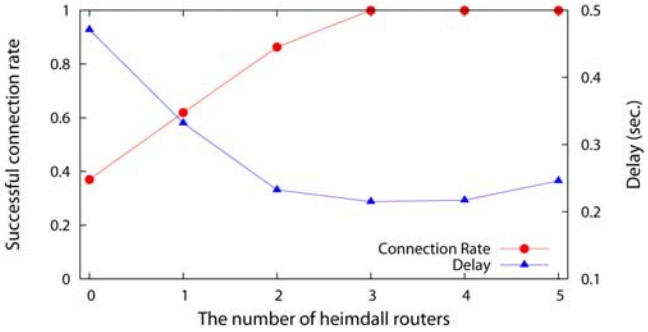


Figure 6. Average connection delay.

5. Conclusions

It is nontrivial to minimize the bilateral damages to legitimate traffic while trying to filter off malicious DDoS attack flows. DDoS defenses may unintentionally deny a certain portion, to a greater or lesser percent, of legitimate users' access by mistaking them as attackers.

This paper reports Heimdall, a novel countermeasure system that effectively validates new initial request for communication and open valid channels between users and the protected server. The experimental results verified the effectiveness of the Heimdall system. However, the reported results are merely based on NS-2 simulation, which is still not close enough to what happened in real world. Currently, we are planning conduct more comprehensive and larger scale emulation experiments on the DETER testbed [15].

Next, we will develop a complete mathematic model to gain deeper insight. On the other hand, this paper focused on the system architecture level study, and assumed any existing puzzle-solving approach can be adopted. In our on-going efforts, a novel proof-of-work system is being designed to take full advantage of the established CAT tree once a DDoS attack has been detected.

References

[1] B. Bencsath, and I. Vajda, "Protection Against DDoS Attacks Based on Traffic Level Measurements," *Western Simulation MultiConference*, January 2004.

[2] A. Challita, M. E. Hassan, S. Maalouf, and A. Zouheiry, "A Survey of DDoS Defense Mechanisms," *FEA Student Conference*, 2004.

[3] Y. Chen, K. Hwang, and W.-S. Ku, "Collaborative Detection of DDoS Attacks over Multiple Network Networks," *IEEE Transaction on Parallel and Distributed Systems*, Vol. 18, No. 12, December 2007.

[4] Y. Chen, Y. Kwok, and K. Hwang, "MAFIC: Adaptive Packet Dropping for Cutting Malicious Flows to Push Back DDoS Attacks," *IEEE International Conference on Distributed Computing Systems Workshops*, June 2005, pp. 123 – 129.

[5] D. L. Cook, W. G. Morein, A. D. Keromytis, V. Misra, and D. Rubenstein, "WebSOS: Protecting Web Servers from DDoS attacks," *Proceedings of the 11th IEEE International Conference on Networks (ICON 2003)*, September 2003, pp. 455 – 460.

[6] J. Ioannidis and S. Bellovin, "Implementing Pushback: Router-Based Defense Against DDoS Attacks," *Network and Distributed System Security Symposium*, 2002 pp. 100 – 108.

[7] G. Jin, H. Wang, and K. G. Shin, "Hop-count Filtering: An Effective Defense Against Spoofed DDoS Traffic," *Proceedings of the 10th ACM Conference on Computer and Communication Security*, 2003.

[8] R. Mahajan, S. Bellovin, and S. Floyd, "Controlling High Bandwidth Aggregates in the Network," *ACM SIGCOMM Computer Communications Review*, July 2002, pp. 62 – 73.

[9] J. Mirkovic, J. Martin, and P. Reiher, "A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms," *ACM SIGCOMM Computer Communications Review*, April 2004, pp. 39 – 54.

[10] J. Mirkovic, G. Prier, and P. Reiher, "Attacking DDoS at the Source," *IEEE International Conference on Network Protocols*, 2002.

[11] B. Parno, E. Shi, A. Perrig, B. Maggs, and Y.-C. Hu, "Portcullis: Protecting Connection Setup from Denial-of-Capability Attacks," *ACM SIGCOM'07*, Aug. 27 – 31, 2007, Japan.

[12] S. Tanachaiwiwat and K. Hwang, "Differential Packet Filtering Against DDoS Flood Attacks," *ACM Conference on Computer and Communications Security*, October 2003.

[13] A. Yaar, A. Perrig, and D. Song, "StackPi: New Packet Marking and Filtering Mechanisms for DDoS and IP Spoofing Defense," *IEE Journal on Selected Areas in Communications*, October 2006, pp. 1853 – 1863.

[14] S. Zhang and P. Dasgupta, "Denying Denial-of-Service Attacks: A Router Based Solution," *International Conference on Internet Computing*, June 2003.

[15] DETER testbed, http://www.deterlab.net/, as of Sept. 2009.