WRAPS: Denial-of-Service Defense through Web Referrals

XiaoFeng Wang[*] Michael K. Reiter[†]

Abstract

The web is a complicated graph, with millions of websites interlinked together. In this paper, we propose to use this web sitegraph *structure to mitigate flooding attacks on a website, using a new* web referral architecture for privileged service *("WRAPS"). WRAPS allows a legitimate client to obtain a* privilege URL *through a click on a referral hypherlink, from a website trusted by the target website. Using that URL, the client can get privileged access to the target website in a manner that is far less vulnerable to a DDoS flooding attack. WRAPS does not require changes to web client software and is extremely lightweight for referrer websites, which eases its deployment. The massive scale of the web sitegraph could deter attempts to isolate a website through blocking all referrers. We present the design of WRAPS, and the implementation of a prototype system used to evaluate our proposal. Our empirical study demonstrates that WRAPS enables legitimate clients to connect to a website smoothly in spite of an intensive flooding attack, at the cost of small overheads on the website's ISP's edge routers.*

1 Introduction

The web is a complicated referral graph, in which a node (website) refers its visitors to others through hyperlinks. In this paper, we propose to use this graph (called a *sitegraph* [37]) as a resilient infrastructure to defend against distributed denial-of-service (DDoS) attacks that plague websites today. Suppose eBay allows its trusted *neighbors* (websites linking to it) such as PayPal to refer legitimate clients to its privileged service through a privileged referral channel. A trusted client need only click on a privileged referral hyperlink on Paypal to obtain a *privilege URL* for eBay, which certifies the client's service privilege. When eBay is undergoing a DDoS attack and not accessible directly, routers in its local network will drop unprivileged packets to protect privileged clients' flows. As such, a client being referred can still access eBay even during the attack. Referral relations can be extended over the sitegraph: e.g., PayPal may refer its neighbors' clients to eBay. In this way, a website could form a large-scale referral network to fend off attack traffic.

The architecture we propose to achieve such privileged referrals, which we refer to as the "web referral architecture for privileged service" or "WRAPS", is built upon *existing* referral relationships among websites. Incentives for deployment, therefore, are not a significant barrier, provided that the overhead of the referral mechanism is negligible. Indeed, a website that links to others provides a better experience to its own customers if the links it offers are effective, and so websites have an incentive to serve privileged URLs for the sites to which they link. The overheads experienced by this website's users will be either nonexistent if the website offers privileged referrals to only customers that have already authenticated for other reasons, or minimal if the website will refer any client after it demonstrates it is driven by a human user (in the limit, asking the user to pass a reverse Turing test or "CAPTCHA" [34]). As we will show, the referrer incurs only negligible costs in order to make referrals via our technique.

In order to evaluate WRAPS, we implemented it in an experimental network environment which includes a software router (Click [21, 20]) and Linux-based clients and servers. Our empirical study shows that WRAPS enables clients to circumvent a very intensive flooding attack against a website, and induces reasonable costs on both edge routers and referral websites. In contrast to other capability-based approaches [40, 5], WRAPS does not require installing *anything* on a Web client. We explore the pertinence of web topology to the efficacy of WRAPS. We also describe a simple mechanism which helps a website acquire many referral sites at a low cost and helps legitimate clients obtain referrals seamlessly.

2 Related work

Numerous DDoS countermeasures have been proposed in the last decade [23, 16, 13, 24, 12, 22, 32, 17, 39, 9, 7, 27, 29, 11, 3, 28, 18, 35, 36]. In this section, we focus on the mechanisms which are most related to our proposal, in

[*] School of Informatics and Department of Computer Science, Indiana University at Bloomington; xw7@indiana.edu

[†] Electrical & Computer Engineering and Computer Science Departments, Carnegie Mellon University; reiter@cmu.edu

25th IEEE Symposium on Reliable Distributed Systems (SRDS'06)
0-7695-2677-2/06 $20.00 © 2006 IEEE

particular, overlay-based and capability-based approaches, and compare them with WRAPS.

Overlay networks have been applied to proactively defend against DoS attacks. Keromytis et al. propose a *secure overlay services* (SOS) architecture [19], which has been generalized by Anderson [4] to take into account different filtering techniques and overlay routing mechanisms. Morein et al. further propose to protect web services using SOS [25]. In these approaches, an overlay network is composed of a set of nodes arranged in a virtual topology. The routers around the protected web server admit http traffic from only trusted locations known to overlay nodes. A client who wants to connect to the web server has to first pass a CAPTCHA posed by an overlay node, which then tunnels the client's connection to an approved location so as to reach the web server. Similar work is that of Adkins et al. [2], which employs the Internet Indirection Infrastructure ($i3$) to enable the victim to selectively stop individual flows by removing senders' forwarding pointers.

WRAPS differs from overlay-based approaches in several important ways. First, these approaches assume the existence of an overlay infrastructure in which a set of dedicated nodes collaborate to protect an important website, and need to modify protocols and client-side software. This could introduce substantial difficulties for deployment. WRAPS, however, asks only referral websites to offer a lightweight referral service, which allows WRAPS to take advantage of *existing* referral relationships on the web to protect important websites. WRAPS also alters neither protocols nor client software. Second, overlay routing could increase end-to-end latency [19]. WRAPS does not change packets' routing paths and thus avoids such overhead.

Recently researchers have studied capability-based approaches that authorize a legitimate client to establish a privileged communication channel with a server using a secret token (capability). Anderson et al. present an infrastructure from which a client can obtain a capability to send packets to a server [5]. Yaar et al. designed an approach called SIFF that utilizes a client's secret path ID as its capability for establishing a privileged channel with a receiver [40]. Yang et al. propose an DoS-limiting Internet architecture which improves SIFF [41]. Gligor proposes to implement end-to-end user agreement [15] to protect connections against flooding attacks on the TCP layer.

Similar to these approaches, WRAPS also uses capability tokens to admit traffic. However, WRAPS focuses on important challenges that have not been addressed previously. First, the capability distribution service itself could be subject to DDoS attacks [6]. Adversaries may simply saturate the link of the server distributing capabilities to prevent clients from obtaining capabilities. The problem becomes especially serious in an open computing environment, where a service provider may not know most of its clients beforehand. WRAPS distributes capabilities through a referral network. In many cases, the scale of this referral network will make it difficult for the attacker to block clients from being referred to a target website. Second, existing capability-based approaches require modifications to client-side software, which WRAPS avoids.

A fundamental question for both overlay-based and capability-based mechanisms is how to identify legitimate but unknown clients. So far, the only viable solution is a CAPTCHA, which works only in human-driven activities and has been subject to various attacks (e.g., [26]). Although WRAPS can also use CAPTCHA, essentially our approach relies on social-based trust to identify legitimate clients: a party can refer new clients to a target server; the server trusts them on the basis of its trust of the referrer. This offers a new and potentially more effective way to protect an open system against DoS attacks. For instance, suppose a group of universities agree to offer referral services for each other. Once one's website is subject to a flooding attack, the trusted users of other schools (those with proper university accounts) can still access the victim's website through WRAPS.

The idea of embedding a secret token inside part of an IP address and port number (as we do here) also appeared in a DoS defense mechanism proposed by Xu et al. [38] However, they use this approach only as an extension of syn-cookies, for the purpose of detecting flows with spoofed IP addresses. Our approach can also be viewed as an extension of network address translation (NAT), taking part of the IP and port number fiels to hold an authorization secret.

3 Attack model

We assume that adversaries can modify at most a small fraction of the legitimate packets destined for the target website. Attackers capable of tampering with these packets on a large scale do not need to flood the target's bandwidth. Instead, they can launch a DDoS attack by simply destroying these packets.

We assume that adversaries cannot eavesdrop on most legitimate clients' flows. In practice, monitoring a large fraction of legitimate clients' flows is difficult in wide area networks. However, WRAPS still works when adversaries are capable of eavesdropping on some privileged clients' flows. In this case, a defender could control the damage caused by these clients through standard rate limiting.

We assume that routers inside a website's protection perimeter are trusted. In practice, routers usually enjoy better protection than end hosts. We further assume that a DoS flooding attack on a website does not significantly affect the flows from the website to clients. This is generally true in today's routers which employ full-duplex links and switched architectures.

4 Design

In WRAPS, a website grants a client greater privilege to access its service by assigning to it a secret fictitious URL called *privilege URL* (Section 4.1) with a capability token embedded in part of the IP and port number fields. Through that URL, the client can establish a privileged channel with that website (referred to as the *target website*) even in the presence of flooding attacks.

A client may obtain a privilege URL either directly from the target website or indirectly from the website's trusted neighbors. A website offers a client a privilege URL if the client is referred by one of the site's *trusted* neighbors, or is otherwise qualified by the site's policies that are used to identify valued clients, for example, those who have paid or who are regular visitors. A qualified client will be redirected to the privilege URL generated automatically using that client's identity, service information and a server secret.

A privilege URL leads its holder to the target website through a *protection mechanism* (Section 4.2) which protects the website from unauthorized flows. The border of this mechanism is the site's ISP's edge routers, which classify traffic into privileged and unprivileged flows, and translate fictitious addresses in privilege URLs into the website's real address. Within the protection perimeter, routers protect privileged traffic by dropping unprivileged packets during congestion.

A neighbor website refers a trusted client to the target website's privileged service. The referral is done through a simple proxy script running on the referrer site, from which the client acquires a redirection instruction leading to the privilege URL. This is discussed in Section 4.3.

4.1 Privilege URLs

Resources available on the Internet are located via Uniform Resource Locators (URL). An http URL is of the following format: `http://⟨host⟩:⟨port⟩/⟨urlpath⟩`, where the `host` and `port` fields could be the (IP, port) pair of an http service on the Internet, which are accessible to routers.

In WRAPS, we utilize privilege URLs to set up priority channels. These URLs are fictitious because they do not directly address a web service. Instead, they contain secret *capability tokens* which are verified by edge routers for setting priority classes, and unambiguously translated by these routers to the real location of the service. A privilege URL hides a capability token inside the suffix of the destination IP address (last one or two octets) and the whole destination port field. The following fields are present in the token.

- **Key bit** (1 bit). This field is used to indicate the "authentication key" currently in use (see Section 4.2).
- **Priority field**. This is an optional field which allows the website to define more than one service priority. Here, we use one priority class (i.e., a request is either privileged or unprivileged) to describe our approach for clarity of presentation, and so this field is unecessary and omitted.
- **Message authentication code**. A message authentication code (MAC) prevents adversaries from forging a capability. The algorithm computing a MAC over a message takes as inputs a secret key k and the message to produce a w-bit tag. MAC generation is based on a cryptographically-strong pseudorandom function (PRF) so that the probability to compute the right tag without knowing k is negligibly larger than 2^{-w}. For a privileged client i, its MAC is denoted by $MAC(k, IP_i)$, where IP_i is i's IP address.

Encoding a capability token into the destination IP and port fields limits the length of MAC, especially for IPv4. For example, a Class C network may be able to support only a 16 to 20-bit MAC. This seems to make WRAPS vulnerable, allowing an adversary to forge a capability token through a brute-force search. As we will show, however, WRAPS contains a mechanism that effectively mitigates this threat: any adversary without global eavesdropping capability will be unable to confirm its guess of a MAC value. We present a detailed security analysis in Section 4.4.

It is possible that some clients' fictitious (IP, port) pairs coincide with a real application in the local network. However, this happens with a very small probability with the MAC in place. One approach to prevent this problem with certainty is to reduce the address range that can be mapped to the web server, i.e., by reducing the MAC length, so that this range does not intersect other servers. Another choice is to use the most significant bit on the port field as a "token indicator", by which edge routers can identify the packets that need address translation.

4.2 Protection mechanism

A website (the target) is protected by the edge routers of its ISP or organization, the routers inside its local network, and a firewall directly connected to or installed on the site's web server.

The target website shares a secret long-term key k with its edge routers on the protection perimeter. Using this key, the website periodically updates to all its edge routers a shared verification key. We call a period between updates a *privilege period*. Specifically, the verification key used in the privilege period t is computed as $k(t) = h_k(t)$ where h is a pseudorandom function family indexed by the key k.

The http server of the website listens to two ports, one privileged and one not. The local firewall controls access to those ports. Only the port corresponding to the unprivileged traffic, typically port 80, is publicly accessible. The other port can be accessed only by packets with source IP addresses explicitly permitted by the firewall (as instructed by the web server); this port is called the *privilege port*.

Below we describe a protection mechanism which allows a client to acquire a privilege *directly* from a website and establish a privileged channel with that website.

Privilege acquisition

1. A client that desires privileged service from a website first applies for a privilege URL online. This application process is site-specific and so we do not discuss it here, but we presume that the client does this as part of enrolling for site membership, for example, and before an attack is taking place.

2. In period t, if the website decides to grant privilege to a client i, it first interacts with the firewall to put i's source IP address IP_i onto a *whitelist* of the privilege port. It then constructs a privilege URL containing a capability token $\tau_i(t) = b_t \| MAC(k(t), IP_i)$, where b_t is the one-bit key field for period t. The website uses the standard http redirection to redirect the client to this privilege URL.

Privileged channel establishment

1. Edge routers drop packets to the website addressed to the privilege port of that website.

2. According to the position and the length λ of a capability token, an edge router takes the appropriate string θ of λ bits from every TCP packet. Denote a substring from the e_1-th bit to the e_2-th bit on θ by $\theta[e_1, e_2]$. A router processes a packet from a client i as follows.

```
If ($\theta[2, \lambda] = MAC(k(t), IP_i)$) then
    Translate the fictitious destination IP address to
        the target website's IP address.
    Set the destination port number of the packet to
        the privilege port.
    Forward the packet.
else
    Forward the packet as an unprivileged packet.
```

3. Routers inside the protection perimeter forward the packets toward the target website, prioritized according to the ports of these packets. [1]

4. Upon receiving a packet with a source IP address IP_i and destined for the privilege port, the firewall of the website checks whether IP_i is on the port's whitelist. If not, the firewall drops the packet.

5. Using the secret key, the web server or firewall translates the source IP and port of every packet emitted from the privilege port to the fictitious (IP, port) containing the capability token. Note that no state information except the key needs to be kept to perform this translation.

To allow the update of privileged URLs to clients, there exists a transition period between privilege periods t and $t + 1$ during which both $k(t)$ and $k(t + 1)$ are in use. This transition period could be reasonably long to allow most privileged clients (who browse that website sufficiently frequently) to visit. Once visited by a privileged client i in this transition period, the website generates a new privilege URL using $k(t + 1)$ and with key field b_{t+1} equal to the parity of $t + 1$. The website then redirects the client to that URL. Also upon entering the transition period, the website and edge routers record a single bit b indicating the parity of $t+1$. If during the transition period, the edge router receives a packet with key field $b_t = 1 - b$, the edge router uses $k(t)$ in the MAC computation; otherwise it uses $k(t + 1)$.

A website can remove a client's privilege by not updating its privilege URL. Standard rate-limiting technology can also be used to control the volume of traffic produced by individual *privileged* clients in case some of them fall prey to an adversary. An option to block a misbehaving privileged client within a privilege period is to post that client's IP to a blacklist held by the website's ingress edge routers. This does not have to be done in a timely manner, as the rate-limiting mechanism is already in place. In addition, an adversary cannot fill the router blacklist using spoofed IPs because the blacklist here only records the misbehaving *privileged* clients who are holding the correct capability tokens. This blacklist is emptied periodically after the router updates its verification key.

4.3 Referral protocol

When a website is under a flooding attack, legitimate but as-yet-unprivileged clients will be unable to visit that site directly, even merely to apply for privileged service. The central idea of WRAPS is to let the trusted neighbors of the website refer legitimate clients to it, even while the website is under a flooding attack. The target website will grant these trusted neighbors privilege URLs, and may allow transitive referrals: a referrer can refer its trusted neighbor's clients to the target website.

A privilege referral is done through a simple proxy script running on the referrer website. A typical referrer website is

[1] One way to do this is to set priority queues for different (IP, port) pairs of the website, which can be easily configured in a modern router.

one that linked to the target website originally and is willing to upgrade its normal hyperlink to a *privilege referral link*, i.e., a link to the proxy. This proxy could be extremely simple, containing only a few dozen lines of Perl code, and very lightweight in performance. The proxy communicates with the target server through the referrer's privilege URL to help a trusted client to acquire its privilege URL. The target server publicizes an online list of all these referrers, the neighbors it trusts (Section 7), and only accepts referrals for privileged service from these websites.

Only clients trusted by referrers are given privilege referral links. Such trusted clients could be those authenticated to the referrer website for other purposes, those referred from the referrer's trusted neighbors, or those proved to be driven by a human through CAPTCHA tests. In other words, these clients must be "authenticated" somehow by the referrer.

Below, we describe a simple referral protocol.

1. A client i trusted by a referrer website r clicks on a privilege referral link offered by r, which activates a proxy on the referrer site.

2. The proxy generates a reference including the client's IP address IP_i and sends the reference to the target server through a privileged channel established by $purl_r$, r's privilege URL to the target website. As we discussed in the previous section, edge routers of the target website will authenticate the capability token in $purl_r$.

3. Upon receiving r's reference, the target website checks its list of valid referrers. If r's IP does not appear on the list, the website ignores the request. Otherwise, it generates a privilege URL $purl_i$ for client i using IP_i, embeds it into an http redirection command and sends the command to referrer r.

4. The proxy of referrer r forwards the redirection command to client i.

5. Running the redirection command, client i's browser is automatically redirected to $purl_i$ to establish a privileged channel *directly* with the target website.

An important issue is how to contain trusted but nevertheless compromised referrers who might introduce many zombies to deplete the target website's privileged service. One mitigation is to prioritize the privileged clients: those who are referred by a highly trusted referrer have higher privileges than those from a less trusted referrer. WRAPS assures that the high priority traffic can evade flooding attacks on the low priority traffic. Within one priority level, WRAPS rate limits privileged clients' traffic. The target website can also fairly allocate referral quotas among its trusted neighbors. This, combined with a relatively short privilege period for clients referred from referrer websites, could prevent a malicious referrer from monopolizing the privileged channel. The target website may update a *reputation* value for each of its trusted neighbors. A malicious privileged client detected will be traced back to its referrer, whose reputation will be negatively affected.

4.4 Brute-force attacks on a short capability token

An adversary may perform an exhaustive search on the short MAC in a privilege URL. Specifically, the adversary first chooses a random MAC to produce a privilege URL for the target website. Then, it sends a TCP packet to that URL. If the target website sends back some response, such as syn-ack or reset, the adversary knows that it made a correct guess. Otherwise, it chooses another MAC and tries again.

This threat has been nullified by our protection mechanism. The firewall of the target website keeps records of all the website's privileged clients and only admits packets to privilege port from these clients. If the adversary uses its real IP address for sending probe packets, the website will not respond to the probe unless the adversary has already become the site's privileged client. If the adversary spoofs a privileged client's IP address to penetrate the firewall, the website's response only goes to that client, not the adversary. Therefore, the adversary will never know whether it makes a correct guess or not.

We reiterate that this approach does not introduce new vulnerabilities. Only a client trusted by the target website directly or referred by trusted neighbors will have its IP address on the firewall's whitelist. The storage for the whitelists is negligible for a modern computer: recording a million clients' IP addresses takes only 4 MB. Therefore, our approach does not leave an open door to other resource depletion attacks. The performance of filtering can also be scaled using fast searching algorithms, for example, Bloom filters [8] that are capable of searching for entries at link speed. To prevent a compromised referrer from attacking the whitelist, a target website can set the quota of the number of referrals a referrer can make in one privilege period.

5 Implementation

To evaluate the design of WRAPS, we implemented it within an experimental network with Class C IP addresses. We utilized SHA-1 to generate a MAC [33] for privilege URLs. The web server we intended to protect had a Linux OS and an *Apache* http server that was configured to listen to both port 80 and 22433. The latter was the privilege port.

We built the protection mechanism into a Click software router [21, 20] and the TCP/IP protocol stack of the target's kernel, using Linux *netfilter* [1] as the firewall. We constructed the referral protocol on the application layer, on the basis of simple cgi programs running on the referrer and the target websites. In this section, we elaborate on these implementations.

5.1 Implementation of the protection mechanism

WRAPS depends on the target's ISP's edge routers to classify inbound traffic into privileged and unprivileged, and to translate fictitious addresses of privileged packets to the real location of the service. It also depends on routers inside the website's local network to protect privileged flows. We implemented these functions in a Click software router.

Click is a modular software architecture for creating routers [21, 20]. A Click router is built from a set of packet processing modules called elements. Individual elements implement certain router functions such as packet classification, queuing and scheduling. A router is configured as a directed packet-forwarding graph, with elements on its vertices and packet flows traversing its edges. A prominent property of Click is its modularity, which makes a Click router extremely easy to extend.

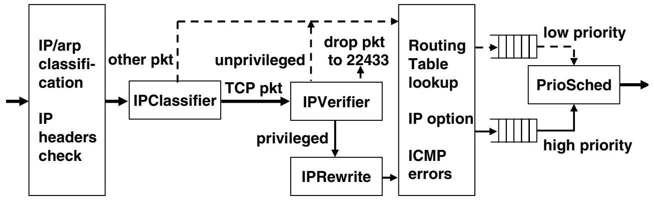

Figure 1. WRAPS elements on a Click packet forwarding path.

We added WRAPS modules to an IP router configuration [21, 20]. WRAPS elements planted in the standard IP forwarding path are illustrated in Figure 1. We added 5 elements: IPClassifier, IPVerifier, IPRewrite, Priority queue and PrioSched. IPClassifier classifies all inbound packets into TCP packets which are forwarded to IPVerifier, and other packets, such as UDP and ICMP, which are forwarded to the normal forwarding path. IPVerifier verifies every TCP packet's capability token embedded in the last octet of the destination IP address and the 2-octet destination port number. Verification of a packet invokes the MAC over a 4-byte input (the source IP address) and a 64-bit secret key. The packets carrying correct capability tokens are sent to IPRewrite, which sets a packet's destination IP to that of the target website and destination port to port 22433. Unprivileged packets, except for those destined for port 22433, follow UDP and ICMP traffic; unprivileged traffic to port 22433 is dropped.

Both privileged and unprivileged flows are processed by some standard routing elements. Then, privileged packets are queued into a high priority queue while other packets flow into a low priority queue. A PrioSched element is used to multiplex packets from these two queues to the output network interface card (NIC). PrioSched is a strict priority scheduler, which always tries the high priority queue first and then the low priority one, returning the first packet it finds [21, 20]. This ensures that privileged traffic receives service first. Though we explain our implementation here using only two priority classes, the whole architecture can be trivially adapted to accommodate multiple priority classes.

On the target server front, *netfilter* is used to filter incoming packets. Netfilter is a packet filtering module inside the Linux kernel. Users can define their filtering rulesets to the module through *iptables*. In our implementation, the target website first blocks all access to its privilege port. Whenever a new client obtains privilege, a cgi script running on the web server adds a new filter rule to iptables, explicitly permitting that user's access to the privilege port. Direct utilization of iptables may potentially cause the performance degradation of netfilter when there are many privileged clients. Our general approach, however, could still scale well after proper modification of the netfilter module, adding a fast searching algorithm like Bloom filters to the kernel.

To establish a privileged connection, packets from the target web server to a privileged client must bear the fictitious source address and port in that client's privilege URL. In our implementation, we modified the target server's Linux kernel to monitor the privilege port (22433). Whenever a packet is emitted with that source port, the kernel employs the secret key and the MAC to generate a capability token, and embeds this token into last octet of the source IP field and the source port field. This address translation can also be done in the firewall, and configured to support more than two priority classes.

5.2 Implementation of the referral protocol

The referral protocol is performed by two simple scripts running on the referrer and the target websites. The script on the referrer acts as a proxy which is activated through privilege referral links accessible to the trusted clients of that website. A privilege referral link is a simple replacement of a normal hyperlink. For example, a normal hyperlink to eBay (`http://www.ebay.com`) can be replaced with a privilege hyperlink on Paypal (http://www.paypal.com/cgi-bin/proxy.pl?http://www.ebay.com), where “proxy.pl” is the proxy written in perl. Clicking on that hyperlink, a client triggers the proxy which in turn invokes a cgi script on the target website through the referrer’s privilege URL, conveying the client’s source IP address as a parameter.

The cgi script on the target website first checks whether the proxy is entitled to make such a referral by searching its referrer list. If the referrer is on the list, the script inputs a filtering rule to iptables to permit the access of the client being referred, generates a privilege URL and then sends to the referrer proxy a new webpage containing an http redirection command to the new URL. Here is an example of the redirection command: <meta HTTP-EQUIV="Refresh" CONTENT="1; URL=http://A.B.C.τ/index.htm">, where τ is a capability token and “A.B.C” is the IP prefix of the target website.

Receiving the webpage with the redirection command, the proxy relays it to the client. Interpreting the page, the client’s browser will be automatically redirected to the target website through a privileged channel, and communicate with the website directly afterwards.

6 Empirical Evaluation

In this section, we report our empirical evaluation of WRAPS in an experimental network composed of a set of Linux servers, each with up to 2.8 GHz CPU and 1 GB memory. The objectives of this study are: (1) evaluation of the overheads of WRAPS, both on edge routers (Section 6.1) and referrer websites (Section 6.3); and (2) testing the performance of WRAPS under DoS flooding attacks (Section 6.2). We elaborate our experimental results in the following sections.

6.1 Overhead on edge router

The target server’s ISP’s edge routers play an important role in WRAPS, undertaking the tasks of classifying packets, verifying capability tokens, and translating addresses. An important question is whether the overheads of these tasks, especially computing a MAC for every TCP packet, could be afforded by an edge router. We investigated this problem by comparing the packet forwarding capabilities of a Click router with and without WRAPS elements.

In this experiment, we connected two computers to a router (a computer installed with Click software router) through two Gigabit NICs. One of them generated a constant and high-rate flow of 64-byte UDP packets, and the other received these packets from the router. We utilized Click’s UDP traffic generator [20] as a traffic source; it works in the Linux kernel and is capable of generating more traffic more evenly than a user-level program. Our original design does not verify UDP, though to test the performance of IPVerifier, we had the router check the MAC on every UDP packet in this test according to the last octet of the destination IP address and the port number.

Using a Pentium-4 2.6GHz computer as the router, we observed a maximal forwarding rate of 350k packets per second (pps) over the standard packet forwarding path. Surprisingly, this rate did not change after we added in WRAPS elements. This could result from the constraints of hardware: the 1GB memory and PCI bus of the router might become performance bottlenecks before the CPU did. When this happens, the router reaching its performance limits might still have sufficient CPU cycles to check the MAC for every packet. Such a conjecture was confirmed after we moved the software router to a slower computer (Pentium-3 800MHz). This time, a difference emerged: we observed 290k pps for the normal forwarding path[2] and 220k pps for the one with WRAPS elements.

The experimental results show that verification of the MAC affected the performance of edge routers. However, such overheads seem to be affordable. In a Pentium-3 computer, running SHA-1 on a 4-byte input and an 8-byte key takes about 1.39 microseconds, which is reduced to 0.33 microseconds in a Pentium-4 system. Since IPVerifier performs a MAC on just a few bytes of every packet, the rate of verification seems able to keep up with the forwarding rate.

Moreover, this performance could be improved through hand-tuning the functions used to implement the MAC. For example, an optimized AES program written in assembly code is reported to be able to work at 17 cycles/byte over a Pentium-3 system [14]. In our setting, this might lead to over one million pps even with a Pentium-3. A hardware implementation of IPVerifier will be even faster. Therefore, we tend to believe that the cost of WRAPS should be affordable in practice.

6.2 Performance under flooding attacks

We evaluated the performance of WRAPS under intensive bandwidth-exhaustion attacks. Our experimental setting included 6 computers: a Pentium-4 router was linked to three Pentium-4 attackers and a legitimate client through four Gigabit interfaces, and to a target website through a 100Mb interface. We deliberately used Gigabit links to the “outside” to simulate a large group of ISP’s edge routers which continuously forward attack traffic to a link on the path toward the website.

Under the above network setting, we put WRAPS to the test under UDP and TCP flooding attacks. In the UDP

[2] The Click project reported a faster forwarding speed (about 350k pps) over a similar hardware setting [20]. This could be because they used a “simple” forwarding path to test the router, without processing IP checksum, option, fragmentation and ICMP errors.

flooding test, we utilized Click's UDP generators to produce attack traffic. Three attackers attached to the router through Gigabit interfaces were capable of generating up to 1.5Mpps of 64-byte UDP packets, which amounts to 0.75Gbps. On the other hand, the 100Mb channel to the web server could only sustain up to 148,800pps of 64-byte packets. As such, the flooding rate could be set to 10 times as much as the victim's bandwidth, and so these attackers could easily saturate the victim's link.

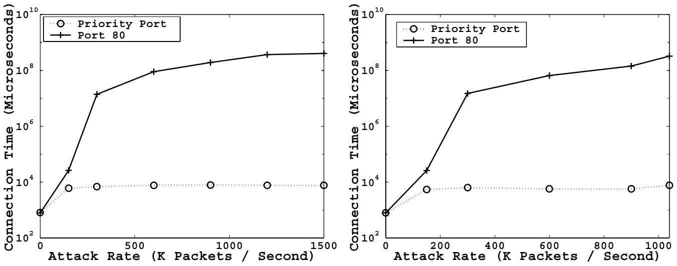

Figure 2. Left: UDP flooding; Right: TCP flooding

On the legitimate client, a test program continuously attempted to connect to the target website, either to port 80 or to a privileged URL which was translated to port 22433 by the router, until it succeeded. The overall waiting time for a connection attempt is called *connection time*. We computed the *average connection time* over 200 connection attempts. Our experiment compared the average connection times of unprivileged connections (to port 80) with those of privileged connections (through capability tokens). Figure 2 describes the experiment results. Note that the scale of Y-axis is exponential.

As illustrated in the figure, average connection times of both normal and privileged channels jumped when the attack rate hits 150kpps, roughly the bandwidth of the target website's link. Above that, latency for connecting to port 80 kept increasing with the attack rate. When the attack rate went above 1Mpps, these unprivileged connections no longer had any reasonable waiting times; e.g., a monstrous 404 seconds was observed at 1.5Mpps. Actually, under such a tremendous attack rate, we found it was extremely difficult to get even a single packet through: an attempt to ping the victim server was effectively prevented, with 98% of the probe packets lost. Connections through the privileged channel, however, went very smoothly: the average connection delay stayed below 8 milliseconds while the attack rate went from 150kpps to 1.5Mpps. Comparing this with the latency when there was no attack, we observed a decent increase (0.8ms to 8ms) for the connections under WRAPS protection, versus a huge leap (0.8ms to 404,000ms) for unprotected connections.

A TCP-based flooding attack differs from UDP flooding in that TCP packets will go through the IPVerifier on the router, potentially adding more cost to forwarding. In our experiment, TCP-flooding traffic was provided by an attack program which generates packets through socket system calls as a typical DDoS attack tool does. This application-level generator cannot work as fast as the Click UDP generator does in the kernel. We got a peak rate of 1.14Mpps, which nevertheless was enough to deplete a 100Mb channel. Figure 2 presents the experimental results. Similar to what we observed in the UDP flooding, privileged connections effectively circumvented flooding flows, with the worst-case delay of about 7.7 milliseconds. On the other hand, the average connection time for unprotected clients was about 323 seconds.

6.3 Overheads on referrer website

We evaluated the performance of a referrer website when it is making referrals to a website under a flooding attack. Our experimental setting was built upon the setting for bandwidth-exhaustion attacks. We added two computers, along with the original client, and connected them through a 100Mb switch to the router. One of these three computers acted as a referrer web server and the other two were used as clients. On every client, a script simulated clicks on the referrer website's privilege referral link. It was capable of generating up to 100 concurrent referral requests. One client was also continuously making connections to the referral website to collect connection statistics. The attackers kept on producing TCP traffic of 1.14Mpps to saturate the target's link.

Through the experiment, we found that making referrals adds a negligible cost to the referrer website. Even in the situation when 200 users were simultaneously requesting referrals, the average latency for connecting to the referrer website only increased by less than 40 microseconds. Recall that only authenticated clients or those that passed CAPTCHA tests are allowed to click on a privilege referral hyperlink. Therefore, the rate of referral requests will not be extremely high in practice. In this case, it seems that the normal operation of the referrer website will not be affected noticeably by our mechanism.

7 Discussion

7.1 Web topology

A fundamental question for WRAPS is whether a website under DoS threats is able to find enough referral websites to protect it. In our research, we studied this problem using a real sitegraph, the .gov data collection distributed by CSIRO [10]. Due to space limits, here we sketch our findings.

We observed two important statistical properties from the sitegraph which are highly related to WRAPS. First, more important websites (with higher siteranks [37]) tend to have more neighbors. A neighbor website would be willing to offer referral service because referrals cost little in WRAPS and the referrer is expected to benefit from offering reliable links to its customers. Of course, one will not trust all its neighbors. However, we believe that important websites are more likely to have many trusted neighbors. Actually, our research discovered that an important site might also have many outbound links to those who link to it: on average, a top 10 website in the CSIRO dataset connects to about 334 neighbors. These outbound links are widely perceived as "trust transfer" from the important site to another site, as discovered by studies in operations research [30, 31]. In addition, an important website could also use its high siterank as an asset to attract many unknown small websites to protect it. Trust can be established in this case through some external means, for example, contract. The reward the important website offers to its referrers could be as small as a reward link. Such a link will make an unknown website look more trustworthy and help it to establish its reputation, with the trust "transferred" from the famous website [30, 31]. This could encourage many small websites to join efforts to protect an important site. Second, we found almost all important websites are linked together. Since referrals can be transitive, this property allows these websites to pool their referral websites to protect each other.

To facilitate search of referrers, a website can list all its referrer sites on a webpage to let search engines, such as Google, index and cache that page. Legitimate clients, therefore, can easily discover these referrers even during a DoS flooding attack.

7.2 Limitations

There are several limitations of WRAPS, mostly arising from the fact that it encodes privilege URLs as IP addresses instead of as domain names:

1. WRAPS supports only clients that use fixed (or infrequently changed) IP addresses. An extension of WRAPS that supports dynamic NAT users is to generate the client's capability using the client's IP prefix, instead of its whole IP address. This extends privileged service to clients using dynamic addresses in a subnet.

2. A user must access the target site using its privileged URL, if she wants privileged service. That is, the domain name of the server cannot be resolved via DNS, and moreover the user must save (e.g., bookmark) her privilege URL from the server when it is updated at the end of a privilege period. In these ways, WRAPS is not transparent to users, and would require client-side modifications to make it transparent.

3. WRAPS could also break SSL/TLS service because while privilege URLs are encoded as IP addresses instead of domain names, the target site certificate will typically certify a key for a site name, not its address. This, however, is not a problem for applications that perform reverse lookup on the IP address to find a domain name before verifying the SSL/TLS certificate. Another way to solve this problem is to confine the capability token to the port number field, with the only cost being some degradation in defense against floods with randomly spoofed addresses. For example, Paypal can redirect a client to `https://www.ebay.com:capability` to allow the client to establish a TLS session with eBay.

Finally, discovery of referrer websites is not transparent to clients in WRAPS. One way to mitigate this problem is to let a client's ISP be its referrer. For example, the websites of all academic institutions could agree to offer referral services to each other. Whenever an academic client sends a request to a server within an institution's domain, that request will be automatically redirected by the client's ISP (i.e., its institution) to the local referral website. Moreover, we might take a technique similar to dynamic DNS to allow a target website to dynamically map its domain name to its referrer sites' IP addresses when it is undergoing a DoS attack. We plan to further investigate these approaches in future research.

References

[1] The netfilter/iptables project. `http://www.netfilter.org`.

[2] D. Adkins, K. Lakshminarayanan, A. Perrig, and I. Stoica. Taming ip packet flooding attacks. In *Proceedings of Workshop on Hot Topics in Networks (HotNets-II)*, November 2003.

[3] M. Adler. Tradeoffs in probabilistic packet marking for IP traceback. In *Proceedings of 34th ACM Symposium on Theory of Computing (STOC-02)*, 2002.

[4] D. Andersen. Mayday: Distributed filtering for internet services. In *Proceeding of USITS*, 2003.

[5] T. Anderson, T. Roscoe, and D. Wetherall. Preventing internet denial-of-service with capabilities. In *Proceedings of Workshop on Hot Topics in Networks (HotNets-II)*, November 2003.

[6] K. Argyraki and D. R. Cheriton. Network capabilities: The good, the bad and the ugly. In *Proceedings of the 4th Workshop on Hot Topics in Networks*, Nov. 2005.

[7] S. Bellovin, M. Leech, and T. Taylor. The ICMP traceback messages. In *Internet-Draft, draft-ietf-itrace-01.txt*, December 1999. `ftp://ftp.ietf.org/internet-drafts/draft-ietf-itrace-01.txt`.

[8] B. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of ACM*, 13(7):422–426, 1970.

[9] H. Burch and B. Cheswick. Tracing anonymous packets to their approximate source. In *Proceedings of the 14th USENIX System Administration Conference*, Dec. 1999.

[10] CSIRO. Web Research Collections (TREC Web and Terabyte Track). `http://es.csiro.au/TRECWeb/`.

[11] D. Dean, M. Franklin, and A. Stubblefield. An algebraic approach to IP traceback. In *Proceedings of Network and Distributed System Security Symposium (NDSS-01)*, February 2001.

[12] P. Ferguson and D. Senie. RFC 2267: Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing, Jan. 1998. `ftp://ftp.internic.net/rfc/rfc2267.txt`.

[13] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the internet. *IEEE/ACM Transactions on Networking*, August 1999.

[14] B. Gladman. Aes and combined encryption/authentication modes. `fp.gladman.plus.com/AES/index.htm`.

[15] V. Gligor. Guaranteeing access in spite of service-flooding attack. In R. Hirschfeld, editor, *Proceedings of the Security Protocols Workshop*, Lecture Notes in Computer Science. Springer-Verlag, 2004.

[16] J. Ioannidis and S. Bellovin. Implementing pushback: Router-based defense against ddos attacks. In *Proceedings of the Symposium on Network and Distributed System Security (NDSS-02)*, 2002.

[17] C. Jin, H. Wang, and K. Shin. Hop-count filtering: An effective defense against spoofed traffic. In *Proceedings of ACM CCS*, 2003.

[18] A. Juels and J. Brainard. Client puzzle: A cryptographic defense against connection depletion attacks. In S.Kent, editor, *Proceedings of NDSS'99*, pages 151–165, 1999.

[19] A. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure overlay services. In *Proceedings of ACM SIGCOMM*, August 2002.

[20] E. Kohler. *The Click modular router*. MIT, November 2000. PhD thesis.

[21] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. Kaashoek. The click modular router. *ACM Transactions on Computer Systems*, 18(3), August 2000.

[22] J. Li, J. Mirkovic, and M. Wang. Save: Source address validity enforcement protocol. In *Proceedings of IEEE INFOCOM*, 2002.

[23] R. Mahajan, S. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker. Controlling high bandwidth aggregates in the network. *CCR*, 32(3):62–73, July 2002.

[24] R. Mahajan, S. Floyd, and D. Wetherall. Controlling high-bandwidth flows at the congested router. In *Proceedings of ICNP*, November 2001.

[25] W. Morein, A. Stavrou, D. Cook, A. Keromytis, V. Misra, and D. Rubenstein. Using graphic turing tests to counter automated ddos attacks against web servers. In *Proceedings of ACM CCS*, 2003.

[26] G. Mori and J. Malik. Recognizing objects in adversarial clutter: Breaking a visual CAPTCHA. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2003.

[27] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Network support for IP traceback. In *Proceedings of ACM SIGCOMM*, August 2000.

[28] A. Snoeren, C. Partridge, L. Sanchez, C. Jones, F. Tchakountio, S. Kent, and W. Strayer. Hash-based IP traceback. In *Proceedings of the ACM SIGCOMM*, August 2001.

[29] D. Song and A. Perrig. Advanced and authenticated marking schemes for IP traceback. In *Proceedings of IEEE INFOCOMM*, April 2001.

[30] K. J. Stewart. Trust transfer on the world wide web. *Organization Science*, 14(1), 2003.

[31] K. J. Stewart and Y. Zhang. Effects of hypertext links on trust transfer. In *ICEC '03: Proceedings of the 5th international conference on Electronic commerce*, pages 235–239, New York, NY, USA, 2003. ACM Press.

[32] R. Stone. An IP overlay network for tracking dos floods. In *Proceedings of USENIX Security Symposium*, 2000.

[33] G. Tsudik. Message authentication with one-way hash functions. In *Proceedings of Infocom*, 1992.

[34] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. In *Advances in Cryptology — EUROCRYPT 2003*. Springer-Verlag, 2003.

[35] X. Wang and M. Reiter. Defending against denial-of-service attacks with puzzle auctions. In *IEEE Symposium on Security and Privacy*, May 2003.

[36] X. Wang and M. Reiter. Mitigating bandwidth-exhaustion attacks using congestion puzzles. In *Proceedings of the 11th ACM conference on Computer and Communication Security*, November 2004.

[37] J. Wu and K. Aberer. Using siterank for p2p web retrieval. Technical Report IC/2004/31, Swiss Federal Institute of Technology, Lausanne, Switzerland, March 2004.

[38] J. Xu and W. Lee. Sustaining availability of web services under severe denial of service attacks. *IEEE Transaction on Computers, special issue on Reliable Distributed Systems*, 52(2):195–208, February 2003.

[39] A. Yaar, A. Perrig, and D. Song. Pi: A path identification mechanism to defend against DDoS attacks. In *IEEE Symposium on Security and Privacy*, May 2003. `http://www.ece.cmu.edu/~adrian/projects/pi.ps`.

[40] A. Yaar, A. Perrig, and D. Song. An endhost capability mechanism to mitigate DDoS flooding attacks. In *Proceedings of the IEEE Symposium on Security and Privacy*, May 2004.

[41] X. Yang, D. Wetherall, and T. Anderson. A dos-limiting network architecture. In *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 241–252, New York, NY, USA, 2005. ACM Press.