Deloitte.

Offensive Defense – DDoS Disruption

A simple method to tarpit and mitigate the Dirt Jumper Drive –smart attack

The attached paper describes a countermeasure for the "-smart" attack. This countermeasure is an example of actionable, offensive security. Security research surrounding DDoS attacks has traditionally examined ways to harden networks against intrusions. This research demonstrates the strategies available for active intervention in the attack process.

Introduction to Dirt Jumper

Modern DDoS attacks are generally executed via a botnet, a large collection of machines that have been infected with a specialized malware that can effectively disable the function of a targeted system or device by flooding it with communication requests. Hosting providers have responded by developing pattern-based detection capabilities to support rapid response. However, malware developers have adjusted with recently developed capabilities to circumvent the measures employed by anti-DDoS hosting providers.

The Dirt Jumper variant, "Drive", is one of the most harmful strains of DDoS malware specially designed to avoid detection. Drive's DDoS engine allows for a wide variety of attack commands. This functionality included the ability to send HTTP requests with randomized User Agents, Refers, and Post requests. These changes made Drive a more flexible and resilient DDoS toolkit than many of its predecessors.

What is –smart?

Dirt Jumper allows the botnet administrator to utilize the attack command "-smart". This command was designed to follow redirections. It was simultaneously released with other significant updates, and the upgrade improved Dirt Jumper's DDoS functionality. However, the complexity of the "–smart" logic also exposed the malware to new weaknesses.

Actions and remediation

Deloitte's Advanced Solutions & Research Group effectively reversed engineered the "–smart" logic of Dirt Jumper and identified a mechanism of the malware that can be counter-exploited to significantly cripple the malware's function. This work is an example of how advanced malware can sometimes be countered not by signature- and controls-based strategies, but by actively countering the attacks with methods that actively intervene in the attack process. The primary contribution of this report is to show how the mechanism of the "–smart" attack can itself be exploited to prevent an attacking Dirt Jumper bot from reaching its desired target application webserver as well as tar-pitting the botnet, reducing its request rate more than a hundred fold.

Contact us

Vikram Bhat, Principal, Cyber Risk Services, Deloitte & Touche LLP, +1 973 602 4270, vbhat@deloitte.com
Christopher Stevenson, Head of Research and Development, Cyber Risk Services, Deloitte & Touche LLP, +1 201 499-0584, chstevenson@deloitte.com
Lance James, Head of Cyber Intelligence, Cyber Risk Services, Deloitte & Touche LLP, +1 760 262 4141, ljames@deloitte.com

When -smart is Stupid

A simple method to tarpit and mitigate the Dirt Jumper Drive -smart attack

Joel Lathrop
Advanced Research & Solutions Group
Deloitte & Touche LLP
jlathrop@deloitte.com

Lance James
Advanced Research & Solutions Group
Deloitte & Touche LLP
lancejames@deloitte.com

ABSTRACT

Over the past few years, a particularly virulent strain of distributed denial-of-service (DDoS) malware known as Dirt Jumper has emerged. It has progressed through several iterations and has recently developed capabilities to circumvent measures employed by certain anti-DDoS hosting providers; this new capability was exposed as a new attack type named `-smart`. The primary contribution of this paper is to show how the mechanism of the `-smart` attack can itself be exploited to prevent an attacking Dirt Jumper bot from reaching its desired target application webserver as well as tarpitting the botnet, reducing its request rate more than a hundred fold.

1. INTRODUCTION

In recent years, Distributed Denial-of-Service (DDoS) attacks have increasingly been raised to the forefront of Internet security concerns. What was once the domain of network security professionals has now become an issue affecting large public institutions and is being reported in the mainstream media [14]. A 2012 Neustar survey of North American IT professionals found that 35% of companies reported experiencing a disruptive DDoS attack [8], and a 2012 Arbor Networks survey of network operators found that 76% reported experiencing DDoS attacks against their customers [16]. These attacks can be expensive: Neustar's survey of e-commerce companies found that 36% reported a DDoS attack would cause outage losses of over $100K per hour [8].

While some of the most well-known DDoS attacks appear driven by ideology [14], there are other motivations. Frankly stated, there's money in DDoS. One of the simplest cases is ordinary extortion: An attacker threatens to launch a DDoS attack against a website unless he is paid a certain sum of money [3, 4]. A more elaborate case would be a DDoS against a financial institution shortly after attempting a fraudulent wire or ACH transfer; in this case the DDoS acts as a distraction to keep attention away from the fraudulent financial transaction [13]. In some situations, this application of DDoS has been associated with attempted fraud involving totals ranging from $180k to $1 million [17, p. 12].

Modern DDoS attacks are generally effected by means of a large collection of malware-infected machines, colloquially referred to as a botnet. In the case of a DDoS botnet, there is usually a specialized piece of malware used to control individual bots for the purposes of DDoS. One such strain of malware commonly used for this purpose is known as Dirt Jumper, and our research into its workings is the focus of this paper.

2. A SHORT HISTORY OF DIRT JUMPER

While we are primarily interested in the `-smart` functionality of the recent versions of Dirt Jumper Drive, it is helpful to approach the subject with a sense of historical context. So we will first briefly sketch the evolution of Dirt Jumper.

2.1 Evolution from Russkill to Drive

In mid-2011 ArborSERT, released a write-up on a new form of malware referred to as Dirt Jumper which appeared to be a descendant of the Russkill malware strain [19]. The researchers had observed it used for DDoS attacks against various ordinary Russian sites, but then also noticed it being used to attack a Russian electronic trading site, which suggested a financial motivation. Postings on underground forums around that time advertised Dirt Jumper version 3 as part of a DDoS-for-hire business.

Over time Dirt Jumper's popularity increased leading to a number of variants, such as Simple, September [5], Kahn [6], Pandora, Di BoTNet and Dirt Jumper version 5 [20]. This growth of variants may have been accelerated by multiple leaks of the Dirt Jumper source code. Eventually, the collective of Dirt Jumper variants began to supplant other prominent DDoS tool sets in the underground market for DDoS products and services [12].

Then a new Dirt Jumper variant named "Drive" arrived on the scene. It appeared to have a new DDoS engine, with new attack commands and a new, more expressive syntax for those commands [10]. The HTTP requests sent in certain types of DDoS attacks now had multiple possible `User-Agent` headers with randomized values for certain parameters within them, such as version numbers. They would also contain a `Referer` header with a randomly generated URL. Additionally, it was now possible to provide a skeleton for HTTP POST request attacks that could be filled in with

randomly generated values. These changes made Drive a more flexible and resilient DDoS toolkit than many of its predecessors.

2.2 Hosting provider response

This activity amongst the underground community did not go unnoticed. As more sophisticated DDoS malware was created, and the frequency of DDoS attacks rose, security researchers took note and looked for ways to detect and mitigate the attacks. Common techniques used would look for a simple, static repeated string pattern in the DDoS traffic, such as a fixed sequence of headers not commonly received from legitimate browsing traffic. But with Drive's move toward randomization, the days of that technique's effectiveness were numbered.[1]

Another approach to DDoS mitigation took advantage of the simplicity of DDoS malwares' attack code. A common form of DDoS attack is to simply send a flood of HTTP GET requests for the website the attacker wishes to bring down. If these requests look similar enough to ordinary requests from a person using a web browser, they can be challenging to detect and filter. But most DDoS malware lacks the complex processing logic that a full web browser has. So one way to partially mitigate an attack is to require the malware to perform some type of response processing.

Specifically, some anti-DDoS hosting providers made configurations something like the following: The domain name for a website would not resolve to the IP address of the server hosting the website but instead would resolve to the IP address of an initial gateway server. Upon receiving an initial HTTP GET request, this gateway server would respond with one or more redirections. The client would then have to follow these redirections in order to finally reach the actual website server. The redirection might be further complicated by requiring the client to also set a cookie which must be returned with all subsequent requests, or the redirection might be embedded in HTML code instead of given via an HTTP response code.

This series of hoops to dance through presents no difficulty at all to any modern web browser – in fact to be compliant with the Internet standards defining the protocols in use, any web browser *must* be able to accommodate them – but most DDoS malware cannot manage them. A browser is orders of magnitude more complex than most malware, and so the simple techniques employed by malware are insufficient to perform full browser logic.

Therefore, faced with the redirection, the DDoS malware would continue to send its flood of requests to the gateway server instead of finally getting them to reach the target web server that it wants to take down. Since the gateway server only has to serve very simple, relatively static redirection responses, it can absorb a much higher request rate than the web server behind it, which likely would have to do much more computationally expensive logic. And as the anti-DDoS hosting providers using this solution were generally the ones running the gateway server(s), they could provide enough hardware and bandwidth to their gateways to absorb all but the most severe DDoS attacks.

[1]This is not to say that Drive traffic cannot be identified by any form of pattern matching. In fact, the very use of randomization can be a weakness, as ordinary traffic has header values that follow a statistical distribution other than the uniform distribution used in randomization. For example, a randomized domain name will have a much higher distribution of 'q's and 'x's than a typical English domain name would.

2.3 Counter-response: The -smart attack

As is the way in the eternal game of digital cat-and-mouse between security researchers and attackers, this development did not go unanswered. Roughly two months after Drive was publicly reported, a new version of Drive was reported with a new attack command named `-smart` designed to follow redirections [9]. The new variant also introduced three other new attack types and a revision to the malware's Command & Control (C&C) communication protocol.

While the Drive author(s) didn't implement the full logic that would be in a modern web browser, they did accept a good deal of added complexity. Drive will follow `Location` headers from HTTP 3xx responses as well as scanning the HTTP response body for `meta http-equiv="refresh"` HTML tags or `location` or `document.location.href` assignments. It can also set cookies and return them in its future requests. In short, there was enough processing in this version of Drive to circumvent hosting providers' redirection mitigation technique.

The discovery of Drive made quite a splash in the security community. The very fact that DDoS malware authors were now attempting to bypass mitigation techniques was enough to get the story picked up by several news outlets [7, 11, 15, 18]. While Dirt Jumper had been advertised as having anti-DDoS mitigation capability since version 3, this appears to have been one of the first complex, successful attempts at actually pulling it off.[2]

3. BREAKING DIRT JUMPER DRIVE

While the `-smart` logic enabled Drive to circumvent certain anti-DDoS provider methods, it introduced a new weakness: greater complexity. Malware is software, and – like any other piece of software – new code introduces the chance for new bugs, and the more complex the code becomes the more likely bugs will remain initially undiscovered.

The complexity of the `-smart` attack's processing code presented itself to us as an opportunity for bugs to arise, some of which may be exploitable. Proceeding to conduct analysis, we were not disappointed: There is in fact an exploitable flaw in `-smart`'s logic, and a rather significant one at that.

Throughout the rest of this paper when mentioning the malware we reverse engineered, we will be referencing an executable with the following hashes:

```
SHA-256   b4eb5dd9c760472ed636beb53e913411f2999b95b40c8903d591e808e67f4196
SHA-1     472c098383cd47f60f9bc71b084e9adc7a07289a
MD5       63a3fbba0070f9ff56ea83dc76ef9e9e
```

[2]There was an earlier type-5 "anti-DDoS" attack type in Dirt Jumper version 5 which may have briefly been effective [2], yet reports from ASERT researchers concluded that they were unable to get the feature to function [20].

```
HTTP/1.1 301\r\n
Content-Length: 0\r\n
Location: http://example.com/first-example\r\n
\r\n
```

Figure 1: Example HTTP redirection with a total length $\leq$ 100 bytes

3.1 The -smart response-parsing logic oddity

The logic for a `-smart` attack involves a loop around a single large procedure which is responsible for forming and sending HTTP GET requests. This procedure will send a GET request and parse the corresponding response from the server. If the response involves a redirection – either via a `Location` header or in the response body – it will parse out the new redirection URL and assemble a new GET request for that address. When the loop causes the procedure to be called again, it will issue the GET request to the new URL. In this way, a `-smart` attack can follow an arbitrary number of redirects.

Of particular interest is how the HTTP response is read from the network. Figure 2 contains an annotated assembly listing from the main `-smart` procedure. Here one can see that the `recv` call will be repeated up to 10 times if it returns successfully but yields less than 100 bytes, and a one second wait will be made between attempts. As soon as the `recv` call either returns more than 100 bytes or returns an error, control flow will exit the block. Furthermore – and not visible in the listing – if 100 bytes cannot be read from the network, the entire procedure will abort (though it will promptly be reinvoked by the greater `-smart` loop and try making a new request).

This logic seems reasonable if one were writing innocent client software; sometimes servers or networks fail, so wait-and-retry-N-times is a common pattern to enhance client reliability. Furthermore, the 100-byte threshold makes sense if the HTTP response were fragmented between two IP packets which arrive with a pause between them; the client wants to wait until the whole response has arrived.

However, for a DDoS bot this logic presents a weakness. The primary purpose of a DDoS bot is to generate as much load on the target server as possible, which in this scenario translates into sending as many HTTP requests as possible in the shortest possible time window. Yet given the right conditions, this code *waits*, the very opposite from what a DDoS bot should be doing. And even worse – or better, from our perspective – the target server can influence the bot's code control flow through the response it sends the bot!

3.2 The tarpit technique

At this point a simple question arises: What would happen if a targeted server were to send a response of 100 bytes or less to a Drive DDoS bot performing a `-smart` attack? The answer is twofold:

1. The bot will retry 10 times, waiting a total of 10 seconds.
2. The `-smart` main procedure will abort further processing on the response, which *completely disables* all ability to follow redirects!

The net result is that by sending responses no larger than 100 bytes we can not only prevent a Drive DDoS bot from ever following the redirects to reach and DDoS its intended target, but we can also tarpit the bot, i.e., drastically reduce its request rate from a flood to a trickle.

This raises the question of whether a gateway server can provide a useful HTTP response in 100 bytes or less that is processed correctly by browsers. Thankfully, the answer is yes. Figure 1 shows an example HTTP redirection to `http://example.com/first-example` with a size not greater than 100 bytes.[3] This is an *almost* strictly legal HTTP response. We must say "almost" because technically any server with a reasonably accurate and functioning clock is required to send a `Date` header in this case [1, §14.18]. However, servers which do not have such clocks *must not* send a `Date` header, so an HTTP client (such as a web browser) is required to be able to process a response without one. Therefore, though our example response is not technically valid for our gateway server to send,[4] it *is* within the set of responses that any standards-compliant browser must be able to process. So we bend the rules a little. In practice, our tests showed that modern browsers handled these responses without any difficulty. Therefore, we have a technique which can be universally applied by a redirecting gateway without impacting legitimate browser traffic yet while still debilitating Drive DDoS bots.[5]

3.3 Results

We applied our tarpit technique in a controlled environment to test its efficacy. Taking the live Dirt Jumper Drive malware sample we had analyzed, we infected a host machine with the malware. We then deceived the malware to communicate with one of our own servers instead of its true C&C server and instructed the malware to launch a 100-thread `-smart` attack against a URL hosted by a web application running on another one of our servers. We tested this scenario with a gateway redirection server, first with ordinary responses greater than 100 bytes and then with responses no greater than 100 bytes.

In the first case in which ordinary responses without length constraints were issued by the gateway, the Drive bot's `-smart` attack successfully followed the gateway's redirection and proceeded to pummel the web application server; the average request rate from the bot was 610 HTTP requests/second.

[3]The literal `\r\n` sequences in the example represent a 2-byte carriage return (ASCII 13) and line feed (ASCII 10) sequence.

[4]If you're a real stickler for the rules, you could make the response valid by simply rendering the gateway server's clock horridly inaccurate. However, while adhering to the highest letter of the law, that would be quite silly.

[5]For cases where the redirection URL is long enough to bring the entire response over 100 bytes in length, an internal URL shortening service could be employed to bring the response size back within limits.

```
recv_victim_http_resp_loop:
                push    2               ; recv flags (MSG_PEEK)
                push    4096            ; length
                lea     eax, [ebp+victim_resp]
                push    eax             ; buffer
                push    esi             ; socket
                call    recv            ; attempt to receive up to 4096 bytes
                mov     edi, eax
                cmp     edi, 100        ; if more than 100 bytes, we're done
                jg      short recv_victim_http_resp_done
                test    edi, edi        ; if recv error, then we're done
                jl      short recv_victim_http_resp_done
                push    1000            ; 1 second in milliseconds
                call    Sleep           ; sleep for one second
                inc     ebx
                cmp     ebx, 10         ; try this loop body up to 10 times
                jl      short recv_victim_http_resp_loop

recv_victim_http_resp_done:
```

Figure 2: Code for receiving a victim host's response

In the second case in which the gateway issued redirections of no more than 100 bytes in length, the Drive bot's `-smart` attack failed to follow the redirection; all requests were sent solely to the gateway web server, and the web application server was left completely untouched. Furthermore, the average request rate of the attack was only 4.2 HTTP requests/second. This represents a **145× slowdown** of the attack!

In both cases, an ordinary web browser could still follow the redirection and successfully load the web application page served behind the redirection gateway.

We therefore verified that by issuing redirection responses of 100 bytes or less we could completely disable the redirection-following ability of Dirt Jumper Drive's `-smart` attack and tarpit the bot, causing it to emit only a small fraction of the requests/second that it normally would. Even better, all this can be accomplished without impacting the experience of ordinary users on modern web browsers.

So in an ironic twist, the complexity introduced by Drive's `-smart` attack ultimately led to an even worse situation for the malware than it started with!

4. CONCLUDING REMARKS

Throughout Dirt Jumper's evolution, it is interesting to observe the back-and-forth nature of the struggle between the malware authors & operators and the defenders. A stronger attack motivates the creation of a stronger defensive tactic which in turns motivates an even stronger attack. This leads to increasing complexity for both sides.

What we found particularly interesting in our research was that the increasing complexity on the side of the malware authors opened up a second front in the exchange. In addition to defense by preventing or mitigating attacks, a new form of defense is now possible: exploiting weaknesses arising from complexity in an attack. As a concrete example, our research demonstrated that the more complex logic of the `-smart` attack exposed a weakness that could render the attack inoperative and damage the bot's DDoS attack rate. At a more abstract level, this is simply the defensive exploitation of complexity in attack logic or structure, and this abstract concept extends to any part of the attacker's system, not merely to malware.

While this type of defensive exploitation raises a number of very challenging questions, it also presents a potential set of defensive opportunities, a set which will only grow larger as attackers' tools and systems become increasingly more sophisticated and complex. We believe exploring these questions, challenges, and opportunities will provide interesting avenues for further research.

5. ACKNOWLEDGMENTS

The authors would particularly like to express gratitude for the excellent in-depth analysis of the Arbor Networks Security & Engineering Response Team on the changing face of Dirt Jumper [19, 20, 12, 10, 9]. Their work provided a solid foundation on which ours could build.

6. REFERENCES

[1] Hypertext transfer protocol – HTTP/1.1. RFC 2616, 1999.

[2] Prolexic issues global warning on DDoS blackmail attempts targeting online gambling sites. `http://www.prolexic.com/company/news-events/prolexic-issues-global-warning-about-recent-ddos-blackmail-attempts-targeting-online-gambling-sites.html`, April 2012.

[3] CryptoMe DDoS extortion letter, June 2013. `http://cryptome.org/2013/06/cryptome-opsecure-extort.pdf`.

[4] Cyber-attacks: Computer says no. *The Economist*, June 2013. `http://www.economist.com/news/international/21579816-denial-service-attacks-over-internet-are-growing-easier-and-more-powerful-their`.

[5] Andre M. DiMino and Mila Parkour. Dirt Jumper DDoS bot – new versions, new targets. Technical report, DeepEnd Research, October 2011. http://www.deependresearch.org/2011/10/dirt-jumper-ddos-bot-new-versions-new.html.

[6] Jeff Edwards. Reversing the Wrath of Khan. Technical report, Arbor Networks, March 2012. http://www.arbornetworks.com/asert/wp-content/uploads/2012/03/Wrath-of-Khan11.pdf.

[7] Kelly Jackson Higgins. DDoS botnet now can detect denial-of-service defenses. *Dark Reading*, August 2013. http://www.darkreading.com/attacks-breaches/ddos-botnet-now-can-detect-denial-of-ser/240160466.

[8] Neustar Insights. Hope is not a strategy: 2012 annual DDoS attack and impact survey. Technical report, Neustar, 2013. http://www.neustar.biz/enterprise/docs/whitepapers/ddos-protection/2012-ddos-attacks-report.pdf.

[9] Jason Jones. DirtJumper Drive shifts into a new gear. Technical report, Arbor Networks, August 2013. http://www.arbornetworks.com/asert/2013/08/dirtjumper-drive-shifts-into-a-new-gear/.

[10] Jason Jones. DirtJumper's DDoS engine gets a tune-up with new "Drive" variant. Technical report, Arbor Networks, June 2013. http://www.arbornetworks.com/asert/2013/06/dirtjumpers-ddos-engine-gets-a-tune-up-with-new-drive-variant/.

[11] Michael Mimoso. Updated Drive/DirtJumper DDoS toolkit includes mitigation bypasses. *Threatpost*, August 2013. http://threatpost.com/updated-drivedirtjumper-ddos-toolkit-includes-mitigation-bypasses.

[12] Jose Nazario. Dirt Jumper DDoS bot increasingly popular. Technical report, Arbor Networks, May 2012. http://www.arbornetworks.com/asert/2012/05/dirt-jumper-ddos-bot-increasingly-popular/.

[13] Federal Bureau of Investigation, Financial Services Information Sharing and Analysis Center (FS-ISAC), and Internet Crime Complaint Center (IC3). Fraud alert – Cyber criminals targeting financial institution employee credentials to conduct wire transfer fraud, September 2012. http://www.ic3.gov/media/2012/FraudAlertFinancialInstitutionEmployeeCredentialsTargeted.pdf.

[14] Nicole Perlroth. Attacks on 6 banks frustrate customers. *The New York Times*, September 2012. http://www.nytimes.com/2012/10/01/business/cyberattacks-on-6-american-banks-frustrate-customers.html.

[15] Brian Prince. DirtJumper DDoS toolkit variant adds mitigation bypass techniques. *SecurityWeek*, August 2013. http://www.securityweek.com/dirtjumper-ddos-toolkit-variant-adds-mitigation-bypass-techniques.

[16] Arbor Special Report. Worldwide infrastructure security report. Technical report, Arbor Networks, 2012. http://www.arbornetworks.com/asert/report/ (direct link http://pages.arbornetworks.com/rs/arbor/images/WISR2012_EN.pdf).

[17] Dell SecureWorks Counter Threat Unit Research Team. 2012 threatscape report. Technical report, Dell SecureWorks, 2013. http://www.secureworks.com/assets/pdf-store/other/2012.threat.report.pdf.

[18] Robert Westervelt. Dirt Jumper DDoS toolkit gets security evasion functionality. *CRN*, August 2013. http://www.crn.com/news/security/240160429/dirt-jumper-ddos-toolkit-gets-security-evasion-functionality.htm.

[19] Curt Wilson. Dirt Jumper caught in the act. Technical report, Arbor Networks, August 2011. http://www.arbornetworks.com/asert/2011/08/dirt-jumper-caught/.

[20] Curt Wilson. A DDoS family affair: Dirt Jumper bot family continues to evolve. Technical report, Arbor Networks, April 2012. http://www.arbornetworks.com/asert/2012/04/a-ddos-family-affair-dirt-jumper-bot-family-continues-to-evolve/.