3.5 Alejandro Nolla/ Amplification DDoS attacks with game servers

3.5.1 Alejandro Nolla

- Security consultant and ethical hacking. Madrid, España.
- twitter: @z0mbiehunt3r

3.5.2 Amplification DDoS attacks with game servers

This paper describes how a DDoS amplification attack using game servers works as well as various methods to find vulnerable games and techniques to detect this kind of attack and how to try to mitigate these attacks at different levels of OSI topology as well as different levels at a network schema.

- Talk and paper can be downloaded from `http://grehack.org`

Amplification ddos attacks with games servers from the perspective of both the attacker and the defender

Alejandro Nolla Blanco – Madrid,Spain – alejandro.nolla@gmail.com

No Institute Given

Abstract

Due to the increase in DDoS attacks in recent years, and more specifically those called "amplified attacks", the author presenting this paper explains the results of his work focused on amplified DDoS attacks that leverage Online gaming servers as amplifiers.

This work explains the operation of this kind of attack, methods to identify vulnerable implementations in game servers and different mitigation approaches that vary greatly depending on the defender's point of view and the level at which remediation measures are taken.

Lastly, statistics are included to show the danger in the rise of such types of techniques, due to the growing number of vulnerable servers and the ease with which an attacker could launch an attack using thousands of amplifying servers in a few minutes.

This paper describes how a DDoS amplification attack works using game servers and the various methods to identify vulnerable games/servers. It also explains the techniques to detect and mitigate these attacks.

Keywords: ddos attacks, network security, amplification attacks

1 Introduction

Nowadays, Distributed Denial of Service Attacks, usually known as DDoS attacks, are growingly common and seek very different aims.

The goal can range from financial loss due to service disruption to the use of a DDoS as a smoke screen while other Network Assets are being attacked, usually with the intention of stealing information. DDoS are also used as a way of online extortion where the victim is asked for money to stop the attack.

There are different ways of launching a DDoS attack at different OSI levels, from a simple TCP flooding to resource exhaustion forcing SSL re-negotiations, as well as heavy query exploitation through SQL injection. Currently, the most common attacks are based on TCP/UDP traffic with forged headers and more recently amplification attacks that relay on DNS protocol to force large/heavy replies[15].

While these methods of attack are becoming more and more popular with the passing of time, there are other less known ways to achieve similar results, which luckily are not exploited so often. Among these methodologies we find DDoS attacks that use NTP protocol to force an unsolicited flood of troubleshooting information, attacks that forge SNMP requests or in the case of the attack detailed in the present paper, where Online gaming servers are used to launch amplification DDoD attacks[5].

Currently most videogames that reach the market include multiplayer support and increasingly rely the whole gaming experience on multiplayer action. This feature needs an engine to allow the integration of player's models, actions and achievements within a specific MAP.

Due to the real-time nature of multiplayer gaming, communications latency is an effect that must be minimized as much as possible, increasing the importance of gameplay data speed over re-transmission of information lost in transit. These necessities point at a protocol like UDP as the de-facto standard for the transport layer.

The use of a connectionless protocol like UDP brings several advantages, mainly related to network performance, but also carries certain disadvantages like delegating the verification of session integrity to the upper OSI layers. If upper layers don't properly check that data received belongs to the current session, someone else's machine without an established session could interfere with current sessions and, for example, force the server hosting the game to send unsolicited information to an arbitrary host.

2 How the attack works

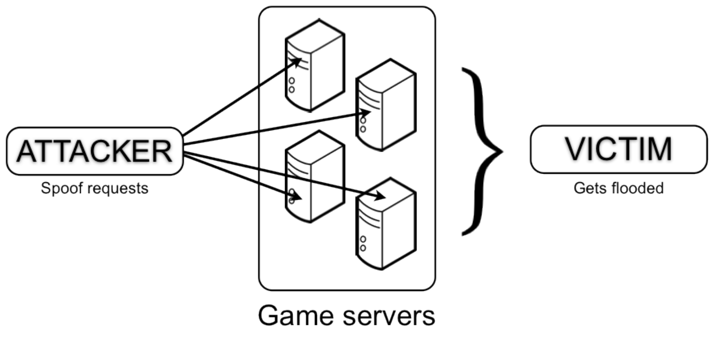

Fig. 1. Amplification DDoS attack with game servers

During gameplay, gamers are constantly interchanging information with the server that hosts the game to maintain, for example, game statistics. Depending on how the network library is implemented on the server, an attacker could exploit the mentioned behavior and force the server to send unsolicited information to a third party.

Thanks to the fact that, generally, information requests to the server only need a few bytes sent but in turn generate much larger replies, this technique is an ideal candidate for amplified DDoS attacks.

```
$ tshark -r udp_quake3_reflected_clean.pcap.cloaked
1 0.000000 192.168.1.39 -> 128.66.0.59 QUAKE3 56 Connectionless Client to Server
2 0.213635 128.66.0.59 -> 192.168.1.39 QUAKE3 1373 Connectionless Server to Client
```

Fig. 2. Info about Quake3 protocol queries and responses.

The above is an example of an amplification factor of nearly 24,5 times using a Quake 3 server.

3 Identifying vulnerable implementations

To try and distinguish if a multiplayer game is vulnerable to this type of attack there are initially three different approaches:

- Sniffing traffic during gameplay with the intention of reproducing the same behavior later.
- Using `Fuzzing` techniques at network level against the hosting server to try and force a stimulus/(non-standard reply).
- Analyzing the source code of the network implementation used by the server, if available.

The first option, capturing legitimate traffic and re-injecting after modifying it, is probably the simplest and quickest way to identify potentially vulnerable implementations, as many games use clear text commands in the communication between client and server.

```
>>> hexdump(sniffed_query['Raw'])
0000 FF FF FF FF 54 53 6F 75 72 63 65 20 45 6E 67 69 ....TSource Engi
0010 6E 65 20 51 75 65 72 79 00                      ne Query.
```

Fig. 3. Example of plain-text based Source protocol

The second option, using `fuzzing` against the hosting server, has been useful in lab1 tests to identify requests that, although being smaller than the original, resulted in specific standard sized replies or even unusually large replies[12].

The third approach used, source code review, resulted useful with Quake 3 and other similar implementations thanks to Id Software publishing the source code of Quake 3[2] which is also used by `ioquake`[3], a community maintained port based on the same implementation. Thanks to the existence of those source code repositories, white box analysis techniques were used instead of just figuring things out from the behavior showed during tests.

4 Results obtained

Apart from the results included in this section, different server side network implementations where found to misbehave, allowing the attacker to force the same UDP reply to be sent hundreds of times, observing situations where more than 800 replies were sent for a single request, therefore reaching an amplification factor of more than 800[6].

The following is a list of affected games and results obtained:

Game	Request size	Response size (min.)	Response size (max.)	Response size (avg.)	Amplification factor (avg.)
Counter-Strike 1.6	67 bytes	748 bytes	1174 bytes	961,3 bytes	x14,34
Quake 3	55 bytes	941 bytes	1566 bytes	1329 bytes	x24,16
Left4Dead 2	67 bytes	181 bytes	232 bytes	205,9 bytes	x3.07
Half-Life 1	67 bytes	149 bytes	891 bytes	562 bytes	x8,38
Counter-Strike Source	67 bytes	215 bytes	329 bytes	252,3 bytes	X3,76
Call Of Duty 4	53 bytes	1230 bytes	1566 bytes	1448,4 bytes	X27,32
Counter-Strike Global Offensive	67 bytes	191 bytes	335 bytes	238,7 bytes	X3,56

Fig. 4. Results obtained during our research

To compile the above list, sites like www.gametracker.com where checked to identify servers with biggest number of concurrent users, 10 servers per game.

For each of the 10 servers analyzed for every game, only one UDP request was sent and the reply quantified, including heading information in the packet.

During tests 3 different payloads where used, each corresponding to game engine[1]:

- Source: "\xff\xff\xff\xffTSource Engine Query\x00"
- id Tech3: "\xff\xff\xff\xffgetstatus"
- IW engine: "\xff\xff\xff\xffgetinfo"

In a few implementations of these game engines it was possible to get the same reply using only a portion of the payloads above.

As can be seen in the table, replies bigger than Ethernet's MTU (1500 bytes) were obtained and meaning that the frame would have to be fragmented further, utilizing even more resources.

It's important to stress the so-called "backscatter" effect, which happens when the victim receives an unsolicited request for the spoofed query and processes it as usual. During tests, behaviors have been seen where the answer to unsolicited UDP replies consisted in "Port unreachable" ICMP messages plus a portion of server response, therefore adding more network traffic to victim's interface.

5 Potential impact

With the intention of evaluating the viability of this attack technique in the Internet, different statistics were gathered regarding servers hosting widespread games or games with features that would make them ideal for DDoS amplification attacks.

To complete such task, two web pages were identified that would report the biggest number of servers per game, which in fact are reference sites for most "hardcorer" gamers: www.game-monitor.com and www.gametracker.com.

Game	**game-monitor.com**	**gametracker.com**
Counter-Strike 1.6	18.810	26,750
Team Fortress 2	7515	10,767
Quake 3	360	1,115
Left 4 Dead 2	3120	1,392
Half-Life 1	510	216
Counter-Strike Source	6600	12.180
Call Of Duty 4	4035	4,598
Counter-Strike Global Offensive	6975	15,060

Fig. 5. Publicly listed game servers

As can be observed in the above table, at any given time, the number of servers vulnerable to this type of attack is quite high, and therefore a potential attacker could obtain a list of thousands of servers ready to be used in a few minutes just by parsing the two server lists mentioned.

6 Mitigating the attack

Mitigation at application layer

In order to mitigate the attack at application level and try and avoid the servers from being used as attack amplifiers, developers would commonly implement the following protection measures:

– Limiting the number of requests an IP Address can make per second[9].
– Checking the request source IP Address against current gamers'IPs.
– Implementing a challenge/response verification scheme using tokens.

Based on the results obtained analyzing the first two mitigation methods, both of them can be considered insufficient protection due to the following reasons:

– If the number of requests per IP has been limited, an attacker could decrease the number of requests per server but use more servers concurrently to reach the same attack bandwidth.
– In case valid IP's were limited to those of active players, the attack remains useful against one or more players, taking advantage of the asymmetry between server and domestic Internet connections, an advantage that can be used in game contests as it has already a few times.

On the other hand, if tokens are used for session control, special care must be taken to protect the process against predictability or re-injection attacks, apart from restricting the use of a specific token to the IP Address that requested it.

Mitigation at network layer

The different approaches to mitigate this attack at network level heavily depend on the role played at infrastructure level regarding the attack[16].

Generally we can define three types of defensive role or attitude regarding this kind of attack:

– ISP/Owner of network under attack.
– Owner of server under attack.
– Owner of the game server under attack.

Defense from a Network Owner perspective Mitigation strategies against DDoS attacks amplified by game servers will depend on the type of ISP, the relationship between peers and the network infrastructure deployed. Also in most cases the work between upstream peers will be key for proper attack mitigation[8].

If possible, it is recommended that traffic is filtered at the network edge by using specific rules that include source and destination IP addresses, while accounting for the performance penalty associated with increasing the rule base[14].

As a last resort, using BGP techniques can be handy, either by using RTBH (Remotely-Triggered Black Hole) routing/filtering or modifying advertised routes to redirect traffic to a filtering farm/gateway, so it doesn't reach the victim[13].

On each case, the pros and cons of the mitigation strategy chosen to manage backscatter traffic must be balanced, considering the amount of traffic and the performance degradation introduced by filtering.

Defense from the perspective of an attacked Server Owner In this scenario the number of possible mitigation actions can be very scarce, with firewalling traffic from amplifying game servers being the only solution in most cases.

This approach is only valid in case the bandwidth available in the attacked server is not overwhelmed by the aggregated traffic provoked by the attack plus legitimate traffic, otherwise ISP intervention will be necessary.

Defense from the perspective of the Game Server Owner For the proper/correct mitigation of this attack, controls should be placed at the Application layer by establishing a communications protocol between client and server that prevents the use of the hosting server as an amplifier. This could be achieved by implementing a challenge/response scheme and tokens as explained before in this paper.

In case this kind of protection can't be established, flooding traffic could be stopped through firewall rules that perform Traffic Rate Limiting. This technique effectively creates a threshold of requests per second above which all traffic from a specific IP will be discarded.

To properly analyze traffic to identify the actual DDoS source IPs, a specific approach must be taken, and such an approach will greatly depend on the infrastructure under attack and the OSI level at which the mitigation will be most effective.

For example, say we want to mitigate an amplified DDoS attack that uses only Quake 3 servers. In this scenario, UDP datagram replies sent by the game server most probably contain "StatusResponse" at the beginning of the payload[11], but this depends of factors like game engine version or different security measures implemented to limit communication rate.

```
$ tshark -r udp_quake3.pcap.cloaked -R 'udp.srcport == 27960' -x
 33   0.002043  128.66.0.32 -> 128.66.7.9   QUAKE3 60 Connectionless Unknown
 0000  76 4b 1e 19 4c 7a c3 bf 4a 3e 59 3c 08 00 45 00   vK..Lz..J>Y<..E.
 0010  00 2a 00 00 40 00 3e 11 35 16 80 42 00 20 80 42   .*..@.>.5..B. .B
 0020  07 09 6d 38 be c3 00 16 39 5c ff ff ff ff 64 69   ..m8....9\....di
 0030  73 63 6f 6e 6e 65 63 74 00 00 00 00               sconnect....

 1924   0.112073 128.66.169.222 -> 128.66.7.9   QUAKE3 896 Connectionless Server
 to Client
 0000  76 4b 1e 19 4c 7a c3 bf 4a 3e 59 3c 08 00 45 00   vK..Lz..J>Y<..E.
 0010  03 72 00 00 40 00 76 11 50 0f 80 42 a9 de 80 42   .r..@.v.P..B...B
 0020  07 09 6d 38 82 d9 03 5e c4 cc ff ff ff ff 73 74   ..m8...^......st
 0030  61 74 75 73 52 65 73 70 6f 6e 73 65 0a 5c 73 76   atusResponse.\sv
 0040  5f 61 6c 6c 6f 77 64 6f 77 6e 6c 6f 61 64 5c 30   _allowdownload\0
 0050  5c 67 5f 6d 61 74 63 68 6d 6f 64 65 5c 30 5c 67   \g_matchmode\0\g
 0060  5f 67 61 6d 65 74 79 70 65 5c 30 5c 73 76 5f 6d   _gametype\0\sv_m
 0070  61 78 63 6c 69 65 6e 74 73 5c 31 36 5c 73 76 5f   axclients\16\sv_
 [..]
```

Fig. 6. Example of UDP received under DDoS attack using Quake3 servers

Once the protocols used and their inner workings have been identified, a traffic fingerprint must be identified to distinguish legitimate traffic from malicious traffic[10][4]. By using this fingerprint to analyze network traffic, either from a previous capture or a live capture using a SPAN port, IP addresses from servers used as amplifiers can be detected and blocked at network level.

As proof of concept, Appendix I includes a small Python script that analyzes pcaps and extracts Quake 3 servers used as amplifiers in the attack and automatically configures a Cisco IOS device to block them to try and mitigate the attack.

The application's workflow is as follows:

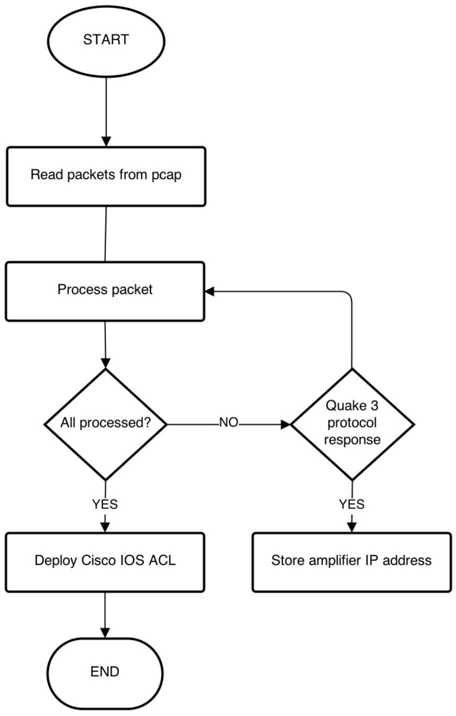

Fig. 7. Pcap processing workflow

7 Conclusions

Although DDoS attacks amplified with gaming servers usually achieve amplification factors lower than DDoS attacks amplified with DNS Open Resolvers, (x28 vs x70) this technique has certain features that can be attractive to a potential attacker:

- DDoS attacks amplified with game servers are possible thanks to insecure server-side session control the and therefore mitigation is more costly than a DDoS amplified via DNS, as DNS servers usually include secure configurations to mitigate certain DDoS attacks[7].
- Packets generated when using game servers are more common than those used to force large replies from open DNS resolvers.
- DDoS attacks amplified with game servers are not very well known, and this makes it difficult to identify and mitigate the attack.
- Due to the variety of UDP ports used by different game engines and the use of not very well known high ports, it can be misleading and difficult to identify the attack.
- Online gaming servers use high bandwidth connections to permit hundreds of concurrent users, which in turn allows for bigger attack capacity.
- It is relatively easy to obtain a list of servers vulnerable to this attack just by checking one of the many web sites that list online game servers.

For all this reasons, this attack methodology represents a serious enough menace to raise the alert level against this type of attacks. Rising awareness of the existence, identification and mitigation of these kinds of attack can become critical to development departments, system administrators or security consultants.

References

1. Wikipedia, List of game engines. `http://en.wikipedia.org/wiki/List_of_game_engines`.
2. d-Software GitHub repository. `https://github.com/id-Software`.
3. oquake GitHub repository. `https://github.com/ioquake`.
4. Richard Bejtlich. *The practice of network security monitoring : understanding incident detection and response.* No Starch Press, San Francisco, 2013.
5. CVE-1999-1066. Quake I "smurf" attack. `http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-1066`.
6. CVE-2000-0041. The "Mac DoS Attack", a Scheme for Blocking Internet Connections. `http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0041`.
7. CWE-406. Insufficient Control of Network Message Volume (Network Amplification). `http://cwe.mitre.org/data/definitions/406.html`.
8. P. Ferguson and D. Senie. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. RFC 2267 (Informational), January 1998. Obsoleted by RFC 2827.
9. Ryan C. Gordon. [cod] Query limiting message at icculus message boards. `https://icculus.org/pipermail/cod/2011-August/015397.html`.
10. Borja Merino. *Instant traffic analysis with Tshark how-to master the terminal-based version of Wireshark for dealing with network security incidents.* Packt Publishing, Birmingham, U.K, 2013.
11. Wireshark Project. Quake 3 protocol dissector. `http://anonsvn.wireshark.org/wireshark/trunk/epan/dissectors/packet-quake3.c`.
12. Michael Sutton. *Fuzzing : brute force vulnerabilty discovery.* Addison-Wesley, Upper Saddle River, NJ, 2007.

13. Cisco Systems. Remotely Triggered Black Hole Filtering Destination Based and Source Based. http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6586/ps6642/prod_white_paper0900aecd80313fac.pdf.
14. G. Raj Upadhaya. BGP Best Practices for ISPs. http://archive.apnic.net/meetings/22/docs/tut-routing-pres-bgp-bcp.pdf.
15. US-CERT. Alert TA13-088A, DNS Amplification Attacks. http://www.us-cert.gov/ncas/alerts/TA13-088A.
16. S.T. Zargar, J. Joshi, and D. Tipper. A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks. *Communications Surveys Tutorials, IEEE*, PP(99):1–24, 2013.

APPENDIX I - Quake3_DDoS_parser.py

Example of python script to parse pcap files captures while being under a DDoS amplification attack using Quake 3 servers.

```
#!/usr/bin/env python
#coding:utf-8
# Author: Alejandro Nolla - z0mbiehunt3r
# Purpose: Example for identifying Quake 3 amplifiers and block them
# with Cisco access-list
# Created: 21/06/13

import sys

try:
    from Exscript.util.interact import read_login
    from Exscript.protocols import SSH2
except ImportError:
    print 'You need exscript (https://github.com/knipknap/exscript)'
    sys.exit(-1)

import logging
# supress everything below error
logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
try:
    from scapy.all import rdpcap
except ImportError:
    print 'You need scapy (http://www.secdev.org/projects/scapy/)'
    sys.exit(-1)
#-----------------------------------------------------------------
def extract_quake3_amplifiers(pcap_file_path):

# It will classify an IP address as an amplifier if UDP payload
# consists of "....disconnect" or "....statusResponse" command
#
# @param pcap_file_path: Path to pcap file to parse
# @type pcap_file_path: str
#
# @return: Set with amplifiers servers
# @rtype: set


    amplifiers_servers = set()

# rdpcap will read all packets at once, if you need to read
# it sequentially take a look to PcapReader
# http://www.sourcecodebrowser.com/scapy/1.0.2/classscapy_1_1_pcap_reader.html

    packets = rdpcap(pcap_file_path, count=1000)

    for packet in packets:
        if not packet.haslayer('UDP'):
            continue
        if packet.haslayer('Raw'):
            raw_udp_payload = packet.getlayer('Raw')
            ip_layer = packet.getlayer('IP')
            if raw_udp_payload.load == '\xff\xff\xff\xffdisconnect' or\
               raw_udp_payload.load[0:18] == '\xff\xff\xff\xffstatusResponse':
                amplifiers_servers.add(ip_layer.src)

    return amplifiers_servers


if __name__=='__main__':
    PCAP_FILE = './udp_quake3.pcap.cloaked'
    print '''Example of Quake 3 DDoS amplification attack parser to
    automatically deploy Cisco IOS access-list - by Alejandro Nolla
    (z0mbiehunt3r)'''
    print '[*] Parsing %s' %PCAP_FILE

    amplifiers_servers = extract_quake3_amplifiers(PCAP_FILE)

    print '[+] Got %i amplifiers servers being used in the
    attack...' %len( amplifiers_servers)

    account = read_login() # read login from prompt
    conn = SSH2()
    conn.connect('192.168.1.245')
    conn.login(account)
    print conn.response
    conn.execute('config t')
    print conn.response
    # create access-list
    print '[!] Deploying access-list, take a coffee...'
    conn.execute('ip access-list extended quake3_ddos')

    for server in amplifiers_servers:
# here we directly block IP protocol but we could block UDP for Quake 3
# responses and ICMP protocol for traffic potentially being generated
# for hosts/ports unreachable and so on typical in DDoS attacks (backscatter effect)
#
# Also, we could block only ports being used in the attack (game ones, finite)

        conn.execute('deny ip host %s any' %server) # add one rule per amplifier

    # caution with implicit deny (legitimate users' traffic, routing protocols, etc)
    # and with allowing everything else
    conn.execute('permit ip any any')
    # apply access-list to interface
    conn.execute('interface fastEthernet 1/1')
    conn.execute('ip access-group quake3_ddos in')
    # quick'n dirty way for copy running-config startup-config
    conn.execute('do wr')
    print conn.response

    conn.send('exit\r')
    conn.close()

    print '[-] SLD-26 shield deployed'
```