Flow-Cookies: Using Bandwidth Amplification to Defend Against DDoS Flooding Attacks

Martin Casado, Pei Cao
Stanford University
{casado,cao}@cs.stanford.edu

Aditya Akella
University of Wisconsin, Madison
akella@cs.wisc.edu

Niels Provos
Google, Inc.
niels@google.com

I. Introduction

Flooding attacks, where an attacker attempts to exhaust the downstream bandwidth of a server, are particularly difficult to defend against. Unlike other forms of DDoS such as SYN-flooding, computation attacks or request floods, the downstream bandwidth is not under a web-server's control. And therefore, while there exist server side protection mechanisms to protect server resources, such as syn-cookies [1] and secure admission control schemes, few practical solutions to defend against flooding exist today.

Part of the problem is the difficulty in pushing filtering requests into the network where there is sufficient bandwidth to handle the flood. An in-network element cannot distinguish a single packet as being part of a legitimate flow without doing flow tracking, a tricky proposition [5] that traditionally requires per-flow state which may be untenable for a high-bandwidth link. Other solutions are easily fooled by source spoofing [2] or require massive architectural changes [4].

We present "flow-cookies", a mechanism in which a website can reliably send filtering requests to a cooperating node in the network, leveraging its protection bandwidth. In this approach, a third party provider installs a flow-cookies enabled middlebox (Figure 1), called the cookie box, in the network at a high bandwidth link. All traffic to or from the protected webserver must traverse the cookie box. The cookie box guarantees that all packets that pass between it and the server belong to a legitimate TCP flow with a valid sender. Further if a web-server deems a client to be misbehaving, it can request the cookie box to filter the offending IP.

Flow-cookies, is a simple extension to SYN cookies[1], wherein the protection middlebox places a secure, limited lifetime cookie within the TCP timestamp of every outgoing data packet from the protected server. Flow-cookies is lightweight, low-state, does not require modification to clients, and is resistant to source spoofing. Further flow-cookies does not have the first-packet flooding problem [4] and does not require massive infrastructural changes to be effective.

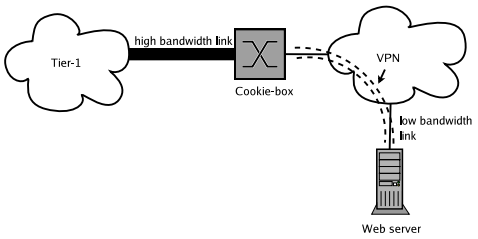

Fig. 1. With flow-cookies, a webserver cooperates with a "cookie box" connected to a high bandwidth link. The cookie box provides high-bandwidth protection for the web-server by filtering all blacklisted IPs and only allowing packets that are part of legitimate flows to pass.

II. Mechanism Overview

Flow-cookies operates using a stand-alone device—which we call the flow-cookie box, or cookie box for short—deployed by a third-party service provider. We assume that the cookie box is deployed in a data center with a very high speed connection (e.g. a tier-1 ISP). All traffic to and from the protected web-server must traverse the cookie box. The cookie box maintains a private channel to each back-end web-server it is protecting.

Flow-cookies is a defense for TCP services. Therefore, we assume that incoming packets for other protocols, such as UDP or ICMP, are placed in a separate queue and cannot affect TCP flows.

The cookie box and protected web-servers cooperate to perform the following four tasks:

1) All incoming clients complete the TCP 3-way handshake with the cookie box. The box uses SYN cookies for the handshake. SYN cookies does not require state maintenance at the cookie box and can be run at gigabit line speeds [3]. This step ensures that SYN floods cannot traverse the link between the cookie box and the protected website.

2) Once a client has completed the handshake, the cookie box hands off the connection request to the website using a TCP-handoff scheme.
3) For outgoing packets from the web-server to its clients (this happens after the web-server has accepted the client connection request), the cookie box adds a secure flow-cookie. The cookie is computed using a keyed message authentication code over a counter and the connection 4-tuple: **cookie = MAC(**$S_r, C_r|src_{ip}|src_{port}|dst_{ip}|dst_{port}$**)** Where S_r is a secret known only to the cookie box and C_r is a counter that increments periodically to time-out the cookie.
4) The flow-cookie is echoed back by the client, and checked by the cookie box. This is to ensure that only packets belonging to flows accepted by the server traverse the link between the cookie box and the web server.
5) The webserver understands the accepted usage policy of its local administration, and is already keeping per-flow state for each outstanding connection. Therefore, the backend server is in the best position to determine if a client is misbehaving. IPs (and associated ports) deemed malicious by the webserver (e.g. deviant flows) are passed to the cookie box, which filters current and future packets from the offending clients.

Figure 2 contains psuedo-code of the flow-cookies algorithm as performed by the cookie box.

To be backward compatible, we exploit the *TCP timestamp* option, and place the *flow-cookie* from step #3 in the timestamp field of packets (which is echoed back by the client). Flow-cookies are valid for a limited period and are non-forgeable. The cookie box verifies that all packets contain a legitimate cookie thus ensuring that only packets from clients accepted by the server are forwarded. All others are dropped. Therefore the cookie box provides protection proportional to its line-speed for one or more backend servers which, themselves, have low speed connections.

```
Sr: secret
Cr: current counter value
Ts: last server timestamp
FLOW_START: known constant

1 if packet is SYN:
2   if scrIP in IPBlacklist:
3      drop packet and exit
4   syn_cookie = HMAC(Sr, Cr | 4-tuple)
5   send back SYN_ACK with ISN = syn_cookie and
    timestamp = FLOW_START
6 else:
7   cookie = HMAC(Sr, Cr | 4-tuple)
8   if timestamp == FLOW_START:
9      if ACK sequence != cookie:
10        drop packet and exit
11  else:
12     if timestmp != cookie:
13       drop packet and exit
14     else:
15       if <srcIP and srcPort> in FlowBlacklist:
16          drop packet and exit
18  forward the packet
```

Fig. 2. Per-packet logic at the cookie box for packets destined to the protected web-server.

III. Properties

Requiring the cookie box to offload the TCP-handshake in the network serves a duel purpose. First, all flows that are passed back to the server must have completed the three way handshake and therefore are unlikely to be spoofed. This greatly increases their utility in making filtering decisions. Second, spoofed SYN floods, which are difficult to detect in the network, are handled at a high-bandwidth link raising the resources requirements of an attacker to launch an effective flood.

The use of flow-cookies relieves the cookie box of having to maintain state during normal operations (however, the box may store IP filters upon a webserver's request). The flow-cookies allow the cookie box to determine if packet belongs to a legitimate flow. That is, whether the packet was sent by a host that completed the three-way handshake (required to get the cookie) and whether or not flow has been revoked.

The use of the TCP timestamps to carry to cookie allows flow-cookies to operate without modification to clients. To demonstrate this, we have implemented flow-cookies in software and tested our implementation using live connections between various commodity client operating systems (WindowsXP, MacOSX, NetBSD, Linux2.4, Linux2.6) and multiple popular, public web sites. Our implementation is able to operate at gigabit speeds including per-packet IP filtering of millions of addresses. We also found our approach to be very effective against high volume SYN flooding attacks. Full details of our implementation can be found in [6].

References

[1] D. Bernstein. Syn cookies. http://cr.yp.to/syncookies.html, 1996.
[2] J. Ioannidis and S. M. Bellovin. Implementing pushback: Router-based defense against DDoS. In *Proceedings of Network and Distributed System Security Symposium*, 2002.
[3] Netscaler syn flood protection, http://www.netscaler.com/docs/library/NetScalerSYNDefense.pdf.
[4] K. Argyraki and D. Cheriton. Network capabilities: The good, the bad and the ugly. In *ACM HotNets IV*, 2005.
[5] M. Handley, C. Kreibich and V. Paxson, Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics In *USENIX Security Symposium*, 2001.
[6] M. Casado, P. Cao, A. Akella and N. Provos, Flow-Cookies: Using Bandwidth Amplification to Defend Against DDoS Flooding Attacks http://yuba.stanford.edu/∼casado/flow_cookies.pdf, 2006.