A Pay-per-Use DoS Protection Mechanism For The Web*

Angelos Stavrou, John Ioannidis, Angelos D. Keromytis, Vishal Misra, and Dan Rubenstein

Columbia University
{*angel,ji,angelos,misra,danr*}@*cs.columbia.edu*

Abstract. Internet service providers have resisted deploying Denial-of-Service (DoS) protection mechanisms despite numerous research results in the area. This is so primarily because ISPs cannot directly charge users for the use of such mechanisms, discouraging investment in the necessary infrastructure and operational support.
We describe a pay-per-use system that provides DoS protection for web servers and clients. Our approach is based on WebSOS, an overlay-based architecture that uses reverse Turing tests to discriminate between humans and automated processes that are part of an attack. We extend WebSOS with a credential-based micropayment scheme that combines access control and payment authorization in one operation. Contrary to WebSOS, we use Graphic Turing Tests (GTTs) to prevent malicious code, such as a worm, from using a user's micropayment wallet. Our architecture allows ISPs to accurately charge web clients and servers. Clients can dynamically decide whether to use WebSOS, based on the prevailing network conditions.

1 Introduction

One of the main threats against the reliability of the Web services are (DoS) attacks: attacks that produce an excessive surge of bogus service requests against the target forcing it to processing and (or) to link capacity starvation. These attacks have dire consequences for the target's service viability, since availability and quality of service are of critical importance for the majority of the modern on-line services.

Despite considerable research on devising methods for protection against such attacks [15, 29, 28, 26, 22, 32], so far none of these mechanisms has been widely adopted. Moreover, it has been argued recently [11] that the network DoS problem is inherently impossible to solve without infrastructure support.

However, ISPs seem to be reluctant to deploy such mechanisms. Investment in the necessary infrastructure and operational support are discouraged because such mechanisms represent a poor value proposition: fundamentally, ISPs cannot charge users for

* This work is supported in part by DARPA contract No. F30602-02-2-0125 (FTN program) and by the National Science Foundation under grant No. ANI-0117738 and CAREER Award No. ANI-0133829, with additional support from Cisco and the Intel IT Research Council. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

the use of such mechanisms. One possible solution would be a system with the ability to both protect against DoS attacks and provide a service payment scheme that would allow ISPs to recoup their costs and support the continued operation and maintenance of this infrastructure. Such incentives would motivate router manufacturers to provide appropriate support in their products.

In this paper, we describe a pay-per-use system that provides DoS protection for web servers and clients. Our approach is based on WebSOS, an overlay-based architecture that uses reverse Turing tests to discriminate between humans and automated processes that are part of an attack. We extend WebSOS with a credential-based micropayment scheme that combines access control and payment authorization. Our architecture allows ISPs to accurately charge web clients and servers. Clients can dynamically decide whether to use WebSOS, based on the prevailing network conditions.

WebSOS [23], an enriched implementation of the *Secure Overlay Services (SOS)*, is a DoS-protection architecture [22] for web services. WebSOS enhances the resilience of Web services against congestion-based DDoS attacks, acting as a distributed firewall and filtering attack traffic before it reaches the target. The network immediately surrounding attack targets is protected by high- performance routers that aggressively filter and block all incoming connections from hosts that are not approved. Only a small number of secretly selected secure access points within WebSOS are allowed to contact the target directly. The rest of the nodes use the overlay network as a routing mechanism to forward the requests to these secret nodes (the identity of which varies in time). WebSOS uses Graphic Turing Tests [33] as a means to differentiate anonymous users from automated zombies. Upon connection to the access point, the user was prompted with a GTT test. By preventing large-scale automated attacks, these tests allowed enough time for the overlay system to heal in case of an attack. Contrary to WebSOS, we use Graphic Turing Tests (GTTs) after to prevent malicious code, such as a worm, from using a user's micropayment wallet. This change in order can be done because our service is not anonymous: we have a means of authenticating the user credentials.

We extend WebSOS to include a lightweight offline electronic payment scheme. Although practically any micropayment system can be used in our model, we chose a payment system that can inter-operate with WebSOS' distributed architecture and provide the necessary user credentials. OTPchecks [16] encompasses all these properties: it is a simple distributed scheme, intended for general Internet-based micropayments that produces bank-issued users' credentials which can in turn used to acquire small-valued payment tokens. It has very low transaction overhead and can be tuned to use different risk strategies for different environments making it a suitable payment solution for a wide range of on-line services.

The remainder of this paper is organized as follows: Section 2 gives an overview of Secure Overlay Services (SOS) and discusses the specifics of the WebSOS architecture giving an overview of the Graphics Turing Tests. At the end of this section we provide details on OTPchecks, our micropayment scheme, and its risk strategies. Section 3 presents a detailed description of the extended WebSOS system. The related work is presented in section 4. Section 5 concludes the paper.

2 Background

Since our approach is based on the Secure Overlay Services (SOS) [22] architecture, we will start by giving a brief overview of its important aspects.

2.1 Overview of SOS

Fundamentally, the goal of the SOS infrastructure is to distinguish between authorized and unauthorized traffic. The former is allowed to reach the destination, while the latter is dropped or is rate-limited. Thus, at a very basic level, SOS requires the functionality of a firewall "deep" enough in the network that the access link to the target is not congested. This imaginary firewall performs access control by using protocols such as IPsec [21]. This generally pre-supposes the presence of authentication credentials (*e.g.,* X.509 [6] certificates) that a user can use to gain access to the overlay.

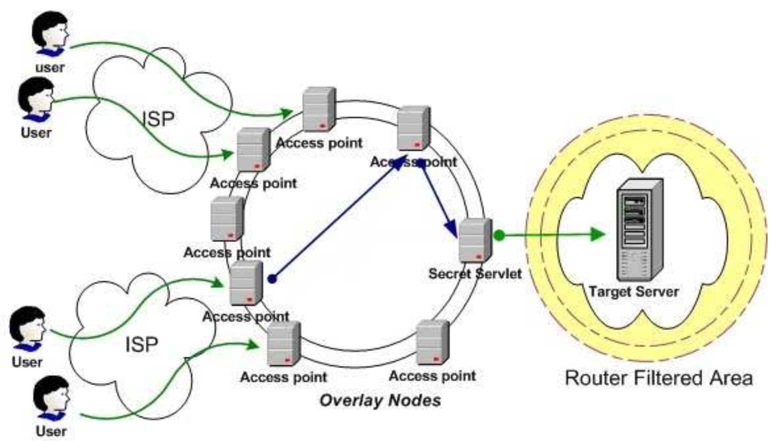

Fig. 1. Basic SOS architecture. Access Points represent an entry point to the SOS overlay. SOS nodes can serve any of the roles of secure access point, beacon or Secret Servlet.

Since traditional firewalls themselves are susceptible to DoS attacks, what is really needed is a distributed firewall [2, 17]. To avoid the effects of a DoS attack against the firewall connectivity, instances of the firewall are distributed across the network. Expensive processing, such as cryptographic protocol handling, is farmed out to a large number of nodes. However, firewalls depend on topological restrictions in the network to enforce access-control policies. In what we have described so far, an attacker can launch a DoS attack with spoofed traffic purporting to originate from one of these firewalls, whose identity cannot be assumed to remain forever secret. The insight of SOS is that, given a sufficiently large group of such firewalls, one can select a very small number of these as the designated authorized forwarding stations: only traffic forwarded from these will be allowed through the filtering router. In SOS, these nodes are called *secret servlets.* All other firewalls must forward traffic for the protected site to these servlets. Figure 1 gives a high-level overview of a SOS infrastructure that protects a target node or site so that it only receives legitimate transmissions. Note that the secret servlets can change over time, and that multiple sites can use the same SOS infrastructure.

To route traffic inside the overlay, SOS uses Chord [30], which can be viewed as a routing service that can be implemented atop the existing IP network fabric, *i.e.,* as a network overlay. Consistent hashing [19] is used to map an arbitrary identifier to a unique destination node that is an active member of the overlay.

SOS uses the IP address of the target (*i.e.,* web server) as the identifier to which the hash function is applied. Thus, Chord can direct traffic from any node in the overlay to the node that the identifier is mapped to, by applying the hash function to the target's IP address. This node, where Chord delivers the packet, is not the target, nor is it necessarily the secret servlet. It is simply a unique node that will be eventually be reached, after up to $m = \log N$ overlay hops, regardless of the entry point. This node is called the *beacon*, since it is to this node that packets destined for the target are first guided. Chord therefore provides a robust and reliable, while relatively unpredictable for an adversary, means of routing packets from an overlay access point to one of several beacons.

Finally, the secret servlet uses Chord to periodically inform the beacon of the secret servlet's identity. Should the servlet for a target change, the beacon will find out as soon as the new servlet sends an advertisement. If the old beacon for a target drops out of the overlay, Chord will route the advertisements to a node closest to the hash of the target's identifier. Such a node will know that it is the new beacon because Chord will not be able to further forward the advertisement. By providing only the beacon with the identity of the secret servlet, traffic can be delivered from any firewall to the target by traveling across the overlay to the beacon, then from the beacon to the secret servlet, and finally from the secret servlet, through the filtering router, to the target. This allows the overlay to scale for arbitrarily large numbers of overlay nodes and target sites. *Unfortunately, this also increases the communication latency, since traffic to the target must be redirected several times across the Internet.* If the overlay only serves a small number of target sites, regular routing protocols may be sufficient.

2.2 Graphic Turing Tests

Graphic Turing Tests(GTTs) are tests designed to provide a way of differentiating a human from a machine by presenting the user with a set of images and asking a questions about the content of the images. CAPTCHA (Completely Automated Public Turing test to Tell Computers and Humans Apart) is a program that generates and grade GTTs [33].

The particular CAPTCHA realization we use is PIX. It consists of a large database of labeled images. All of these images are pictures of concrete objects (a horse, a table, a house, a flower, etc). The program picks an object at random, finds 6 random images of that object from its database, distorts them at random, presents them to the user and then asks the question "what are these pictures of?" as shown in Figure 2. PIX relies on the fact that humans can relate the objects within the distorted image and current automated tools cannot. The human authenticates himself/herself by entering as the description of the object in ASCII text. Graphic Turing Tests are an independent component of our architecture and thus we can update it without changing any other component.

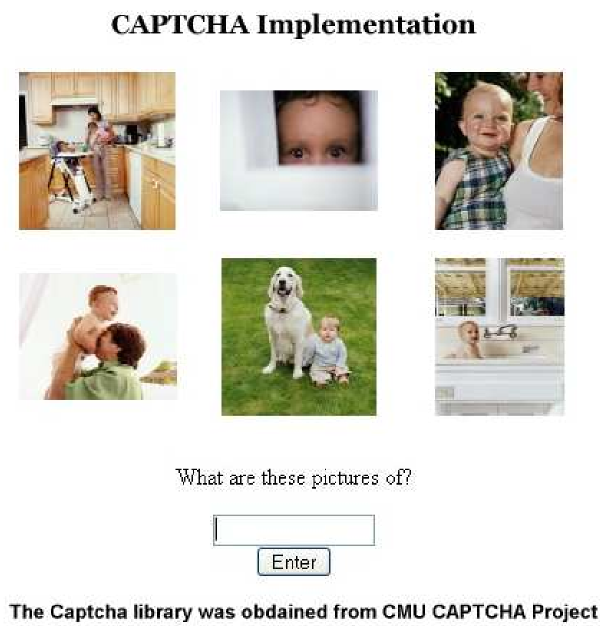

Fig. 2. Web Challenge using CAPTCHA PIX. The challenge in this case is "baby or babies".

Although recent advances in visual pattern recognition [24] can defeat some of the CAPTCHAs, there is no solution to date that can recognize complicated images or relation between images like PIX or Animal-PIX. Although for demonstration purposes in our prototype we use PIX, we can easily substitute it with any other instance of graphic turing test in case a solution to the problem presented by this specific CAPTCHA is discovered.

2.3 WebSOS

WebSOS is the first instantiation of the SOS architecture. The access points participating in the overlay are implemented using Web proxies with SSL to provide two layers of encryption. A source that wants to communicate with the target contacts a random overlay node, the Secure Access Point. After authenticating and authorizing the request via the CAPTCHA test, the overlay node securely proxies all traffic from the source to the target via one of the beacons. The Secure overlay access point(*SOAP*) (and all subsequent hops on the overlay) can proxy the HTTP request to an appropriate beacon in a distributed fashion using Chord, by applying the appropriate hash function(s) to the target's IP address to identify the next hop on the overlay. To minimize delays in future requests, the client is issued a short-duration X.509 certificate, bound to the SOAP and the client's IP address, that can be used to directly contact the proxy-server component of the SOAP without requiring another CAPTCHA test.

In WebSOS, routing decisions are made on a per-connection basis. Any subsequent requests over the same connection (when using HTTP 1.1) and any responses from the web server can take the reverse path through the overlay. While this makes the implementation simpler, it also introduces increased latency, as the bulk of the traffic will also traverse the overlay. To deal with this issue, an adaptation of the initial implementation was created: rather than transporting the request and response through the full overlay network, only routing information travels through the overlay. As before, the requester makes a proxy request to the SOAP. At that point, the SOAP sends a UDP message into the overlay, specifying the target. The message is routed to the beacon, which responds directly to the SOAP with information on the secret servlet for that target. The SOAP then connects to the servlet, which proxies the request as before, in effect creating a *shortcut* through the overlay.

The SOAP caches the servlet information for use in future requests. That information is timed out after a period of time to allow for changes to propagate correctly. The same basic UDP protocol is used by servlets to announce their presence to (and periodically update) the beacons for the various targets.

2.4 OTPchecks Micropayment System

The general architecture of this microbilling system is shown in figure 3. In 3, the Check Guarantor plays the role of *Provisioning*, the Network User plays the role of *Payer*, and the Network Storage Provider (or another NU acting as an NSP) plays the role of the *Merchant*. *Clearing* is done either by a financial institution (if real money is used) or by a selected user of the system (when loyalty points or "play money" are used).

In this system, The *Provisioning* agent issues KeyNote[4] credentials to *Payers* and *Merchants*. These credentials describe the conditions under which a Payer is allowed to perform a transaction, and the fact that a Merchant is authorized to participate in a particular transaction. When a Payer wants to buy something from a Merchant, the Merchant first encodes the details of the proposed transaction into an *offer* which is transmitted to the Payer.

If the Payer wishes to proceed, she must issue to the Merchant a microcheck for this offer. The microchecks are also encoded as KeyNote credentials that authorize payment for a specific transaction. The Payer creates a KeyNote credential signed with her public key and sends it, along with her Payer credential, to the Merchant. This credential is effectively a check signed by the Payer (the Authorizer) and payable to the Merchant (the Licensee). The conditions under which this check is valid match the offer sent to the Payer by the Merchant. Part of the offer is a nonce, which maps payments to specific transactions, and prevents double-depositing of microchecks by the Merchant.

To determine whether he can expect to be paid (and therefore whether to accept the payment), the Merchant passes the action description (the attributes and values in the offer) and the Payer's key along with the Merchant's policy (that identifies the Provisioning key), the Payer credential (signed by Provisioning) and the microchecks credential (signed by the Payer) to his local KeyNote compliance checker. If the compliance

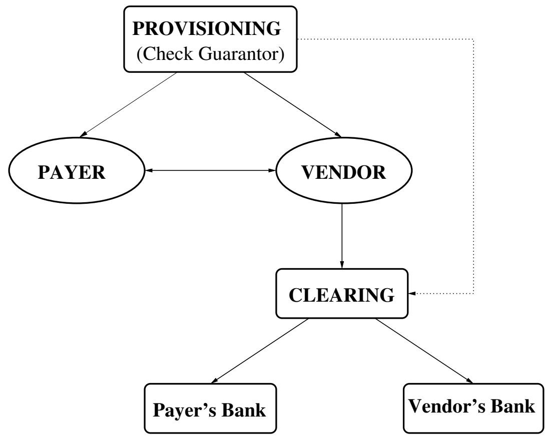

Fig. 3. Microbilling architecture diagram. We have the generic terms for each component, and in parentheses the corresponding players in 3. The arrows represent communication between the two parties: Provisioning issues credentials to Payers and Merchants; these communicate to complete transactions; Merchants send transaction information to Clearing which verifies the transaction and posts the necessary credits/charges or arranges money transfers. Provisioning and Clearing exchange information on the status of Payer and Merchant accounts.

checker authorizes the transaction, the Merchant is guaranteed that Provisioning will allow payment. The correct linkage among the Merchant's policy, the Provisioning key, the Payer key, and the transaction details follow from KeyNote's semantics[4].

If the transaction is approved, the Merchant should give the item to the Payer and store a copy of the microcheck along with the payer credential and associated offer details for later settlement and payment. If the transaction is not approved because the limits in the payer credentials have been exceeded then, depending on their network connectivity, either the Payer or the Merchant can request a transaction-specific credential that can be used to authorize the transaction. Observe that this approach, if implemented transparently and automatically, provides a continuum between online and offline transactions tuned to the risk and operational conditions.

Periodically, the Merchant will 'deposit' the microchecks (and associated transaction details) it has collected to the *Clearing and Settlement Center (CSC).* The CSC may or may not be run by the same company as the Provisioning, but it must have the proper authorization to transmit billing and payment records to the Provisioning for the customers. The CSC receives payment records from the various Merchants; these records consist of the Offer, and the KeyNote microcheck and credential from the payer sent in response to the offer. In order to verify that a microcheck is good, the CSC goes through the same procedure as the Merchant did when accepting the microcheck. If the KeyNote compliance checker approves, the check is accepted. Using her public key as an index, the payer's account is debited for the amount of the transaction. Similarly, the Merchant's account is credited for the same amount.

The central advantage of this architecture is the ability to encode risk management rules for micropayments in user credentials. Other electronic systems have focused on preventing fraud and failure, rather than on managing it. In many cases with such systems, the prevention mechanisms can be too expensive for micropayments, making the risk management approach particularly attractive.

3 Overview of the Pay-per-Use Anti-DoS System

To illustrate the overall system, we now give a thorough description of the necessary software and hardware an ISP needs in order to deploy our a pay-per-use DoS protection mechanism.

3.1 ISP Provisioning

The ISP first creates an overlay network of WebSOS access points ('servlets'). In addition, the routers at the perimeter of the site are instructed to allow traffic only from these servlets to reach the interior of the site's network. These routers are powerful enough to do filtering using only a small number of rules on incoming traffic without adversely impacting their performance.

For a payment scheme, we chose the OTPchecks system because of its inherent flexibility to accommodate different services and its ability to interoperate with a distributed system like WebSOS. Refer to the roles presented in the OTPchecks functional description, in Figure 3; the Payer is the client connecting to the access points, the Vendor is the ISP providing the DoS protection service, and the web service provider (Target) is the clearing entity. The web service provider controls the usage of the service provided via the ISP's network by having the access points delegate payment credentials to each of the clients. In this manner, the service payment can be charged either to the client or to the web service provider. The ISP, using the same transaction information, charges the site providing the web service. The web service itself may charge the user at the same or even a higher rate for the DoS protection and possibly for other Internet commodities (bandwidth, time *etc.*) using the data presented by the access points. The overall system is presented in Figure 4.

3.2 System Operation

We now describe the steps involved in a client using the micropayment scheme in the context of WebSOS. For more details on WebSOS system operation, the reader is referred to [23].

Initialization - System setup When a WebSOS node is informed that it will act as a secret servlet for a site (and after verifying the authenticity of the request, by verifying the certificate received during the SSL exchange), it computes the key k for a number of well-known consistent hash functions, based on the target site's network address. Each of these keys will identify a number of overlay nodes that will act as beacons for that web server.

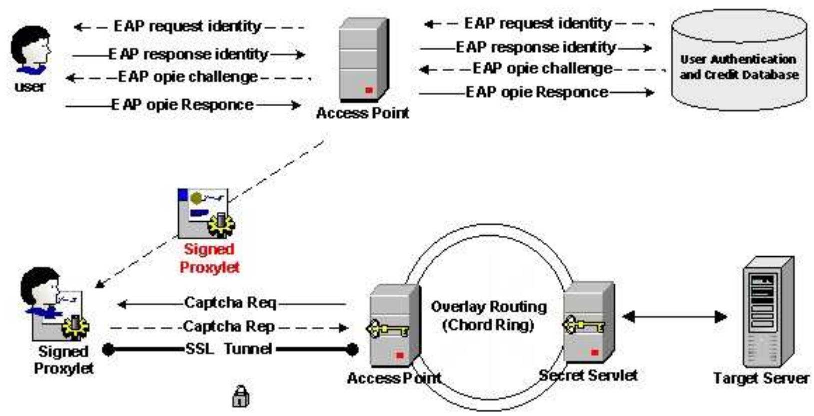

Fig. 4. Pay-per-use DoS protection system operation overview. The user is connected to an access point that in turn authenticates the user credentials and issues an X.509 certificate and a signed proxylet that allows the user to connect securely to the web service for a limited amount time.

Having identified the beacons, the servlets or the target will contact them, notifying them of the servlets' association with a particular target. Beacons will store this information and use it to answer the routing queries of the access points who want to connect to the target. By providing only the beacon with the identity of the secret servlet, traffic can be delivered from any firewall to the target by traveling across the overlay to the beacon, then from the beacon to the secret servlet, and finally from the secret servlet, through the filtering router, to the target.

Since the standard EAP protocol is used, it is possible to use any or all the EAP sub-protocols. However, since neither EAP or EAPoL provide any cryptographic protection themselves, the security of the system depends on the security of the underlying network and on the properties of the EAP sub-protocol. Thus, the risks and the protections must be matched to provide the desired level of security.

Buying OTP coins Whenever a new client host wants to access a service that the ISP protects from DoS attacks, the access point attempts to run the EAPoL protocol with the client. The status of the client is kept unauthenticated as long as the client fails to authenticate through EAPoL. In our case, we provide unauthenticated clients limited access so that they can buy OTP coins, used for the actual EAPoL level authentication (see below).

Using OTP coins Once the Client has acquired a set of OTP coins, it runs the standard EAPoL protocol towards the local access point. The protocol run is illustrated in Figure 4.

Upon connection, the access point requests a user identifier from the client. The client answers with a string that identifies the microcheck used for buying the OTP coins, and the web service the coins where bought for. This allows the access point to contact the correct back-end authenticator, the web service provider (Target). The microcheck fingerprint identifies the relevant unused OTP coin pile.

Once the back-end authenticator receives the identity response, it checks the OTP coin pile and sends an OPIE request, requesting for the next unused OPIE password, *i.e.,* an OTP coin. The Client responds with the next unused coin, H_{i+1}. The back-end authenticator checks the coin, records it as used, and replies with an EAP SUCCESS message. As the access point receives the EAP SUCCESS message from the back-end authenticator, it changes the status of the client into authenticated, and passes the message to the client. Shortly before the OTP coin is used up, the back-end authenticator sends a new OPIE request and a GTT to the client.

For the client to continue, it has to reply with the next OTP coin, and the user must answer correctly the CAPTCHA challenge. This gives us the ability to have a strong protection against malicious code, such as a worm or a zombie process, using a user's micropayment wallet. The lifetime of a coin can be easily configured by the service provider. We expect to prompt a user with a CAPTCHA challenge every 30 to 45 minutes, depending on the service.

On the other hand, if the client does not want to continue access for any reason, he simply does not respond to the request. Thus, if the client goes off-line, the access point automatically changes the status of the client's address into unauthenticated once the coin has been used up.

The access point then issues a short-lived X.509 [6] certificate. This certificate is signed by the ISP operating the overlay, and authorizes the holder to access the web service that was paid for by the coin. The overlay securely proxies all traffic from the source to the target via one of the beacons. The access point (and all subsequent hops on the overlay) can proxy the HTTP request to an appropriate beacon in a distributed fashion using Chord, by applying the appropriate hash function(s) to the target's IP address to identify the next hop on the overlay.

This scheme is robust against DoS attacks because if an access point is attacked, the confirmed source point can simply choose an alternate access point to enter the overlay. Any overlay node can provide all different required functionalities (access point, Chord routing, beacon, secret servlet). If a node within the overlay is attacked, the node simply exits the overlay and the Chord service self-heals, providing new paths over the re-formed overlay to (potentially new sets of) beacons. Furthermore, no node is more important or sensitive than others — even beacons can be attacked and are allowed to fail. Finally, if a secret servlet's identity is discovered and the servlet is targeted as an attack point, or attacks arrive at the target with the source IP address of some secret servlet, the target can choose an alternate set of secret servlets.

3.3 Experimental Evaluation - Latency Results

One of the main concerns of people using DoS systems is the impact of the latency overhead to the end users. Here we include some of the experimental results of WebSOS [23] that show that the end to end latency increases by a factor of two, as shown in Figure 5.

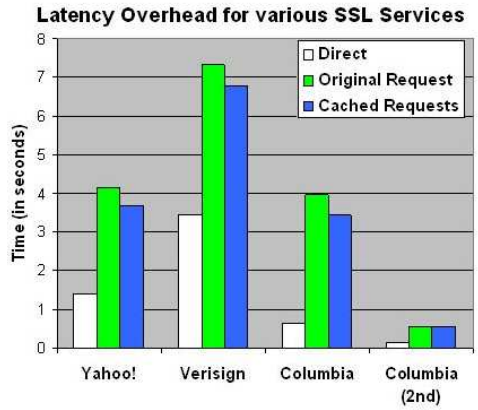

Fig. 5. WebSOS Latency overhead for different SSL-enabled services when using the short-cut routing mechanism

To complete the overhead analysis we measured the number of public key verifications an access point can perform, which indicates how many microchecks it can validate in unit time. We used a 3 GHz Pentium4 processor machine running Linux with the OpenSSL V 0.9.7c library for the measurements. The contribution of the micropayment system to the overall system latency overhead is minimal, even when we issue 1024-bit RSA certificates for the client credentials, as shown in Table 1. These measurements show that the impact of the user verification process on the access points is minimal.

Table 1. Signing and verification times for 1024-bit RSA keys.

Sign	Verify	Sig/sec	Ver/sec
0.0037 sec	0.0002 sec	270.0	5055.9

4 Related Work

Considerable research has been devoted to the problem of network denial of service, with most of the effort focusing on tracing the sources of malicious attacks, filtering out attack traffic at the edges, and filtering inside the network itself.

Methods for tracking down the sources of malicious attacks (*e.g.,* [9, 29, 12] generally require that routers mark packets or that they "remember" whether particular packets (or flows) have been seen in the recent past. Their primary use is in identifying the real sources of attacks involving spoofed traffic (*i.e.,* traffic purporting to originate from an IP address different from that of the real source). As a value proposition, these mechanisms represent the worst approach for ISPs, since there is no way of quantifying their usefulness.

A variant of the packet marking approaches creates probabilistically unique path-marks on packets without requiring router coordination; end-hosts or firewalls can then easily filter out packets belonging to a path that exhibits anomalous behavior [34]. Although this approach avoids many of the limitations of the pure marking schemes, it requires that core routers "touch" packets (rather than simply switch them). Again, however, it is unclear how ISPs can charge for such a service.

Methods that filter at the edges are on the one hand attractive, since they require no action on the part of the ISP, but also (currently) the least successful in defending against DoS attacks, since they require wide deployment (particularly for mechanisms filtering at the sources of attacks). For example, systems that examine network traffic for known attack patterns or statistical anomalies in traffic patterns (*e.g.,* [28]) can be defeated by changing the attack pattern and masking the anomalies that are sought by the filter. The D-WARD system [28] monitors outgoing traffic from a given source network and attempts to identify attack traffic by comparing against models of reasonable congestion control behavior. The amount of throttling on suspicious traffic is proportional to its deviation from the expected behavior, as specified by the model. An extension of D-WARD, COSSACK [25], allows participating agents to exchange information about observed traffic.

An approach that uses BGP to propagate source addresses that can be used for filtering out source-spoofed packets inside the Internet core [26] places undue burden on the core and is useful only in weeding out spoofed packets; unfortunately, the majority of DDoS attacks do not use spoofed packets. [20] proposes using Class-Based Queuing on a web load-balancer to identify misbehaving IP addresses and place them in lower priority queues. However, many of the DDoS attacks simply cause congestion to the web server's access link. To combat that, the load-balancer would have to be placed closer to the network core. Such detailed filtering and especially state-management on a per-source-IP address basis can have performance implications at such high speeds. In [14], the authors use a combination of techniques that examine packet contents, transient ramp-up behavior and spectral analysis to determine whether an attack is single- or multi-sourced, which would help focus the efforts of a hypothetical anti-DoS mechanism. Another interesting approach is that of [18], which proposes an IP hop-count-based filter to weed out spoofed packets. The rationale is that most such packets will not have a hop-count (TTL) field consistent with the IP addresses being spoofed. In practice, most DoS attacks are launched from subverted hosts.

Mechanisms involving filtering inside the network itself (*i.e.,* inside an ISP's infrastructure), such as Pushback [15] require ISP investment (in infrastructure, man power, and operational support). In Pushback, routers push filter towards the sources of an attack, based on the ingress traffic they observe on their various interfaces. Unfortunately, it is unclear how an ISP can charge for such a service; one possibility is as a subscription service, or measuring the number of times a client site invokes the service.

Another approach to mitigating DoS attacks against information carriers is to massively replicate the content being secured around the entire network. To prevent access to the replicated information, an attacker must attack all replication points throughout the entire network — a task that is considerably more difficult than attacking a small number of, often co-located, servers. Replication is a promising means to preserve information that is relatively static, such as news articles. However, there are several reasons why replication is not always an ideal solution. For instance, the information may require frequent updates complicating large-scale coherency (especially during DoS attacks), or may be dynamic by its very nature (*e.g.,* a live web-cast). Another concern is the security of the stored information: engineering a highly-replicated solution without leaks of information is a challenging endeavor.

An extension of the ideas of SOS [22, 23] appears in [1]. There, the two main facets of the SOS architecture: filtering and overlay routing, are explored separately, and several alternative mechanisms are considered. It is observed that in some cases, the various security properties offered by SOS can still be maintained using mechanisms that are simpler and more predictable. However, some second-order properties, such as the ability to rapidly reconfigure the architecture in anticipation of or in reaction to a breach of the filtering identity are compromised. In most other respects, the two approaches are very similar.

The NetBouncer project [32] considers the use of client-legitimacy tests for filtering attack traffic. Such tests include packet-validity tests (*e.g.,* source address validation), flow-behavior analysis, and application-specific tests, including Graphic Turing Tests. However, since their solution is end-point based, it is susceptible to large link-congestion attacks.

[3] examines several different DDoS mitigation technologies and their interactions. Among their conclusions, they mention that requiring the clients to do some work, *e.g.,* [10], can be an effective countermeasure, provided the attacker does not have too many resources compared to the defender. Gligor [11] disagrees with this conclusion, noting that computational client puzzles cannot provide hard bounds (guarantees) on client wait time.

Although we use a particular micropayment system [5], other schemes can also be used, including digital cash systems (*e.g.,* [7]), scrip-based micropayments (*e.g.,* [27]), and offline micropayment protocols (*e.g.,* [31]). MiniPay [13] is particularly attractive, since it was developed primarily for use with a web browser, with considerable effort gone into the user interface aspect. Risk management is implemented as a decision to perform an online check with the billing server based on the total spending by the customer that day, and some parameter set by the merchant. We believe that general transactional payment schemes (*e.g.,* [8]) may prove too heavy-weight for our purposes.

5 Conclusion

We present the first pay-friendly DoS protection system that furnishes ISPs with a better value proposition for deploying anti-DoS systems: a way to turn DoS protection into a commodity. Our pay-per-use system is based on the WebSOS DoS protection architecture, extended to include OTPchecks, a light-weight and flexible pay-per-use micropayment scheme. Its hardware and software deployment can be done without changing any of the current ISP infrastructure. The initial investment and maintenance cost can regulated and scaled depending on the actual services protected.

From the end user perspective, the system acts almost transparently: no modifications are required in the browsers since we are taking advantage of browser extensibility. Moreover, the target site offering the web service can have a more fine-grained control of the users that it serves without altering any of its current servers' protocols. Finally, we allow a web service to charge its clients for the DoS protection service or provide the service as an added value feature.

References

1. D. G. Andersen. Mayday: Distributed Filtering for Internet Services. In *4th USENIX Symposium on Internet Technologies and Systems USITS*, March 2003.
2. S. M. Bellovin. Distributed Firewalls. *;login: magazine, special issue on security*, pages 37–39, November 1999.
3. W. J. Blackert, D. M. Gregg, A. K. Castner, E. M. Kyle, R. L. Hom, and R. M. Jokerst. Analyzing Interaction Between Distributed Denial of Service Attacks and Mitigation Technologies. In *Proceedings of DISCEX III*, pages 26–36, April 2003.
4. M. Blaze, J. Feigenbaum, J. Ioannidis, and A. D. Keromytis. The KeyNote Trust Management System Version 2. RFC 2704, September 1999.
5. M. Blaze, J. Ioannidis, and A. D. Keromytis. Offline Micropayments without Trusted Hardware. In *Proceedings of the Fifth International Conference on Financial Cryptography*, pages 21–40, 2001.
6. CCITT. *X.509: The Directory Authentication Framework*. International Telecommunications Union, Geneva, 1989.
7. D. Chaum. Achieving Electronic Privacy. *Scientific American*, pages 96–101, August 1992.
8. B. Cox, D. Tygar, and M. Sirbu. NetBill security and transaction protocol. In *Proceedings of the First USENIX Workshop on Electronic commerce*. USENIX, July 1995.
9. D. Dean, M. Franklin, and A. Stubblefield. An Algebraic Approach to IP Traceback. In *Proceedings of the Network and Dsitributed System Security Symposium (NDSS)*, pages 3–12, February 2001.
10. D. Dean and A. Stubblefield. Using client puzzles to protect TLS. In *Proceedings of the 10th USENIX Security Symposium*, August 2001.
11. V. D. Gligor. Guaranteeing Access in Spite of Distributed Service-Flooding Attacks. In *Proceedings of the Security Protocols Workshop*, April 2003.
12. M. T. Goodrich. Efficient Packet Marking for Large-Scale IP Traceback. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS)*, pages 117–126, November 2002.
13. A. Herzberg. Safeguarding Digital Library Contents. *D-Lib Magazine*, January 1998.
14. A. Hussain, J. Heidemann, and C. Papadopoulos. A Framework for Classifying Denial of Service Attacks. In *Proceedings of ACM SIGCOMM*, August 2003.
15. J. Ioannidis and S. M. Bellovin. Implementing Pushback: Router-Based Defense Against DDoS Attacks. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, February 2002.
16. J. Ioannidis, S. Ioannidis, A. D. Keromytis, and V. Prevelakis. Fileteller: Paying and Getting Paid for File Storage. In *Proceeding of Financial Cryptography (FC) Conference*, pages 282–299, March 2002.
17. S. Ioannidis, A. Keromytis, S. Bellovin, and J. Smith. Implementing a Distributed Firewall. In *Proceedings of Computer and Communications Security (CCS)*, pages 190–199, November 2000.

18. C. Jin, H. Wang, and K. G. Shin. Hop-Count Filtering: An Effective Defense Against Spoofed DoS Traffic. In *Proceedings of the 10th ACM International Conference on Computer and Communications Security (CCS)*, pages 30–41, October 2003.
19. D. Karger, E. Lehman, F. Leighton, R. Panigrahy, M. Levine, and D. Lewin. Consistent Hashing and Random Trees: Distributed Caching Protocols for Relievig Hot Spots on the World Wide Web. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 654–663, May 1997.
20. F. Kargl, J. Maier, and M. Weber. Protecting web servers from distributed denial of service attacks. In *World Wide Web*, pages 514–524, 2001.
21. S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. RFC 2401, Nov. 1998.
22. A. D. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure Overlay Services. In *Proceedings of ACM SIGCOMM*, pages 61–72, August 2002.
23. 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 10th ACM International Conference on Computer and Communications Security (CCS)*, pages 8–19, October 2003.
24. G. Mori and J. Malik. Recognizing Objects in Adversarial Clutter: Breaking a Visual CAPTCHA. In *Computer Vision and Pattern Recognition CVPR'03*, June 2003.
25. C. Papadopoulos, R. Lindell, J. Mehringer, A. Hussain, and R. Govindan. COSSACK: Coordinated Suppression of Simultaneous Attacks. In *Proceedings of DISCEX III*, pages 2–13, April 2003.
26. K. Park and H. Lee. On the Effectiveness of Route-based PAcket Filtering for Distributed DoS Attack Prevention in Power-law Internets. In *Proceedings of ACM SIGCOMM*, pages 15–26, August 2001.
27. T. Poutanen, H. Hinton, and M. Stumm. NetCents: A Lightweight Protocol for Secure Micropayments. In *Proceedings of the Third USENIX Workshop on Electronic Commerce*. USENIX, September 1998.
28. P. Reiher, J. Mirkovic, and G. Prier. Attacking DDoS at the source. In *Proceedings of the 10th IEEE International Conference on Network Protocols*, November 2002.
29. S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Network Support for IP Traceback. *ACM/IEEE Transactions on Networking*, 9(3):226–237, June 2001.
30. 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.
31. L. Tang. A Set of Protocols for MicroPayments in Distributed Systems. In *Proceedings of the First USENIX Workshop on Electronic Commerce*. USENIX, July 1995.
32. R. Thomas, B. Mark, T. Johnson, and J. Croall. NetBouncer: Client-legitimacy-based High-performance DDoS Filtering. In *Proceedings of DISCEX III*, pages 14–25, April 2003.
33. L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. CAPTCHA: Using Hard AI Problems For Security. In *Proceedings of EUROCRYPT'03*, 2003.
34. A. Yaar, A. Perrig, and D. Song. Pi: A Path Identification Mechanism to Defend against DDoS Attacks. In *Proceedings of the IEEE Symposium on Security and Privacy*, May 2003.