Fine-Grained Capabilities for Flooding DDoS Defense Using Client Reputations

Maitreya Natu
University of Delaware
103 Smith Hall
Newark, DE 19716, USA
natu@cis.udel.edu

Jelena Mirkovic
University of Delaware
103 Smith Hall
Newark, DE 19716, USA
sunshine@cis.udel.edu

ABSTRACT

Recently proposed capability mechanisms offer one part of the answer to the DDoS problem. They empower the victim to control the traffic it receives by selectively granting access to well-behaved clients via short-lived tickets. One major question still remains unanswered: how can victims distinguish between well-behaved and ill-behaved clients during the ticket-granting process. This paper offers one possible answer to this question, while also refining the basic capability mechanism.

We propose the following novel features: (1) Reputation-based ticket-granting – long-term behavior of a client influences whether future tickets will be granted, (2) Fine-grained capabilities, which authorize access to the victim at a specified priority level based on a client's prior behavior, (3) Destination-based capabilities, granted by the defense located at the victim; this reduces operational cost, and breaks dependence of tickets on routes.

Categories and Subject Descriptors: K.6.5 Management of Computing and Information Systems: Security and Protection
General Terms: Management, Measurement, Security.
Keywords: Distributed denial of service defense, Packet capabilities, Dynamic packet stamping, Traffic policing.

1. INTRODUCTION

With the increase in the network usage for business, leisure and time-critical activities, distributed denial-of-service (DDoS) attacks have become an increasing threat. Numerous research and commercial endeavors to design effective DDoS defenses have lead to the following insights: (1) A defense needs to be deployed at or near the victim, where the economic incentive lies. Further, a victim is in the best position to determine if a client's traffic is malicious or benign, and thus has the most accurate information about what to filter. (2) A victim-end defense must be lightweight to support fast packet processing during an attack; otherwise it may become a target of the attack itself. (3) Because a victim may be overwhelmed by a large-scale attack, mechanisms are needed to facilitate attack traffic filtering by upstream routers. This means that a victim must somehow communicate to the routers information needed to discriminate between benign and malicious traffic. The discrimination process must also be lightweight, minimizing router CPU and memory cost.

Recently proposed capability mechanisms, such as SIFF [8] and TVA [9] embody these desirable DDoS defense properties in the following manner. Routers on the path to the victim build tickets (capabilities) cooperatively by appending a hash of the source and destination address, and a router secret, to each packet that does not already carry a ticket. A destination decides to grant the access to a client based on some private policy, and returns tickets to chosen clients. Tickets are granted for a limited period of time (time-based) [8] or for a limited amount of traffic (traffic-based) [9] and carry expiration information. An accepted client appends the ticket to future packets, and routers verify tickets and provide high-priority handling to ticketed traffic. Ticket verification is lightweight since a router only needs to recalculate the hash and verify that it is equal to the router's portion of the ticket contained in the packet. Thus routers pay moderate CPU cost. Memory cost is only paid in case of traffic-based tickets to keep statistics of ticket usage. Time-based tickets incur no memory cost.

While current capability mechanisms show great promise with regard to defense effectiveness and a reasonable operational cost, they suffer from the following deficiencies that we address in this paper:

1. ***Lack of mechanisms for automated ticket granting***: Neither SIFF [8] nor TVA [9] address the question of mechanisms for distinguishing between legitimate and malicious clients. This is a challenging task in case of public servers, where all clients are equal and no prior trust exists between a given client and the server. The only possible approach in this case is to grant short-term access to each new client and evaluate its behavior during this time. Well-behaved clients earn right to future tickets, while ill-behaved clients are shunned. We propose one possible approach to record a long-term client's behavior and incorporate this knowledge into the ticket-granting process. We associate degrees of trust with the clients by assigning a *credit* and a *penalty* to each client based on its long-term behavior. Credit is used to identify aggressive attackers; during congestion, the credit of an ac-

LSAD'07, August 27, 2007, Kyoto, Japan.

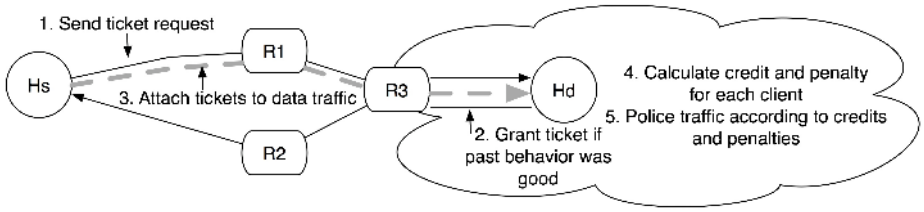

Figure 1: Components of the proposed defense

tive client is decreased proportionally to the amount of traffic it contributes to the congestion. However, credit assignment alone cannot deal with distributed attacks where each malicious client sends traffic at a very low rate. To handle such attacks, we assume that a legitimate client's response to packet drops will be more prominent than the malicious client's, and we assign penalties to clients that do not respond appropriately to packet drops. Jointly, the client's credit and the penalty are used for its traffic policing and to decide whether future tickets should be granted.

2. ***Binary capabilities***: Possession of a ticket grants full access to the victim while the ticket is valid, thus all admitted clients have equal priority. This enables sophisticated attacks where malicious clients first obtain tickets and then launch attacks. If attackers send traffic at a low rate, they may even be granted future tickets, perpetuating the attack. We propose fine-grained capabilities that carry a priority label, dependent on a client's long-term behavior. This enables us to penalize clients for any suspicious behavior, and provide guaranteed high-quality service to consistently well-behaved clients in case of sophisticated attacks.

3. ***Route-dependent capabilities***: Because routers on the path participate in ticket generation, tickets are route-dependent and will lead to legitimate traffic drops in case of a route change or multipath routing, both of which are frequent in today's Internet. Our ticket-generation mechanism involves only the traffic destination, making tickets route-independent. Upstream routers remain inactive unless explicitly authorized by the attack victim to aid in traffic filtering. This further reduces defense operational cost compared to [8, 9].

2. RELATED WORK

IP Easy-pass [6] attaches a source identifier to each packet and uses it to reliably identify clients. Some existing resource reservation protocol (e.g., RSVP) is assumed for access control. In the past, work has been done on identifying flows by assigning a unique handle [2], and on reliably and accurately identifying the traffic source [5]. In this paper, we address the issue of distinguishing a well-behaved client from an ill-behaved client during the ticket granting process, thus our work is orthogonal to work on client identification.

Anderson et. al. [1] propose capabilities (tickets) attached to each client packet, that guarantee privileged access to a resource. They assume a separate overlay for transmitting ticket requests, which incurs high setup cost. SIFF [8] refines the capability approach by eliminating the need for a separate overlay channel. Instead, routers build capabilities collaboratively using a secret key to hash some packet fields and placing the output in ticket request packets. Destination grants access to clients based on some internal policy and returns the capability from request packets to these clients that attach it as a ticket to future packets. The tickets are time-based. TVA [9] improves the design from SIFF [8] by using traffic-based tickets and by rate limiting and prioritizing ticket-request traffic. As discussed in Section 1, SIFF and TVA have certain limitations that we aim to improve.

3. CAPABILITY MECHANISM

Figure 1 illustrates steps in a source's access to a destination. Communication between a source and a destination is preceded with a ticket request. If the source communicated with the destination in recent past, the ticket request will carry the context of the old communication including the old credit and penalty values. The client's credit and penalty serve as inputs to the ticket-granting process, and tickets are returned to accepted clients. Unlike the past work on capabilities [1, 8, 9], a possession of a ticket does not translate into an absolute privilege to access the destination. Instead we associate a degree of trust with each client, expressed via its credit and penalty values and attached to its current ticket. We use this trust information to prioritize access to a critical resource (Section 3.3), thus favoring well-behaved clients over unknown clients, and favoring unknown over known-malicious clients.

3.1 Ticket Structure

A destination generates a *client-ticket* for each client to whom it wishes to grant access, and the client uses this information to generate a *packet-ticket* attached to each future packet sent to this destination. Our generation of client-tickets and packet-tickets has the following properties:

- *Client-tickets are bound to the client*: To prevent ticket falsification and stealing, the destination generates the client-ticket by hashing the client's credit, penalty and IP address with the destination's secret. Client-ticket structure is shown in Figure 2. Including the client's IP in the hash binds the ticket to the specific client, thus ensuring that attackers cannot use stolen tickets to buy a passage for their traffic. A similar mechanism exists in TVA [9]. However, the attacker could use a stolen ticket to generate spoofed traffic with client's IP address as an alleged source. To prevent this we must prevent ticket stealing from a destination's reply to the ticket request, and later from packets with valid tickets.

  To prevent ticket stealing from a destination's reply we

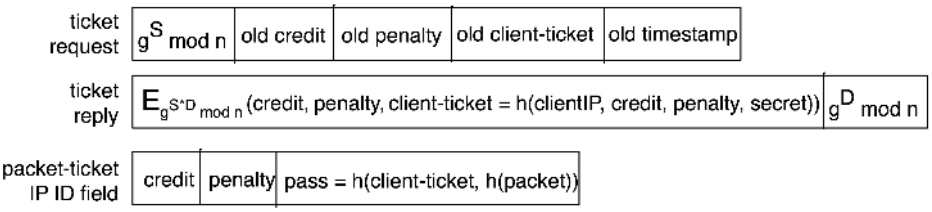

Figure 2: Structure of the ticket-request, ticket-reply, client-ticket and packet-ticket

deploy the Diffie-Hellman key exchange [3] to generate a session secret between the source and the destination, and use this secret to encrypt ticket information in the reply. We assume a general knowledge of numbers g and n. In its ticket requests, the source includes $g^S \mod n$, where S is a random number selected by the source. In the ticket reply, the destination returns $r = E_K(\text{ticket})|g^D \mod n$, where E_K denotes encryption with key K, using some lightweight symmetric encryption protocol, D is a random number selected by the destination, $K = g^{S \cdot D} \mod n$ is the shared secret and | denotes concatenation. The shared secret can be calculated by the source and the destination only, because they possess one part of this secret (the random number S or D). Both parties store the secret and use it for future ticket exchanges. Old secrets can be removed after a period of inactivity. The structure of the ticket reply is also shown in Figure 2. TVA [9] does not encrypt ticket information in replies and is thus sensitive to ticket stealing.

Our current design uses IP address of a host as an identifier, which does not address the presence of NATs in the network, or the dynamic addressing. We plan to investigate these issues in our future work.

- *Packet-tickets are bound to packets*: To prevent ticket stealing from packets, a client generates packet-tickets by binding the client-ticket to each packet. This is done by first calculating the hash of the packet's contents and immutable header fields, and then hashing this result with the client-ticket to produce a per-packet pass. This pass, along with the client's credit and penalty values represents the packet-ticket and is inserted into the IP identification field, as shown in Figure 2. SIFF [8] and TVA [9] do not bind tickets to packets, enabling misuse of stolen tickets to spoof legitimate client's traffic.
- *Client-tickets are short-lived*: If tickets were valid for a long time, a mutable attacker that behaves well to obtain a ticket, and then turns hostile could inflict much harm. Short ticket life limits the damage from a mutable attack, and is also employed in SIFF [8], while TVA [9] employs costly accounting to limit the amount of traffic sent using a single ticket, i.e. it uses traffic-based capabilities. We opted for time-based vs. traffic-based capabilities, to reduce the operational cost. Tickets expire periodically when the destination changes the secret used for ticket generation. We call this interval the *ticket-validity interval*. Delayed packets are handled by accepting the packets with tickets valid during one previous interval.
- *Ticket verification is lightweight*: All the information needed to verify validity of a ticket (packet or client) is encoded in the ticket, thus no memory is needed to store client information at the destination. The destination does pay a small memory cost to record the blacklist of worst offenders, and to keep behavior statistics for currently active clients (Section 3.2).

3.2 Calculating Credits and Penalties

Credits and penalties are used to reflect a client's behavior by describing the aggressiveness of its sending pattern. Credit and penalty calculations are performed at the end of each ticket-validity interval.

3.2.1 Credit Calculation

A client's credit reflects its contribution to congestion during a flooding attack – the higher credit represents the lower contribution, i.e. well-behaved clients will have high credits. The credit is a number ranging from LOW to HIGH. A new client is assigned a credit value of MID, which lies in the middle of [LOW, HIGH] range. If no resource overload is observed during an interval, then an active client c in that interval is rewarded by an additive increase in its credit:

$$credit_c^{new} = \min(credit_c^{old} + \alpha, HIGH) \quad (1)$$

where α is the credit increase factor. During resource overload periods, if a client is identified as non-aggressive, its credit is also calculated using the Eq. 1. The credit of an aggressive client c is decreased multiplicatively and proportionally to its contribution to resource demand:

$$credit_c^{new} = \max(credit_c^{old} \cdot (1 - \frac{E_c}{\sum_i T_i}), LOW) \quad (2)$$

where T_i is total traffic sent by the client i in units that represent a critical resource (e.g., bytes for bandwidth, service requests for server-specific resource, packets for CPU), and E_c is the excess traffic the client c sent above its fair share, which we call a quota, and denote with Q_c. The sum of T_i is calculated over all active clients. Multiplicative decrease ensures prompt action to the observed aggressiveness.

Values of T_i, E_i and Q_i are calculated for a window of several intervals to avoid overreaction to variations in client traffic. The quota of a client c is calculated as:

$$Q_c = \frac{\max(credit_c - penalty_c, LOW)}{\sum_i \max(credit_i - penalty_i, LOW)} \cdot R, \quad (3)$$

where R is the amount of the critical resource (e.g., bandwidth, number of packets or service requests that can be processed per second, etc.), the maximum is calculated over the window for a given client, and the sum is calculated over all active clients. A client c is considered aggressive if

it exceeds its quota in an interval, and its credit is decreased using the Eq. 2, where $E_c = T_c - Q_c$.

The defense proactively renews tickets of all active clients at the end of each ticket-validity interval. A client that has been inactive during an interval must issue a new ticket request. To enable well-behaved clients to benefit from their past good reputation, a new ticket request carries the last received credit and penalty values, along with the client-ticket and the timestamp of the last activity. Using the timestamp, the server locates the secret, which was valid at the given time, and uses it to verify the client-ticket and thus the authenticity of the declared credit and penalty values. Upon success, it uses the past credit value to calculate starting credit for the client c as:

$$credit_c^{new} = \max(credit_c^{old} - \beta \cdot N, MID), \quad (4)$$

where β is the credit decrease factor, and N is the number of intervals since the client's last communication. We deploy credit aging to discount stale information because a longer inactivity period increases the possibility of a client's compromise. The lowest credit assigned to an old client is MID, ensuring that a very old client is treated the same as a previously unknown client. The penalty value of an old client is conservatively set to the past penalty value, declared in its ticket request. A previously unknown client receives the lowest penalty value.

3.2.2 Penalty Calculation

Consider a scenario when many attackers flood a network, but each attacker sends traffic at a low rate. In such scenario, a legitimate client's contribution to congestion is larger than that of an attacker, so credit calculation alone cannot help us precisely identify malicious clients.

To rectify this situation, we use an observation that a legitimate client will reduce its sending rate upon a traffic loss, while an automated attacker will not. One source of rate reduction is the TCP's congestion control mechanism that responds to traffic loss by an exponential decrease in the sending rate. If a malicious client uses modified version of the TCP protocol to send aggressively, its response to congestion will be milder than that of legitimate clients. Even if a malicious client uses unmodified TCP, it will open multiple connections to the destination to send sufficient traffic for service denial making it more aggressive than an average legitimate client.

We postulate that another source of rate reduction could be human response to low service quality – a person that does not receive a response to their service request is unlikely to maintain or increase the rate of request generation. Further study with human subjects is needed to verify this hypothesis and is part of our future work.

We assign *penalties* to clients that experience persistent packet drops in the following manner. Let D_c be the sum of dropped bytes from client c during the window. If $D_c > \delta \cdot T_c$, the client is considered malicious and its penalty is increased as:

$$penalty_c^{new} = \min(penalty_c^{old} + \gamma, HIGH), \quad (5)$$

where δ is the estimate of the legitimate client's aggressiveness in face of persistent drops, and γ is the penalty increase factor. If the client is not identified as malicious its penalty is decreased as:

$$penalty_c^{new} = \max(penalty_c^{old} - \gamma, LOW). \quad (6)$$

3.2.3 Aggressive Client Blacklisting

To reduce the memory cost of the defense, client credits and penalties are carried in tickets. This opens a potential vulnerability since a client with a low credit (or a high penalty) would benefit from posing as a new client, i.e. it would omit the credit and penalty information from its ticket requests. Legitimate clients would then have to contend for bandwidth with attackers, just as is the case in SIFF [8] and TVA [9]. To amend this situation the defense should keep a blacklist of worst offenders. Clients with lowest credits or highest penalties would be stored in this list, with their credit and penalty information, and each new ticket request would be checked against this list.

3.3 Traffic Policing

Previous work on capabilities [1, 8, 9] allowed absolute access to the destination to all ticket-carrying traffic. As discussed in Section 1, this approach can inflict large harm to legitimate clients in case of mutable attackers. To minimize this harm, we use client credits and penalties to prioritize access to the critical resource. Each client c is assigned to the client class identified as:

$$clientClass_c = \max(credit_c - penalty_c, LOW), \quad (7)$$

and each class is assigned a certain share of the resource. A client can access the resource share assigned to its class and that assigned to lower classes. If all such resources are depleted, the client's request is dropped. This facilitates good service to well-behaved clients during an attack that deploys many previously unknown attackers. These attackers fall into the same credit class as previously unknown legitimate clients, and compete with them for the resource, but cannot deplete resources assigned to the higher-credit classes that contain known, well-behaved clients.

One approach to resource assignment would be to uniformly distribute the critical resource among all credit classes. However, this design could lead to under-utilization, in cases when most users lie in low or middle credit ranges. We propose a more sophisticated scheme that estimates future resource requirements of a client class *cc* based on the weighted average of its past demand as follows:

$$R_{cc}^{new} = (1 - \lambda) \cdot R_{cc}^{old} + \lambda \cdot \text{demand}, \quad (8)$$

where R_{cc}^{old} was the estimate at the end of the previous interval, demand is the total resource usage of this client class in the current interval and λ is the weight assigned to new observations.

Traffic policing is performed by the defense located at or near the victim. If the defense is overwhelmed, which is likely during high-rate attacks, it can request help from upstream routers for packet filtering. The help request contains at the minimum the previous and the current destination secret, to enable the router to validate ticket information. Future secrets could also be included in the help request, or they could be communicated periodically through future help requests. It would further be helpful to include the blacklist of recent offenders in the help request, to enable the router to filter new ticket requests from known-malicious clients.

Help requests must be authenticated to prevent denial of service through fake help requests that contain invalid secrets and are sent by a third party. Authentication assumes an existence of a trust relationship between the de-

fense and an upstream router. Since distributed trust is difficult to enforce unless there is an existing business relationship, we envision that help requests would only be propagated one hop upstream, to routers of the victim's ISP. This is a common business practice today, but requests are delivered through human channels, which impose large delays, and they contain imprecise filtering information obtained from intrusion detection systems. The proposed capability mechanism would automate this process and improve filtering accuracy and the response time.

Parameter	Value
Ticket-validity interval	3 s
Window size	4 intervals
α	1
β	0.3
γ	0.4
δ	1
$HIGH - LOW$	20

Table 1: Parameter values

3.4 Parameter Settings

We use several parameters to guide the defense operation, whose values are shown in Table 1. We now briefly discuss tradeoffs in setting their values. In real deployment optimal parameter values will greatly depend on legitimate traffic dynamics in a given network, and should be determined through traffic analysis, training and tuning over several days or weeks.

- Credit range [LOW, HIGH]: A larger range provides a finer granularity for client differentiation and thus better defense, but will cause additional computational and memory cost during traffic policing.
- Credit and penalty change factors (α, γ): Large values of α and γ make credits and penalties very sensitive to traffic variations, which can lead to penalizing normal variations in legitimate traffic. Too small values on the other hand, prolong response time of the defense.
- Estimate of legitimate client's aggressiveness (δ): A large value of δ will increase penalty only for large drop rates, allowing moderately aggressive attackers to evade the defense. A small δ value penalizes small, normal traffic variations of legitimate clients.
- Score aging factor (β): A small value of β preserves history of past good behavior for a long time, while a large value rapidly discounts recent good behavior.

4. COST

We now summarize the cost of the proposed defense. While issuing and updating tickets, the defense performs Diffie-Hellman key exchange once for every new or recently inactive client, followed by ticket encryption once each ticket-validity interval. While Diffie-Hellman key exchange is costly, it is only performed for clients that have not been active recently. The cost of the exchange can thus be controlled by increasing the memory for storage of shared secrets. Symmetric encryption and decryption are moderately costly, but the frequency of these operations is low — once each several seconds. An attacker could attempt to exhaust the defense's resources by sending a lot of new ticket requests and we discuss this case in the Section 5.

Tickets are kept small and ticket validation is not costly. A sender attaches the packet-ticket to each packet and the defense verifies it. Both require two hash operations per packet and can be done at high speed, as shown in [9]. Note that there should be a significant reduction in deployment cost between our defense and SIFF [8] or TVA [9] because our defense is located at the destination only and the help of upstream routers can be invoked on need basis, while SIFF and TVA require constant support from upstream routers.

Tickets carry the client information needed for ticket validation and traffic policing, requiring no additional storage at the defense. Defense incurs a storage cost for storing traffic statistics and the quota of each active client during an interval, for computing credits and penalties. In case of a large number of clients, it is sufficient to store statistics only of aggressive senders that dominate the values in score and penalty computation. Statistics are also stored for each client class, thus the size of the [LOW, HIGH] range determines the cost of this storage. The defense also incurs a small memory cost for a blacklist of worst offenders. It may pay off to propagate this list to some upstream routers that are close to destination, when their help is requested, in which case the upstream routers will incur the memory cost to store this information. The typical size of botnets today is at most 100,000 hosts [4], making the memory cost for storing a blacklist 3.2 MB.

5. SECURITY

We now briefly discuss the security of the proposed defense. As our experiments illustrate in the next section, the defense can successfully identify large and persistent senders, but its performance degrades in case of pulsing attacks. If an attacker used a large number of zombies in smaller groups, such that a single group acts maliciously at a given time and is then replaced by a fresh group, the attack could continuously deny service. All DDoS defenses to date that use a client's identity for traffic prioritization will be ineffective against such attack.

Another possible attack would engage zombies that do respond to congestion, thus avoiding high penalty values. We believe that in this case human behavior (rate of request generation) would differ from the behavior of zombies causing legitimate client's traffic to decrease below malicious client's traffic. We plan to study this in our future work.

Tickets cannot be falsified because secret hash facilitates integrity checks. Our defense is resistant to sniffing due to deployment of cryptographic techniques to protect tickets. It is also resistant to IP spoofing because it encrypts client-tickets and binds packet-ticket values to the packets.

Cryptographic operations make defense vulnerable to flood of bogus ticket requests, that initiate costly Diffie-Hellman key exchange. One way to address this problem is to limit the resources spent for ticket-granting. This ensures that well-behaved and active clients will receive good service, since their secret information is cached. New legitimate clients will have to contend for the access to ticket-granting mechanism along with attackers.

6. EVALUATION

We implemented the proposed capability mechanism in a

Linux software router as a loadable kernel module. Our tests consist of live-traffic experiments in the Emulab testbed [7]. We used the topology shown in Figure 3. Victim node V is connected to the rest of the topology via a bottleneck link of 100 Kbps, which represents our critical resource. All other links in the topology have 100 Mbps bandwidth. There are two legitimate clients L1 and L2 and seven attackers A1–A7.

Legitimate traffic is generated by invoking a character generator program at the client nodes, and tunneling its output to the victim node via SSH. The character generator emulates Telnet traffic – it generates one message per second, whose length is randomly chosen in a predetermined range. A message can be split into several packets. We call the average rate of the character generator the *legitimate client's nominal rate*. Depending on the TCP's congestion control mechanism, legitimate client's traffic will flow into the network at, above or below the nominal rate. As explained in Section 5, to use a real TCP traffic for attack, the attackers would need a large number of zombies due to the congestion responsive nature of TCP. Hence, attack traffic is generated using raw sockets to send TCP packets at a specified rate. The attack rate may vary in some test scenarios in an attempt to trick the defense.

We do not show a simple scenario where the attack traffic does not carry a ticket – all such traffic will be correctly dropped since only ticket-carrying traffic is allowed to reach the victim. We also omit a scenario where a mutable attacker acquires a ticket and then increases its sending rate to a large value. Such attacker will be quickly identified as aggressive and its credit is decreased, providing effective defense. We focus instead on sophisticated attacks involving mutable attackers that send at a relatively low rate to maintain impression of a good behavior and ensure receipt of future tickets.

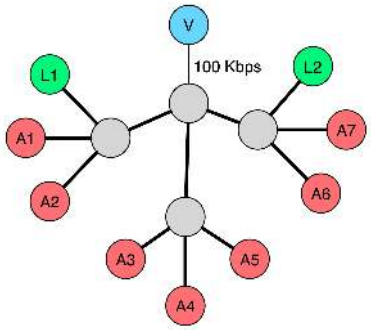

Figure 3: Network topology used for evaluation

6.1 Balanced Attack

To blend in with legitimate clients, each attacker first acquires the highest credit by sending traffic at a low rate (800 bps) for a long time – this behavior does not create resource overload. Afterwards, attackers turn malicious and send at the legitimate client's nominal rate (24 Kbps). Figure 4 shows the credits of one legitimate client and of one attacker; credits of other clients follow the same trend. Before the attack, credits of legitimate and attack clients are at the HIGH value. Soon after the attack starts, an attacker's credit is decreased, thanks to our aggressive sender identification and the multiplicative credit decrease. The credit of legitimate TCP client decreases briefly after the attack's onset, because the traffic computations are performed over statistics collected in a sliding window. Once the TCP's congestion control reduces the sending rate, several intervals are needed for this to sufficiently impact the average rate value in the window. Similarly, Figure 5 shows a legitimate client's and an attacker's penalty. While a legitimate client's penalty remains low throughout the attack, an attacker's penalty quickly reaches the maximum value due to the absence of congestion response in attack traffic.

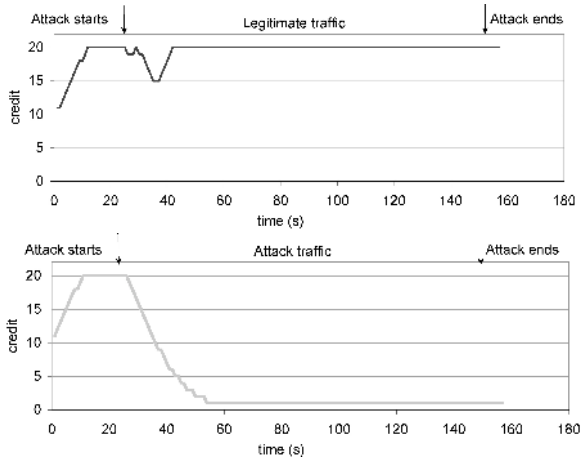

Figure 4: Credits of legitimate and attack clients

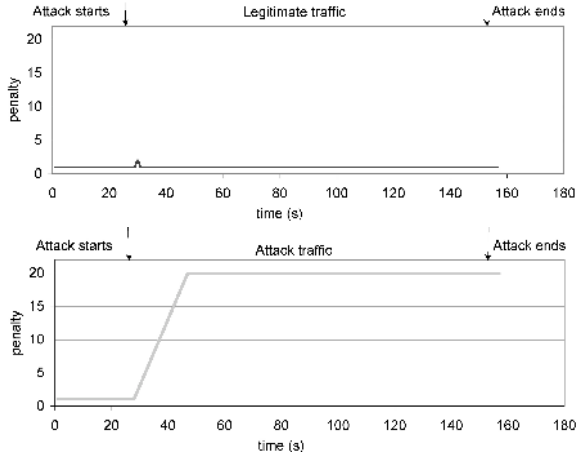

Figure 5: Penalties of legitimate and attack clients

Figure 6 shows the *acceptance ratio* – the percentage of bytes sent by a client that successfully reach the victim. Note that this is different than bandwidth allocation between clients. An acceptance ratio of 100% means that no traffic from this client was dropped, either due to congestion or by defense. The acceptance ratio gives no information about the bandwidth division between the legitimate and the attack traffic.

A legitimate client's acceptance ratio is temporarily lowered when the attack starts, but quickly converges to 100%, while an attacker's acceptance ratio is reduced to around 5%. For comparison, Figure 7 shows the acceptance ratio

without the defense – all traffic drops occur due to the congestion. A legitimate client's acceptance ratio fluctuates, and frequently reaches zero, as the legitimate traffic's sending rate fluctuates due to TCP's congestion control. The attacker's acceptance ratio is around 40% because the bottleneck link bandwidth is 40% of the total traffic arriving at the link. The legitimate traffic is seriously damaged during the attack without the defense, while it is efficiently protected when the defense is present. For space reasons we will only show the acceptance ratio for the following tests.

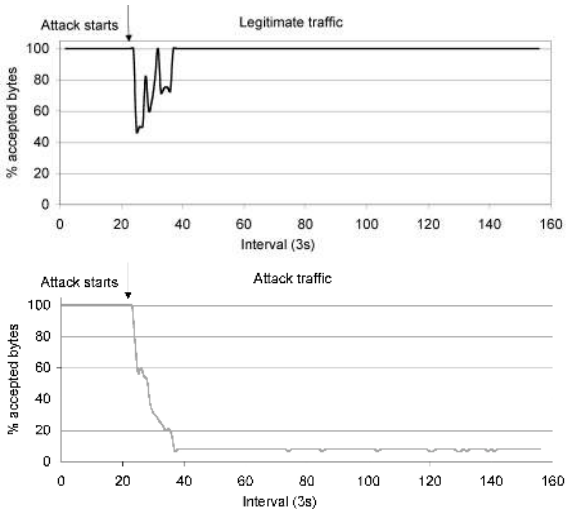

Figure 6: Acceptance ratio during the balanced attack

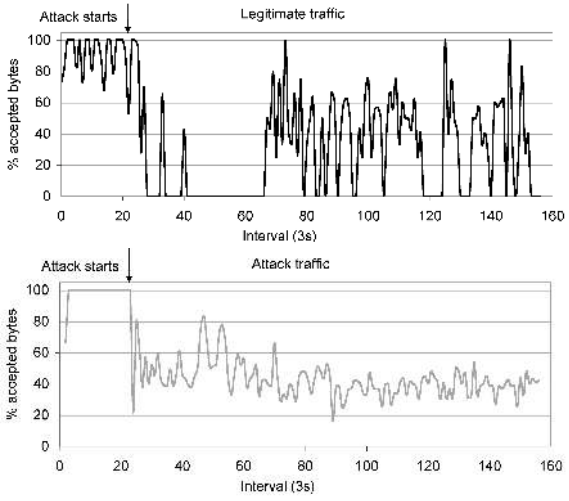

Figure 7: Acceptance ratio during the balanced attack without defense

6.2 Low-rate Attack

In this test each attacker sends at 80% of the legitimate client's nominal rate (19.2 Kbps), thus attempting to avoid being identified as an aggressive sender. Our results, shown in Figure 8, demonstrate that even when a large number of attackers send at a low individual rate to create a denial of service, our defense identifies these attackers via their increased penalties, since their traffic does not exhibit congestion response. The acceptance ratio graph resembles the one in the balanced attack case. After the first 20 intervals, all legitimate traffic reaches the victim. An attacker's acceptance ratio is quickly reduced to 10%. A lower malicious client rate leads to penalties that take longer time to increase, thus the attack interferes with the legitimate traffic longer. An even lower-rate, more distributed attack would inflict damage to legitimate traffic for a longer period of time, but the defense will eventually converge and protect legitimate traffic.

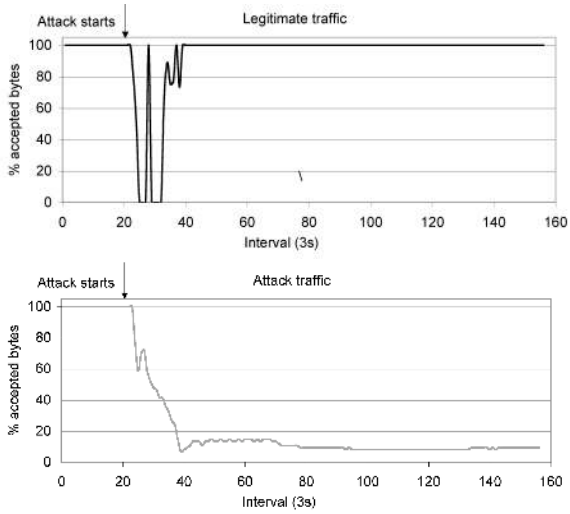

Figure 8: Acceptance ratio during the low-rate attack

6.3 Pulsing Attack

We next test the pulsing attack in which the attacker periodically sends heavy traffic (legitimate client's nominal rate=24 Kbps), and then sends low traffic (800 bps) to build up the trust until the next pulse. The acceptance ratio is shown in Figure 9. While the attackers' credits increase during low-rate periods, the defense quickly identifies attackers as aggressive during high-rate periods and suppresses their traffic. For the legitimate client, the acceptance ratio drops at the onset of high-rate periods (labeled as "High" in the graph) but then returns to 100% where it remains for the rest of the period, and during low-rate periods. The attacker's acceptance ratio is high during low-rate periods because no overload is created. During high-rate periods the acceptance ratio quickly drops to about 5%, which is consistent with our results for the balanced attack.

6.4 Binary Capabilities

We motivated our design of capabilities with multiple degrees of trust by arguing that binary capabilities cannot protect legitimate traffic during mutable attacks. We now support this claim by repeating the balanced attack experiment with binary capabilities. We keep our calculation of credits and penalties the same, but the client's fair share of the resource is obtained by dividing the resource equally among all active clients, regardless of their credit or penalty. Traffic policing component accepts all traffic with the credit greater

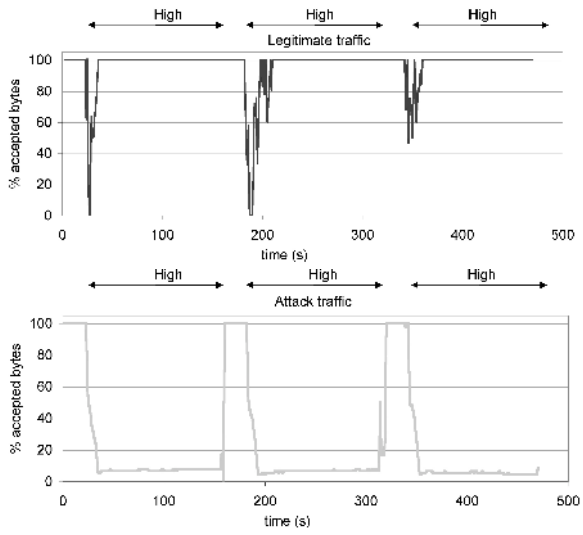

Figure 9: Acceptance ratio during the pulsing attack

or equal to MID/2. Figure 10 shows the acceptance ratio for this experiment. While the attacker's acceptance ratio eventually drops to zero, the legitimate client's traffic experiences significant drops and its acceptance ratio exhibits large variations, frequently reaching 0%. Comparing the Figures 6 and 10, the protection offered to legitimate traffic by binary capabilities is much worse than the protection offered by our proposed defense. In the absence of a sophisticated traffic policing, the legitimate client receives the same bandwidth share as the attacker, causing the client's credit to fluctuate between high and low credit values based on its traffic variations in response to congestion. This leads to large variations in the legitimate client's acceptance ratio.

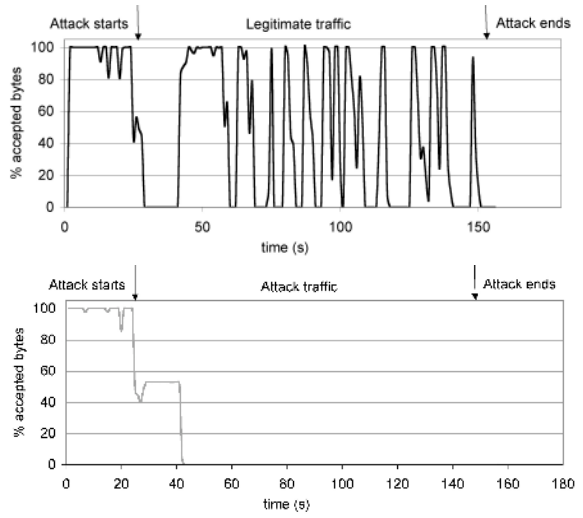

Figure 10: Acceptance ratio during the balanced attack with binary capabilities

Table 2 summarizes experiment results showing the percentage of legitimate traffic throughput during an attack compared to the throughput without an attack. The results show that our defense provides excellent protection to the legitimate traffic, whose throughput is very close to 100%.

Experiment	Throughput (%)
Balanced attack w defense	98.06
Balanced attack w/o defense	2.91
Low-rate attack w defense	98.69
Pulsing attack w defense	99.98
Balanced attack w binary cap.	85.96

Table 2: Legitimate traffic throughput during attack

7. CONCLUSIONS

We proposed several improvements to the original capability design that facilitate automatic ticket-granting and improve security and cost of the defense. Our experiments show that the proposed defense successfully handles sophisticated attacks, offering a consistent good protection to legitimate traffic and quickly identifying and penalizing attack traffic. In our future work we plan to investigate human response to low service quality, and improve our penalty calculation with models derived from this research. We also plan to explore a dynamic setting of parameter values based on the perceived attack severity, and to engage in larger-scale experimentation to validate our proposed defense. Finally, we plan to address remaining security issues related to use of cryptography during ticket issue.

8. REFERENCES

[1] T. Anderson, T. Roscoe, and D. Wetherall. Preventing Internet Denial of Service with Capabilities. In *Proc. of HotNets-II*, 2004.

[2] M. Casado, A. Akella, P. Cao, N. Provos, and S. Shenker. Cookies Along Trust-boundaries (CAT): Accurate and Deployable Flood Protection. In *Proc. of 2nd Conference on Steps To Reducing Unwanted Traffic on the Internet*, 2006.

[3] W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[4] Honeynet Project and Research Alliance. Know your enemy: Tracking botnets. `http://www.honeynet.org/papers/bots/`.

[5] D.R. Simon, S. Agarwal, and D. A. Maltz. AS-Based Accountability as a Cost-Effective DDoS Defense. In *Winter International Symposium on Information and Communication Technologies*, 2004.

[6] H. Wang, A. Bose, M.A. El-Gendy, and K. G. Shin. IP Easy-pass: A Light-Weight Network-Edge Resource Access Control. *IEEE/ACM Transactions on Networking*, 13(6):1247–1260, 2005.

[7] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. In *Proc. of OSDI*, pages 255–270, December 2002.

[8] A. Yaar, A. Perrig, and D. X. Song. SIFF: A Stateless Internet Flow Filter to Mitigate DDoS Flooding Attacks. In *Proc. of IEEE Symposium on Security and Privacy*, 2004.

[9] X. Yang, D. Wetherall, and T. Anderson. A DoS-limiting network architecture. In *Proc. of ACM SIGCOMM*, pages 241–252, 2005.