Kill 'em All -- DDoS Protection Total Annihilation!

Tony T.N. Miu[1], W.L. Lee[2], Alan K.L. Chung[2], Daniel X.P. Luo[2], Albert K.T. Hui[2], and Judy W.S. Wong[2]

[1]Nexusguard Limited
`tony.miu@nexusguard.com`

[2]Network Threats Information Sharing and Analysis Center (NT-ISAC)
Bloodspear Labs
`{leng,alan,daniel,albert,judy}@bloodspear.org`

Abstract. With the advent of paid DDoS protection in the forms of CleanPipe, CDN / Cloud or whatnot, the sitting ducks have stood up and donned armors... or so they think! We're here to rip apart this false sense of security by dissecting each and every mitigation techniques you can buy today, showing you in clinical details how exactly they work and how they can be defeated.
Essentially we developed a 3-fold attack methodology:

1. stay just below red-flag rate threshold,
2. mask our attack traffics inconspicuous,
3. emulate the behavior of a real networking stack with a human operator behind it in order to spoof the correct response to challenges,
4. ???
5. PROFIT!

We will explain all the required look-innocent headers, TCP / HTTP challenge-response handshakes,JS auth bypass, etc. etc. in meticulous details. With that knowledge you too can be a DDoS ninja! Our PoC attack tool "Kill-em-All" will then be introduced as a platform to put what you've learned into practice, empowering you to bypass all DDoS mitigation layers and get straight through to the backend where havoc could be wrought. Oh and for the skeptics among you, we'll be showing testing results against specific products and services.

Keywords: DDoS mitigation, DDoS, large-scale network attack

1 Introduction

DDoS attacks remain a major threat to internet security because they are relatively cheap yet highly effective in taking down otherwise well-protected networks. One need look no further than the attack on Spamhaus to realize the damage potential – bandwidth clog peaked at 300Gbps, all from a mere 750Mbps generated attack traffic [1]!

In the following sections, we first examine DDoS attacks observed in the wild and commercially available mitigation techniques against those attacks, with brief discussion on each technique’s inherent weaknesses. Next, we introduce bypass mechanisms that exploit these weaknesses and, through illustrating our proof-of-concept (PoC) tool “Kill ’em All”, show how bypass mechanisms can be combined to achieve total bypass, thereby defeating defense-in-depth design typically adopted in DDoS mitigation solutions.

To conclude, we substantiate our claim with testing results against specific mitigation solutions, and propose a next-gen mitigation methodology capable of defending against “Kill ’em All”-type attacks.

2 A Quick Overview

DDoS attacks are simple yet highly effective. These days DDoS attacks run so rampant they form a part of the everyday internet traffic norm. Over the past decade, DDoS attack and defense technologies have evolved tremendously through an arms race, leading to old-school protections completely losing their relevance in the modern days. Before we examine the technical details let’s start with a classification system [2].

2.1 DDoS Attack Classification Quadrant

It is helpful to classify DDoS attacks according to their attack volume and complexity, with reference to **Error! Reference source not found.** below.

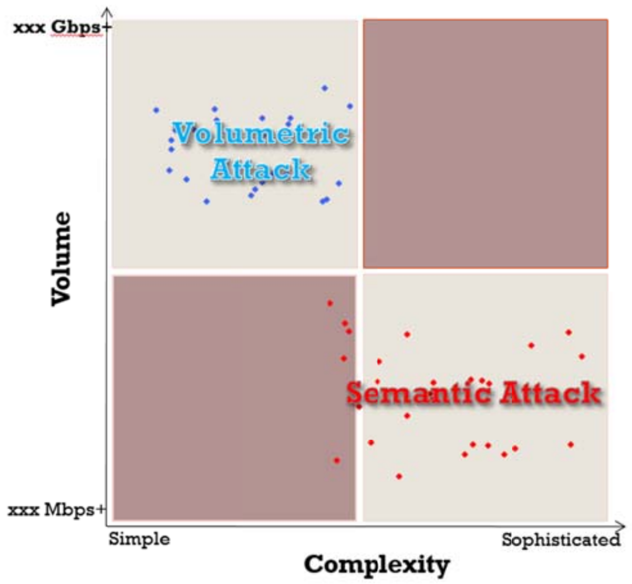


Figure 1. DDoS attack classification quadrant

The crudest form of DDoS attack are volumetric DDoS attacks, whereby a huge volume of traffic pours into the victim in a brute-force manner, hogging all bandwidth otherwise available for legitimate purposes. Execution is expensive, as the attacker would have to send traffic whose volume is on par with the victim's spare capacity. This translates to a higher monetary cost associated with hiring botnets. The age-old ping flood is a prime example.

Semantic DDoS attacks work smarter, amplifying firepower by exploiting semantic contexts such as protocol and application weaknesses [3]. This effectively tips the balance in the attacker's favor, making attacks much cheaper. Examples of semantic attacks include Slowloris [4] and Smurf [5] attacks, as well as application level attacks that make excessive database lookups in web applications.

A third category, *blended DDoS attacks*, aims to achieve stealthy attacks through blending into legitimate traffic, practically rendering ineffective most countermeasures designed to filter out abnormal, presumably malicious, traffic. HOIC [6] is an example of an attack that employs blending techniques via randomized headers.

Note that these categories are by no means mutually exclusive. For instance, blended attacks that also exploit application weaknesses are not at all uncommon in the wild.

2.2 DDoS Mitigation Techniques

Mitigation are techniques used to reduce the impact of attacks on the network. The upcoming paragraphs [2] shall explain the three main types of mitigation such as traffic policing, black/white listing and proactive resource release as shown in Figure 2.

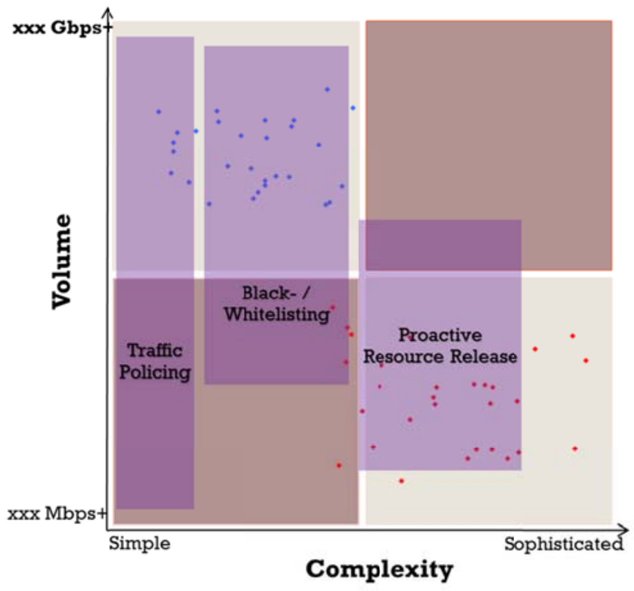


Figure 2. DDoS mitigation techniques

Against volumetric attacks, a direct mitigating tactic employs traffic policing to curb attack traffic. Common implementations typically involve baseline enforcement and rate limiting, whereby traffic that exceeds a capacity threshold or otherwise violates predetermined traffic conditions (baseline profile) are forcibly suppressed to ensure conformance with capacity rules. This is usually achieved through selective packet dropping (traffic policing), or outright blacklisting of infringing traffic sources.

Blacklisting is essentially a short circuit mechanism aimed at cutting down the tedious work of having to classify individual flows by outright dropping traffic from entire IP addresses for a certain period of time or for a certain amount of traffic volume immediately upon identification of one attack from those sources. Blacklisting cannot be permanent, as IP addresses can be dynamically assigned and zombied computers can be repaired. In contrast to blacklisting, whitelisting preapproves traffic from entire IP addresses for a certain period of time or for a certain amount of volume upon determining those sources are well behaving.

Another approach that is most effective against resource starvation attacks is proactive resources release whereby resources prone to starvation are forcibly freed up. For compatibility and scalability reasons, commercial mitigation solutions are usually deployed externally to individual computer systems and networking devices, treating them as black boxes. This precludes housekeeping measures that require host-based mechanisms such as enlarging the TCP concurrent connection pool. That said, resource freeing by means of TCP connection reset can be instrumented externally—sending a TCP RST packet to a server host is sufficient to close and free up a connection. For TCP-based DDoS attacks, forceful TCP connection reset is a very practical control mechanism. Resource holding attacks like Slowloris [2] are best handled with proactive resources release.

2.3 DDoS Detection Techniques

Referring to Figure 3 below, at the top left-hand corner we have high volume simple attacks which are dead obvious in their footprints, easily detectable with rate measurement and baselining methods via straightforward SNMP or Netflow monitoring respectively; at the other end of the spectrum at the bottom right-hand corner attacks get smaller and cleverer, for those, protocol sanity and behavior inspection, application-level examination (via syslog mining / application log analysis), or even protocol statistics and behavior big-data analysis are often required for detection.

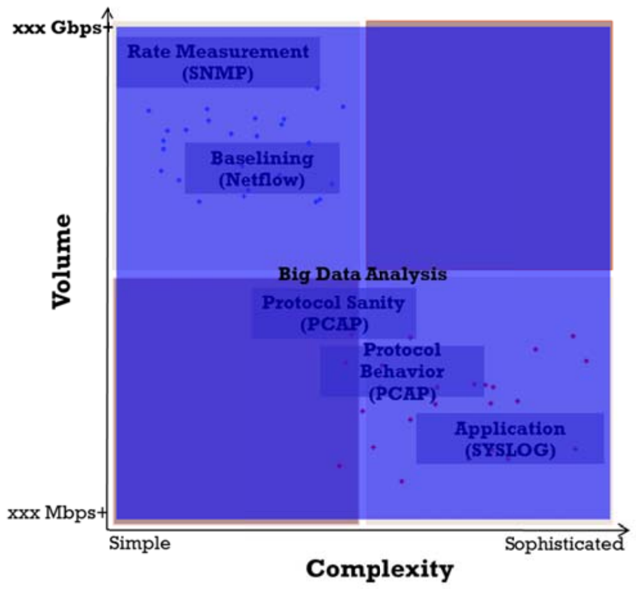


Figure 3. DDoS detection techniques

Indeed, rate metering and baseline enforcement can be applied to specific source IP addresses or to address ranges such as entire subnets. But, a pure traffic policing approach cannot correlate across unrelated sources, because that would require visibility into traffic characteristics deeper than just capacity rule violations.

Semantic attacks usually follow specific patterns. For instance, Teardrop Attack's telltale signature is its overlapping IP fragments. Checking for these signatures may not be trivial to implement but nevertheless provides definite criteria for filtering. It is for this reason that protocol sanity and behavior checking are mostly effective for catching known semantic attacks.

Modern application-level attacks do their dirty works at upper layers, exhibiting no abnormal behavior at the lower layers. Detection therefore would have to work on application-level behaviors, via syslog or application log analysis methods.

Traffic statistics and behavior big data analysis aims at building a baseline profile of traffic such that significant deviation at runtime can trigger a red flag. Generally data-mining can work on profiling protocol parameters, traffic behaviors, and client demographics.

3 Authentication Bypass

In response to mitigation techniques that excel at filtering out malformed traffic, blended attacks gained popularity. They strive to evade filtering by mimicking legitimate traffic, such as for HTTP requests to bear believable real-world User-Agent string, and have variable lengths.

3.1 TCP SYN Authentication

With this method, the authenticity of the client's TCP stack is validated through testing for correct response to exceptional conditions, such that spoofed source IPs and most raw-socket-based DDoS clients can be detected. Common tactics include sending back a RST packet on the first SYN expecting the client to retry, as well as deliberately sending back a SYN-ACK with wrong sequence number expecting the client to send back as RST and then retry.

The best approach to defeating this method is to have the OS networking stack handle such tests. There are essentially two methods:

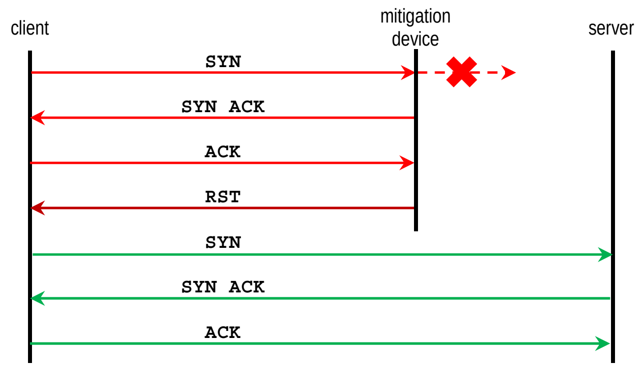


Figure 4. TCP reset

TCP Reset —Anti-DDoS gateway will send a reset (RST) flags to forcefully reset the backend's established TCP connections (those having successfully completed three-way handshaking) as shown in Figure 4. It is the most common method for TCP connection verification, as purpose-built DDoS bots may not have the retry logic coded, unlike a real browser. However, a drawback of this method is that an error page will show up on the browser, confusing the user to no end, who has to manually reload the web page.

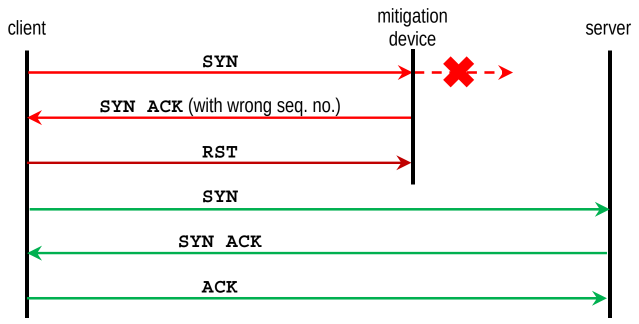


Figure 5. TCP out-of-sequence

TCP Out-of-sequence — Unlike TCP Reset, the anti-DDoS gateway can deliberately pose a challenge to the client by sending SYN-ACK replies with an out-of-sequence sequence number as shown in Figure 5. Since the sequence number is incorrect, the client is supposed to reset the TCP connection and re-establishes the connection again. Again a purpose-built bot would likely not be so smart. Compare with TCP Reset, this method has the added advantage that it would not introduce strange user experience.

3.2 HTTP Redirect Authentication

The basic idea is that a legitimate browser will honor HTTP 302 redirects. As such, by inserting artificial redirects, it would be safe to block non-compliant clients.

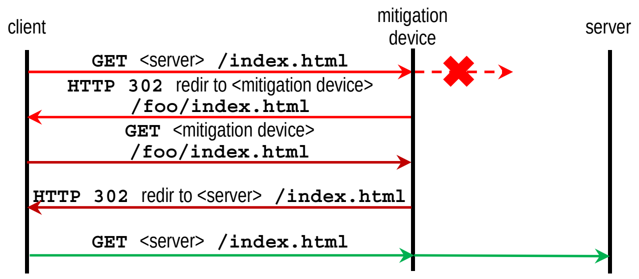

client
mitigation device
server
GET <server> /index.html
HTTP 302 redir to <mitigation device> /foo/index.html
GET <mitigation device> /foo/index.html
HTTP 302 redir to <server> /index.html
GET <server> /index.html

Figure 6. HTTP redirect authentication

Clearly, it is not particularly difficult to implement just enough support for HTTP redirects to fool HTTP Redirect Authentication. +The purpose of this authentication is to distinguish botnet and HTTP compliant applications.

3.3 HTTP Cookie Authentication

For similar purpose, this method as shown in Figure 7 works like, and is usually used together with, HTTP Redirect Authentication. Essentially, browser's cookie handling is tested. Clients that do not carry cookies in subsequent HTTP requests are clearly suspect and can be safely blocked.

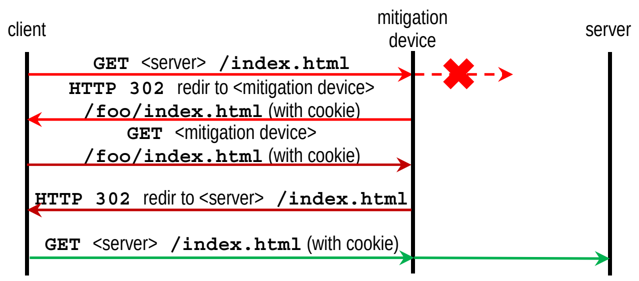

client
mitigation device
server
GET <server> /index.html
HTTP 302 redir to <mitigation device> /foo/index.html (with cookie)
GET <mitigation device> /foo/index.html (with cookie)
HTTP 302 redir to <server> /index.html
GET <server> /index.html (with cookie)

Figure 7. HTTP cookie authentication

Some mitigation devices may allow administrator to configure custom field name for the HTTP cookie instead of standard one as shown in Figure 8. However, not all browsers will support this feature and thus it is not widely used.

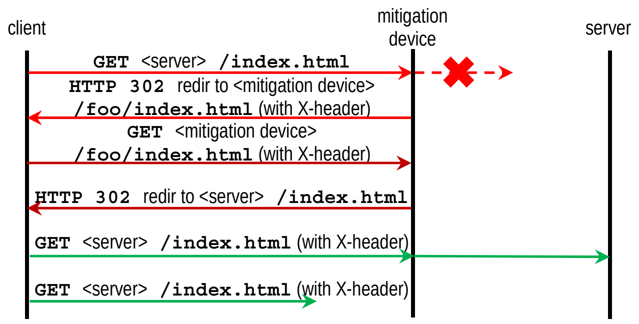

client
mitigation device
server
GET <server> /index.html
HTTP 302 redir to <mitigation device> /foo/index.html (with X-header)
GET <mitigation device> /foo/index.html (with X-header)
HTTP 302 redir to <server> /index.html
GET <server> /index.html (with X-header)
GET <server> /index.html (with X-header)

Figure 8. HTTP cookie authentication with header token

As in adding support for HTTP Redirect Authentication, cookie support does add additional complexity and reduces raw firepower in DDoS attacks, but is nevertheless easily to implement.

3.4 JavaScript Authentication

With JavaScript Authentication, a piece of JavaScript code embedded in the HTML is sent to clients as a challenge as shown in Figure 9. Obviously, only clients equipped with a full-fledged JavaScript engine can perform the computation. It would not be economical for DDoS attack tools to hijack or otherwise make use of a real heavyweight browser to carry out attacks. The purpose of Javascript authentication is to identify whether the HTTP request is send from a real browser or not.

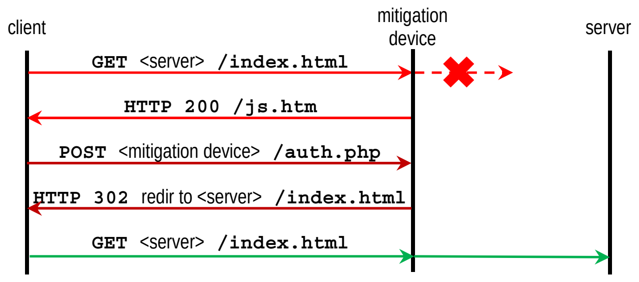


Figure 9. Javascript authentication

An extended implementation would make use of UI elements such as JavaScript dialog boxes or detecting mouse movements in order to solicit human inputs. Going this far would impede otherwise legitimate automated queries, making this mechanism only suitable for a subset of web sites designed for human usages, but not those web APIs such as REST web services.

3.5 CAPTCHA Authentication

A very heavy-handed approach that involves human intervention whereby CAPTCHA challenges are inserted into suspicious traffic as shown in Figure 10. If the client end is successful in solving the CAPTCHA, it will be whitelisted for a certain period of time or for certain amount of subsequent traffic, after which it will need to authenticate itself again. The purpose of this authentication is to distinguish whether the request is initiated by a real human or a bot.

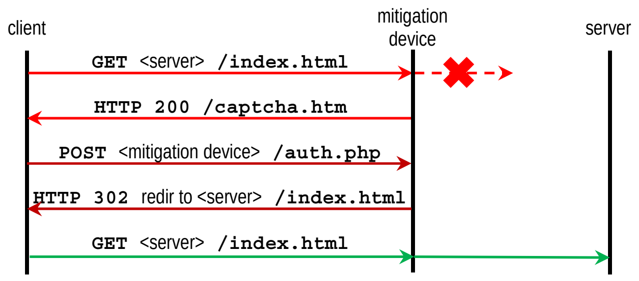
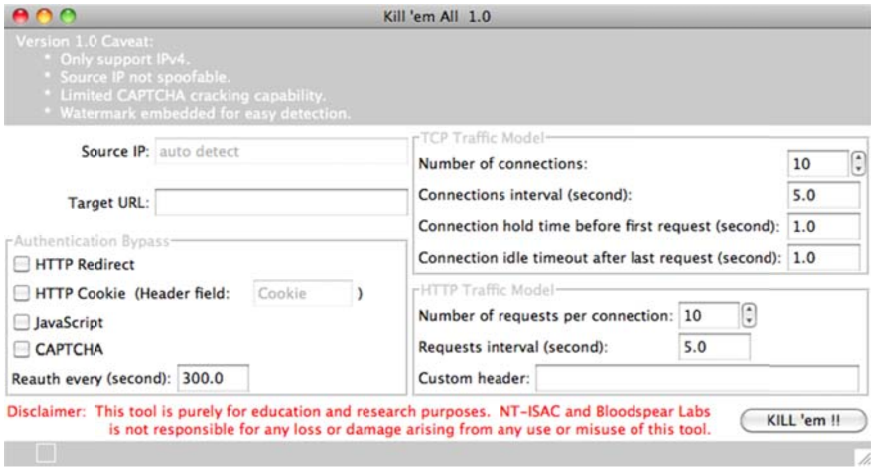


Figure 10. CAPTCHA authentication

This method is, in itself, rather intrusive and in practice used only sparingly. While far from easy, automated means to solve CAPTCHA do exist and is a topic of ongoing research.

4 PoC Tool Design and Implementation

Through extensive testing we have developed a sure-fire methodology capable of bypassing all commercial mitigation solutions. The key idea is to satisfy source host verification (authentication) so as to be cleared of further scrutiny, and then send attack traffics staying just below traffic threshold. A proof-of-concept tool “Kill ‘em All” developed to demonstrate the effectiveness of this approach, is shown in Figure 11.


Kill 'em All 1.0

Version 1.0 Caveat:
- Only support IPv4.
- Source IP not spoofable.
- Limited CAPTCHA cracking capability.
- Watermark embedded for easy detection.

Source IP: auto detect

Target URL:

Authentication Bypass
- ☐ HTTP Redirect
- ☐ HTTP Cookie (Header field: Cookie)
- ☐ JavaScript
- ☐ CAPTCHA

Reauth every (second): 300.0

TCP Traffic Model

Number of connections:	10
Connections interval (second):	5.0
Connection hold time before first request (second):	1.0
Connection idle timeout after last request (second):	1.0

HTTP Traffic Model

Number of requests per connection:	10
Requests interval (second):	5.0
Custom header:	

Disclaimer: This tool is purely for education and research purposes. NT-ISAC and Bloodspear Labs is not responsible for any loss or damage arising from any use or misuse of this tool.

Figure 11. Proof-of-Concept Tool "Kill 'em All"

In practice, an entire suite of authentications mechanisms covering multiple layers work in unison to challenge traffic sources. We examine the approach taken to defeat each of them below.

After successful authentication, oftentimes the source IP addresses would have been place on to the whitelist, meaning a number of expensive checks will be skipped in the interest of performance. This affords the attacker a certain period of free reign over the cleared network path.

As described in [2]. IP addresses would just be whitelisted for a period of time. Therefore, the authentication process will repeat after certain time interval, say for every 5 minutes. All attack requests sent after re-authentication will use the new authentication code.

Nevertheless we found that certain traffic rules must be observed still, as low-level attack discovery mechanisms such as rate measurement are usually not disabled even when the source is whitelisted. Tactics against this is outlined below as well.

“Kill ‘em All” was designed to use the OS stack to handle user requests. Using the OS networking library, it can fully simulate as a bona fide web client. Moreover, even if the anti-DDoS device attempt source host verification, like TCP SYN auth, TCP redirect or some JavaScript test, our program still is capable of “proving” its authenticity, by relegating the task to a real web browser.

“Kill ‘em All” will attempt bypass 3 times for each HTTP redirect challenge, this way TCP Reset and the TCP Out-of-Sequence auth can be properly defeated. Indeed this is how a real client will handle retries and redirects.

4.1 Cookie Authentication

For HTTP cookie authentication, our tools will spawn a web browser to process the cookie request. Cookie is attached to all subsequent attack requests we sent.

4.2 JavaScript Authentication

“Kill ‘em All” can fully handle JavsScript thanks to embedded JavaScript engine. This JavaScript capability makes it look like a real browser, because JavaScript capability is very uncommon in DDoS bots.

For proper handling of Javascript, we have incorporated the V8 JavaScript engine. Ideally a full DOM should be implemented but for the purpose of passing authentication a subset of DOM was sufficient.

Attack tools however, can incorporate standalone JavaScript engines such as Spidermonkey or V8 which are relatively lightweight and would not bog down attacks too much. As of this writing, the major challenge with this bypass method lies with adequate DOM implementation.

4.3 CAPTCHA Authentication

Widely considered the final frontier for source host verification, CAPTCHAs are not completely unbreakable by automated means. In “Kill ‘em All” we have implemented CAPTCHA capability whereby CAPTCHA challenges are automatically processed and answered. Given sufficient baseline training, the success rate can be near prefect.

We couldn’t find a light-weight cross-platform CAPTCHA library, so we’ve implemented our own. The algorithm first convert the CAPTCHA image to black-and-white to enhance contrast. Then a 3 by 3 median filter is applied to remove background noises such as dots and thin lines. Afterwards words are segmented into individual characters and their boundaries detected. Finally, characters are compared against trained baseline based on simple pixel differences. Against NSFocus ADS, success rate of nearly 50% was achieved.

Some CAPTCHA might have rotated or curved characters. This will require a more complex algorithm such as vector quantization or neural network for recognition. As for re-CAPTCHA, their audio CAPTCHA functionality which is much weaker than their standard visual counterpart—simple voice recognition algorithm will be sufficient for breaking it.

4.4 TCP Traffic Model

"Kill 'em All" provides tunable TCP traffic parameters as shown in Figure 12 so that different kinds of DDoS attacks can be executed. The number of connections, connections interval, connection hold time before first request, connection idle timeout after last request are exposed to the user, through setting them to different values, different combinations can be constructed. Many protection systems can be defeated with specific parameter profiles. Figuring out the right set of parameters is, however, a combination of art and science, and a lot of trial and errors.

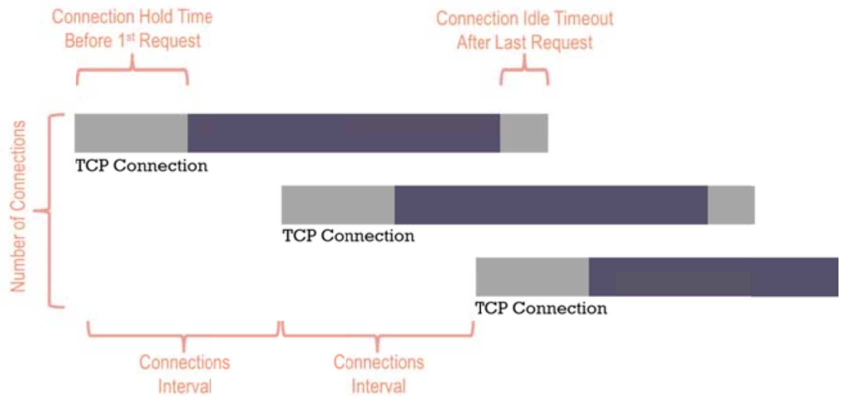


Figure 12. TCP timing controls

On the TCP/IP layer, historically due to many DDoS attack tools were written in a quick and dirty way using raw sockets for performance reason, resulting in non-compliant TCP/IP behaviors that can be used as a factor differentiating attack traffics from legitimate ones.

With web sites and web services gaining overwhelming dominance over the internet, so are contemporary attacks focusing on the HTTP layer. Smarter attacks and proliferation of botnets together serve to lessen the need for super high-efficiency attacks relying on raw sockets, a staple of the old time. The modern day application layer attacks are just as devastating, if not more so, then their old-school counterparts. Application layer attacks have the luxury of simply using the standard OS TCP/IP stacks. Other than easier to implement, this design approach has an added benefit of being able to pass any RFC conformance check.

As described in previous section, rate limiting mechanisms, be they time-based, volume-based, request-based or otherwise, can be defeated via careful assessment of the triggering threshold and control the rate of attack traffics to stay just below it. The reduction in firepower can be more than compensated with the use of large botnets.

4.5 HTTP Traffic Model

“Kill ‘em All” also provides tunable HTTP traffic parameters as shown in Figure 13 to offer various attack vector. For example, large number of requests per connection with short requests interval would yield a GET flood DDoS attack.

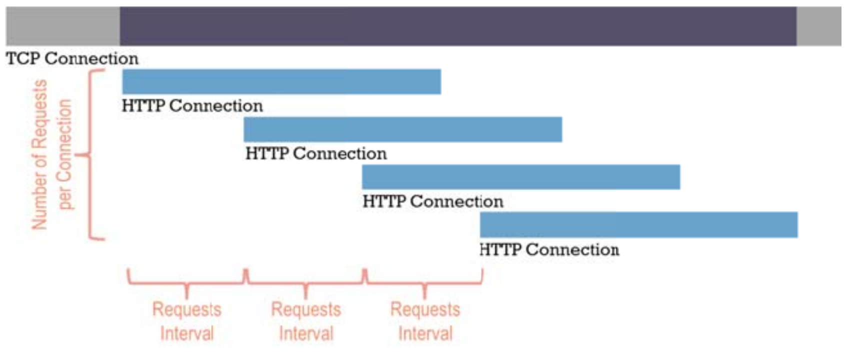


Figure 13. HTTP timing controls

In order to avoid being fingerprinted, we have implemented randomization for such attributes as User-Agent strings and packet sizes.

“Kill ‘em All” allow allows for the construction of certain web server or web app targeted attacks. For instance, CVE-2011-3192 Apache Range Header exploit can be constructed with a custom header setting of “Range: bytes=<long list of numbers with comma as delimiter>”.

5 Performance Testing

Tests were conducted against products:

1. Arbor Peakflow SP Threat Management System (TMS) version 5.7, and
2. NSFocus Anti-DDoS System (ADS) version 4.5.88.2.026

as well as cloud services:

3. Cloudflare, and
4. Akamai.

We are convinced that Arbor TMS and NSFocus ADS represent a majority of the market, with the former most prevalent among Fortune 500 enterprises and the latter deployed in most every publicly listed company in mainland China.

5.1 Testing Methodology

Tests were conducted against products and cloud services. For product testing an attack workstation was connected to a web site through the DDoS mitigation device under test. For cloud service testing a web site was placed under the protection of the service under test, and then subjected to attacks from a workstation directing attacks towards it through the internet.

In order to simulate normal short-term browsing conditions, in all tests a single TCP connection was used to carry a multitude of HTTP requests and responses. Under this vigorous arrangement not a single attack identification mechanism can be triggered lest the entire connection gets blocked.

During testing, attack traffic was sent to the backend at which point received traffic was compared against the original generated traffic. Bypass was considered successful if all attack traffic passed through intact.

5.2 Testing Results

Attacks with bypass capability were applied against individual detection techniques as implemented on the aforementioned products and services. During the attack, effectiveness of the attacks was evaluated and observations were recorded as shown in Table 1 below. A "✓" means the bypass was successful with no mitigation activity observed.

Detection Techniques	Arbor Peakflow SP TMS	NSFocus ADS	Cloudflare	Akamai
Rate Measurement / Baseline Enforcement	✓ (Zombie Removal, Baseline Enforcement, Traffic Shaping, Rate Limiting)	✓	N/A	N/A
Protocol Sanity & Behavior Checking	✓ (HTTP Countermeasures)	✓	N/A	N/A
Proactive Housekeeping	✓ (TCP Connection Reset)	✓	N/A	N/A
Big Data Analysis	✓ (GeoIP Policing)	— (Not implemented in ADS)	N/A	N/A
Malicious Source Intelligence	✓ (Black White List, IP Address Filter List, Global Exception List, GeoIP Filter List)	— (Not implemented in ADS)	N/A	N/A
Protocol Pattern Matching	✓ (URL/DNS Filter List, Payload Regex)	✓	N/A	N/A
Source Host Verification				

TCP SYN Authentication	✓	✓	N/A	N/A
HTTP Redirect Authentication	✓	✓	✓	N/A
HTTP Cookie Authentication	✓	✓	✓	N/A
JavaScript Authentication	— (Not implemented) in TMS)	✓	✓	N/A
CAPTCHA Authentication	— (Not implemented in TMS)	✓	✗	N/A

Table 1. Mitigation bypass testing results.

With reference to Arbor Network's A Guide for Peakflow® SP TMS Deployment[1], against TMSwe were able to defeat all documented or otherwise active detection techniques relevant to HTTP DDoS attacks, passing through the TMS unscathed.

Attacks against NSFocus ADS[2] were met with remarkable success despite the presence of heavy-handed defenses including CAPTCHA Authentication — we were able to achieve a remarkable 50% success rate solving ADS's CAPTCHA implementation with our OCR algorithms. Due to the shotgun approach to attack, and that getting whitelisted is a big win for the attacker, a 50% success rate for solving CAPTCHA is much more impressive than it may appear at first glance.

Cloudflare essentially employs JavaScript that implements all JavaScript, Cookie and Redirect Authentications in one. We were successful in defeating them all and pushing attack traffic to the backend. Even though Cloudflare does support CAPTCHA Authentication, we observed that its use is not particularly prevalent in the wild, and for the purpose of our PoC since we have already demonstrated a workable solution against CAPTCHA for ADS, we have opted not to repeat this for Cloudflare.

Akamai has implemented source host verification techniques in its security solutions for a few months now, with which according to marketing brochure [7] visitors will be redirected to a JavaScript confirmation page when traffic is identified as potentially malicious. However, despite our best effort sending big traffic to our testing site bearing random HTTP query strings (in order to thwart caching) we have been unable to trigger that feature. Whereas we cannot rule out the remote possibility that our test traffic was way below detection threshold, a much more plausible reason might be that our traffic was indistinguishable from that generated by a real browser.

---

1 http://www.arbornetworks.com/component/docman/doc_download/301-threat-management-system-a-technical-overview?Itemid=442

2 http://www.nsfocus.com/jp/uploadfile/Product/ADS/White%20Paper/NSFOCUS%20ADS%20White% 20Paper.pdf

6 Discussions and Next Generation Mitigation

In this era of blended attacks, detection methods designed to pick out bad traffics are rendered fundamentally ineffective. The reason why today to a certain extent they still work is mainly due to implementation immaturity (e.g. the lack of ready-to-use JavaScript engine with a workable DOM). Obviously these hurdles can be easily overcome given a little more time and development resources, as our research demonstrated.

A notable exception is the use of CAPTCHA. Despite the fact that we have also demonstrated defeating certain CAPTCHA implementations in use on security products, and that there have been promising results from fellow researches [8] as well, admittedly CAPTCHA still represent the pinnacle of source host verification technique. However, CAPTCHA is necessarily a heavy-handed approach that materially diminishes the usability and accessibility of protected web sites. Specifically, automated queries and Web 2.0 mashing are made impossible. This shortcoming significantly reduces the scope of its application. It is therefore not surprising that CAPTCHA is often default off in security service offerings.

6.1 Next Generation Mitigation

Seeing as that the underlying issue with a majority of DDoS attacks these days is their amplification property, which tips the cost-effectiveness balance to the attackers' favor, we are convinced that a control mechanism based on asymmetric client puzzle is the solution, as it presents a general approach that attacks directly this imbalance property, making it a lot more expensive to execute DDoS attacks. Prior researches include the seminal Princeton-RSA paper [9] and [10].

7 Acknowledgement

This research was made possible only with data and testing resources graciously sponsored by Nexusguard Limited[3] for the advancement of the art.

References

[1] M. Prince, "The DDoS that Knocked Spamhaus Offline (And How We Mitigated it)," 20 March 2013. [Online]. Available: http://blog.cloudflare.com/the-ddos-that-knocked-spamhaus-offline-and-ho.

[2] T. T. N. Miu, A. K. T. Hui, W. L. Lee, D. X. P. Luo, A. K. L. Chung and J. W. S. Wong, "Universal DDoS Mitigation Bypass," in *Black Hat USA*, Las Vegas, 2013.

[3] http://www.nexusguard.com/

[3] C. Weinschenk, "Attacks Go Low and Slow," IT Business Edge, 3 August 2007. [Online]. Available: http://www.itbusinessedge.com/cm/community/features/interviews/blog/attacks-go-low-and-slow/?cs=22594.

[4] R. Hansen, "Slowloris HTTP DoS," 7 June 2009. [Online]. Available: http://ckers.org/slowloris/.

[5] Carnegie Mellon University, "CERT® Advisory CA-1998-01 Smurf IP Denial-of-Service Attacks," 5 January 1988. [Online]. Available: http://www.cert.org/advisories/CA-1998-01.html.

[6] J. Breeden II, "Hackers' New Super Weapon Adds Firepower to DDOS," GCN, 24 October 2012. [Online]. Available: http://gcn.com/articles/2012/10/24/hackers-new-super-weapon-adds-firepower-to-ddos.aspx.

[7] Akamai, "Akamai Raises the Bar for Web Security with Enhancements to Kona Site Defender," 25 February 2013. [Online]. Available: http://www.akamai.com/html/about/press/releases/2013/press_022513.html.

[8] DC949, "Stiltwalker: Nucaptcha, Paypal, SecurImage, Slashdot, Davids Summer Communication," 26 July 2012. [Online]. Available: http://www.dc949.org/projects/stiltwalker/.

[9] B. Waters, A. Juels, J. A. Halderman and W. F. Edward, "New Client Puzzle Outsourcing Techniques for DoS Resistance," in *ACM Conference on Computer and Communications Security (CCS)*, 2004, 2004.

[10] D. Stebila, L. Kuppusamy, J. Rangasamy and C. Boyd, "Stronger Difficulty Notions for Client Puzzles and Denial-of-Service-Resistent Protocols," in *RSA Conference*, 2011.

[11] T. Miu, A. Lai, A. Chung and K. Wong, "DDoS Black and White "Kungfu" Revealed," in *DEF CON 20*, Las Vegas, 2012.

[12] R. Kenig, "How Much Can a DDoS Attack Cost Your Business?," 14 May 2013. [Online]. Available: http://blog.radware.com/security/2013/05/how-much-can-a-ddos-attack-cost-your-business/.