Preliminary Study of Fission Defenses against Low-Volume DoS Attacks on Proxied Multiserver Systems

Yuquan Shan, George Kesidis Daniel Fleck, Angelos Stavrou
EECS, PSU, Univ. Park, PA CS Dept, GMU, Fairfax, VA
{gik2,yxs182}@psu.edu {dfleck,astavrou}@gmu.edu

CSE Dept, PSU, Technical Report No. CSE-17-001
Feb. 22, 2017

Abstract

Multiserver applications deployed in the public cloud infrastructure continue to be plagued by significant threat of Distributed Denial of Service (DDoS) attacks by large scale botnets, including very notable attack instances just this past Fall. We describe defenses to address different aspects of this problem including: a proactive moving target approach to combat the reconnaissance phase where the botnet ascertains the identities (IP addresses) of the proxy (indirection) servers, and reactive client-to-server “shuffling” and “fission” quarantine approaches to deal with high or low volume DDoS attacks respectively targeting the proxy or replica application servers. In this paper, we overview our preliminary implementation of these defenses and describe the results of a model based evaluation of their performance.

1 Introduction

Consider a generic multiserver virtual system of clients, indirection/proxy servers, and worker/application servers (replicas), where the servers are controlled either by a central coordination server or in a distributed fashion by the proxies themselves [7, 8, 20, 21], see Figure 1. The clients reach the proxies via the public commodity Internet. The proxies and replicas could be implemented in Virtual Machines (VMs) or lightweight containers (instances) on physical servers of one or more data-centers. The proxies in turn assign clients to replicas to fulfill their requests. The clients never know the IP addresses of the replicas, only those of the proxies which they determine by DNS (the coordination server manages the DNS records of the proxies). That is, the proxies segment the sessions between

*Research supported by DARPA XD3 grant, gifts of Amazon AWS and Google GCE credits, and a Cisco Systems URP gift.
clients and replicas. For example, load-balanced content distribution networks or “derivative clouds” can be mounted on such systems.

Generally, the threat of Distributed Denial of Service (DDoS) attacks (e.g., [16, 13, 2]) continue to be a concern because the fundamental lack of economic incentives to secure attack-participating bots\(^1\) and/or deploy egress filtering in their access network. Indeed, two significant botnet based DDoS attacks were witnessed just this past fall [5, 11].

In Section 2 of this paper, for a generic proxied multi-server system, we overview proposed defenses to combat

- the reconnaissance phase of a botnet to identify the current IP addresses of the proxies,
- high-volume floods targeting the proxies (e.g., recently [5, 11]), and
- low-volume attacks targeting the replicas (e.g., [19] and more recently [10]).

We then focus on reactive fission defense to combat low-volume attacks targeting replicas. In Section 3, we describe our developing experimental set-up on

\(^1\)e.g., well known techniques of specification-based behavioral anomaly detection and prevention for IoT devices, e.g., [1, 4]
AWS and preliminary fission experiments. In Section 4, we give some preliminary performance results largely based on numerical computations of a binomial model for fission. The paper concludes with a summary in Section 5.

2 Summary overview of considered threats and proposed defenses

2.1 General assumptions of attack and defense

To reduce costs, multiple client end-users are assigned to each server and clients are not “containerized” within each server. A key assumption in the following is that a server under attack cannot determine which of its assigned client(s) are responsible. This said, it can be reliably determined whether a server is under attack.

In some defense frameworks, new clients are never added to servers (either proxies or replicas) deemed under attack. “Liberated” nominal clients may be consolidated into fewer servers to reduce the number of virtual machines or containers (instances) used by the overall system, particularly when the threat model is such that a single attacker can detectably affect a server (as assumed for application server replicas). However, as discussed below, individual proxy servers subjected to flooding attacks may be able to tolerate more than one attacker - so consolidating proxy servers deemed not under attack may result in servers that are subsequently detected under attack.

Some attacking clients are simple bots that cannot cope with reassignment to a new server. For such bots, continual shuffling of clients to servers may be a reactive defense. For reactive defense, we focus herein on other malware that is more resilient and can continue its attack even after it is transferred to another server (or the server just modifies its IP address).

2.2 Proactive moving-target defense against botnet reconnaissance phase

The botmaster needs to gather the current IP addresses of as many proxies as possible before launching its DDoS attack. During its reconnaissance phase, bots needs to engage in sessions of “normal” length and activity to avoid detection (avoid blacklisting).

Suppose the coordination server attempts to protect against botnet reconnaissance by periodically changing (via DNS) proxy IP addresses. In one approach, if a proxy is designated by the coordination server to have its IP address changed at time some time $t$ in the future, it receives no new clients in the time period $[s, t]$ where $t - s$ is expected to be longer than a session of a (nominal) client. Clients with active sessions during a proxy’s IP address change may be interrupted (the proxy may not assist clients in re-establishing the interrupted sessions). Some bots may not be able to re-establish sessions and need to launch
new ones, while nominal clients may be able reestablish sessions with only minor delay. Also, proxies may change their IP addresses individually (even at random) or collectively at the same time. In the latter case, all proxies known by the botnet are void.

Each proxied multiserver system is a tenant of a datacenter. Suppose that, as a service, the datacenter makes available a large pool of external IPv4 IP addresses available for proactive moving-target defense. Tenants wishing to engage in moving-target defense would dynamically share the entire pool. This would be more effective and less costly than the individual tenants managing their own sets of addresses that would need to be much larger in number than their proxies, or periodically asking the datacenter for fresh addresses. Naturally, under IPv6, a vast number of addresses would be available so that each tenant could cost-effectively mount such moving-target defense on their own.

2.3 Overhead associated with attack detection and response

Again, a key assumption is that a server under attack cannot determine which of its assigned client(s) are responsible. This said, it can be reliably determined whether a server is under attack either through the use of:

- probing “canary” clients that can determine when response times of their mock workloads have grown too high,
- heartbeat signals between proxy/replica servers and coordination server,
- host-based (HIDS) security checks in OS of a VM managing a number of containers\(^2\) in which proxies or replicas operate - this could be part of general server and client.

The last is part of general server and client diagnostic performance checks that are continually run by many public-cloud providers.

Note that a proxy subjected to a DDoS flood may be able to tolerate more than one attacker, \(i.e.,\) attack detection may not occur unless a server is subjected to \(A > 1\) attackers.

Response to volumetric attacks may require employing different physical machines if the downlink (ingress) network I/O is partitioned among its VMs/containers. Response may include spare servers that are kept on standby, \(i.e.,\) idling hot spares\(^3\).

Of course, the overhead of reactive defense also includes the IT resources required for the (shuffling/fission) defense itself (\(e.g.,\) network I/O, disk I/O) and delays experienced especially by stateful nominal clients. Note that, generally, clients have little state in proxies to shuffling clients among proxies would require relatively little overhead compared to migrating stateful clients among

\(^2\)or or hypervisor of a physical server managing a number of VMs, \(i.e.,\) Security-as-a-Service

\(^3\)To fairly account for the use of hot spare attack, one ought to compare performance with \(M\) servers and no hot spares against the use of \(M - H\) servers with \(H\) hot spares.
replica application servers [3, 6] residing on different physical machines. Also note that fission defenses for application servers might only involve dynamically “containerizing” clients within their existing virtual machines and not migrating them to application servers on other physical machines.

2.4 Fission of replicas against low volume attacks targeting them

Low volume threats (e.g., [19] and more recently [10]) are such that just one attacker can take down a replica application server. The idea of a \( n \)-fission defense is that \( n \) containers are spun-up for each container housing a server detected under attack (and this typically on the same physical server). The clients assigned to the attacked server are equally divided among the \( n \) resulting containers, and attack detection is repeated. Containers housing only nominal clients (i.e., containers deemed not under attack) may be consolidated to reduce the number of containers in play. Fission and detection are repeated until the attacking clients are sufficiently quarantined.

Note that if fission is used for high volume attacks targeting proxies, more specifically targeting (external) network I/O, then the new containers may need to reside on separate physical machines for subsequent attack detection. In this case, fission defense would be much more complex and costly.

Clearly, a basic assumption of fission defense is that always individually “containerizing” every clients within each server is too costly from a performance (and detection) point-of-view.

2.5 Shuffling of proxies against high volume attacks targeting them

Classical high volume attacks include TCP SYN floods and reflector attacks. To protect proxies against high volume attacks, all clients using proxies detected under attack could randomly reassigned (to these attacked proxies). By chance, some attacked proxies are assigned only nominal clients. As with the fission defense, attack detection and shuffling among attacked servers is repeated until attacking clients are sufficiently quarantined.

Shuffling can also be used on replica application servers to protect against low-volume DDoS attacks targeting them. Also, shuffling can be used in concert with a client reputation system.

3 Developing experimental set-up

There is substantial prior work on DDoS attack experimentation [14, 18, 12, 17, 15] spanning simulation, emulation, benchmarks and metrics. We are developing our defense system prototype on the Amazon AWS platform.

The system uses a combination of Linux Virtual machines (VMs) and Docker containers to implement and test the defense approach.
3.1 Defense Experimental Setup

The defense consists of a layer of indirection proxies which are used to facilitate shuffling of clients and also serve to hide the application replica layer. The indirection proxies are each launched from an indirection manager VM. A single VM will run an indirection manager and many indirection server Docker containers. The application replica layer is similar. A single VM will run a replica manager process which then launches multiple replica Docker containers within the VM. Each replica container executes a small management process and the protected service (e.g. a webserver).

To scale the system, any number of VMs can be launched and in turn add replica containers and indirection proxy containers into the defense.

The overall defense is controlled by another VM running a coordination server. This server is used to assign clients to both replica and indirection containers. Additionally, state information about each container and VM comes to the coordination server which can bring new servers online and remove others as needed.

To facilitate testing in the prototype a DNS server is also used which allows the coordination server to register DNS names for each of the defense servers. The initial contact from the client comes to a forwarding server which redirects the client to the initial proxy location based on input from the coordination server. To fully simulate a typical n-tier architecture we have also added a back-end MySQL database VM accessible by the web tier on the replica containers. The full architecture is shown in Figure 2 below.

![Figure 2: Physical architecture of the defense and test setup on AWS.](image-url)
3.2 Test Experimental Setup

To facilitate testing we have also built test infrastructure using a similar design as the defense. The test infrastructure has a test controller VM which configures and launches various attacks and load benchmarks against the defense system. The test controller communicates with any number of Bot Manager VMs. Each BotManager VM instantiates multiple Docker containers within the VM to launch attacks from multiple IPs and simulate many users accessing the system. The test architecture can create both nominal clients and attackers running various types of volumetric and application level attacks.

3.3 Experimental Capabilities

Using the described setup the team has implemented multiple types of shuffling to evaluate their differences. These include:

- Reactive random shuffling: Randomly shuffling replica users attempting to create "clean" replicas to label nominal clients.
- Reactive reputation-based shuffling: Randomly shuffling replica users while maintaining reputations to accurately label all clients.
- Fission-based shuffling: Shuffling clients on attacked replicas to new servers to isolate attackers. See Section 4.1.
- Pro-active random shuffling: Shuffling indirection proxies to confound the attacker's reconnaissance phase. See Section 2.2.

In addition to multiple shuffling types the system is designed to support both stateful and stateless clients. Stateless clients are able to be redirected to other servers without transferring state. This would be typical HTTP(S) traffic. Stateful clients must transfer state to the new target server. These are typically streaming media services such as video or VOIP streams. Members of our implementation team are working on efficient reassignment of stateful client reassignment techniques.

The following numerical evaluations in Section 4 agree with experiments implemented using the defense and testing system on AWS.

4 Fission: numerical performance evaluations

In the following, let

- $M$ be the number of servers
- $U = E_u$ be the mean number of nominal users
- $K = E_\kappa$ be the mean number of attacker users
Assume an equal number of users are assigned to each server. The following analysis is based on the binomial distribution:

\[ a(\cdot) = \text{binom}(v, p) \text{ where } v = \frac{U + K}{M}, \quad p = \frac{K}{U + K} \]

i.e., \[ a(i) = \binom{v}{i} p^i (1 - p)^{v-i} \text{ for } i \in \{0, 1, 2, ..., v\} \],

Note that \( K = \mathbb{E}_K = Mvp \) and \( U = \mathbb{E}_U = Mv(1 - p) \). Also note that the standard deviations \( \sigma(\kappa) = \sqrt{Mvp(1 - p)} = \sigma(u) \) so that \( \sigma(\kappa)/\mathbb{E}_K \text{ and } \sigma(u)/\mathbb{E}_U \text{ are small for large } M \).

In the following, we will evaluate performance primarily in terms of the number of nominal (not attacking) clients entirely unaffected by or liberated from the attack. Overhead includes number of additional containers/VMs used and delays experienced by clients owing to the defense itself. If the number of attackers per server is not limited, a kind of Stirling distribution of the second kind can be used [9] instead of the binomial model sketched in the following.

### 4.1 Binomial model for fission

The following model is based on the binomial distribution assuming an equal number of users are assigned to each server. If the number of attackers per server is unlimited, a kind of Stirling distribution of the second kind can be used instead [9]. Define the following binomial distribution

\[ a(\cdot) = \text{binom}(v, p) \text{ where } v = \frac{U + K}{M}, \quad p = \frac{K}{U + K} \]

i.e., \[ a(i) = \binom{v}{i} p^i (1 - p)^{v-i} \text{ for } i \in \{0, 1, 2, ..., v\} \],

Recall \( a = \text{binom}(v, p) \text{ where } v = (U + K)/M, \quad p = K/(U + K) \) and let

\[ \phi(l|k; v) = \binom{k}{l} \binom{v-k}{v/2-l}/\binom{v}{v/2}, \]

where \( \binom{b}{c} = 0 \text{ if } c < 0 \text{ or } c > b \).

The mean number of nominal users not attacked because their servers initially had no attacking users is

\[ M a(0)v = M(1 - p)^v v \]

The mean number of nominal users liberated by \( n \) binary fission steps is

\[ M \sum_{k_0=1}^{v} a(k_0)\Lambda_n(k_0; v) \]

where each binary tree rooted at a replica server initially attacked by \( k_0 \) users is accounted by \( \Lambda_n(k_0; v) \).
After one binary fission step, the number of liberated nominal users is

\[ \Lambda_1(k_0; v) = \sum_{k_1=0}^{k_0} \phi(k_1\mid k_0; v) \left( 1\{k_1 = 0\} \frac{v}{2} + 1\{k_1 = k_0\} \right) \tag{1} \]

and the following recursion holds for \( n \geq 2 \):

\[ \Lambda_n(k_0; v) = \sum_{k_1=0}^{k_0} \phi(k_1\mid k_0; v) \left( \Lambda_{n-1}(k_1; \frac{v}{2}) + \Lambda_{n-1}(k_0 - k_1; \frac{v}{2}) \right), \tag{2} \]

where

\[ \Lambda_n(k; x) = \begin{cases} 0 & \text{if } x = k \\ x & \text{if } k = 0 \end{cases} \]

This framework is easily to \( m \)-ary fissions for \( m > 2 \).

To count the mean number of containers used after \( n \) fission steps (number of leaves in the tree, a measure of overhead), replace terms \( v/2^i \) with 1 in (1):

\[ \Lambda_1(k_0; v) = \sum_{k_1=0}^{k_0} \phi(k_1\mid k_0; v) (1\{k_1 = 0\} + 1\{k_1 = k_0\}) = \phi(0\mid k_0; v) + \phi(k_0\mid k_0; v), \]

and use the same recursion (2). i.e., \( \Lambda_n(k; x) = 1 \) if \( k = 0 \).

Figures 3 and 4 depict typical numerical performance results using the binomial model which, again, agree with simulations. Regarding these figures recall that prior to the first fission (i.e., after zero fission steps), the mean fraction of affected nominal clients is \( 1 - M(1 - p)v/U \).

5 Summary

In summary, we presented an overview of proactive and reactive techniques to defend a generic proxied multiserver application in the public cloud against DDoS attacks by botnets, including a proactive moving-target technique that periodically changes IP addresses of proxy servers to combat the botnet’s reconnaissance phase, and a reactive shuffling approach to combat volumetric attacks targeting the proxies. We described a “fission” based approach to combat low-volume attacks targeting the replica application servers. A preliminary emulation study and a numerical study based on a binomial model of the attack characterized the performance of the fission defense.
Figure 3: Mean fraction of affected nominal clients after one binary fission stage for $U = 50000$, $M = 500, 1000, 5000$, and $0 \leq K \leq 5000$

Attack-reactive fission and shuffling frameworks reassign clients to servers to combat low volume attacks targeting application servers and high-volume floods targeting proxy servers.

Preliminary numerical results of the performance of the defenses were presented.

References


Figure 4: Mean number of unaffected and liberated nominal clients after multiple binary fission stages


