ACIM Botnet : BotNet for Linux
Executive Summary

ACIM is a golang based DDOS botnet that was first seen on May 3rd 2021 and had zero detections on virus total. Similar variants of this malware were seen in the past and it shares a small gene pool with them and GoldMax malware.

A network of computers that is infected by malware and being controlled by a single attacking party often regarded as “bot-herder” is called a botnet and each infected system is called a bot. Botnets are used to carry various large scale attacks such as email spam, targeted intrusion, DDOS attacks etc. In this case it’s a DDOS botnet used to take down web services via http floods.

The malware offers the ability to connect to a command and control server on port 420 and then perform http post and get method based dos attacks when instructed from c2. It has no mechanisms to perform persistence or run any kind of commands on the system. The purpose of the malware is to create an army of infected systems and perform ddoos attacks on http based web services.

The malware does no damage to the system it’s infecting but it will hog bandwidth that can cause delay or downtime for other services. Also since the malware will make the infected systems participate in DDOS attacks thus it can cause ip/ip-range to be blacklisted.
ACIM botnet malware is written in golang and has an active command and control server during the time analysis. The malware is dynamically compiled and has debug symbols and function names still attached to it. It is 80% similar to the previous variants that were found and shares about 7% of its code with GoldMax malware.

The malware does not run commands or executables on the system or has functionality to create backdoors. The motive of the malware is to connect all infected systems to a command and control server and wait for further instructions in order to carry out DDOS attacks on any http based web service.

During the analysis the ip to which malware was connecting was found to be up and running although the botnet connection port i.e 420 was closed. The command and control ip belong to a domain called yumeko.sh and was located in Germany (51.2993 Lat 9.491 Long).
Detailed Static Analysis

Analysis of Malware

File identification

MD5: 6426726f24e72f95168f6dd6687c8865  
Sha1: 32e31e15fe42e4cb9e2a03698a5c7bc386311eb6  
Sha256: 496a46a07ae436b82b87bef642afbb3b06d9dbf0e0fae0199f6389f312fa4e57  
Sha512: 817c750530fd0a781181497-cad28722d9426c3b0128b83c6a513238b0c28eaded4d9a37b8938b3690d4d20aec4d03492f074998702d4c4c36fb936a72a48372  
SSDeep: 98304:hTf27Uapq5dzGq3nX0ZUhxxyLrTZ/ip1g1hUWzH7yxCcOL2:1f27Ua-pqjzGq3nX0ZiZ+Lcg12CuxCZa

File Size: 7034136 Bytes

Meta Data: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/id-linux-x86-64.so.2, Go BuildID=0wyrv52O2sayBgB4XkPC/gPM7sJzobY-coyF1H1wUQ/6cdnl68NyF100M5SKqe/7kg_y0aCyIV_oDDI5W0p, not stripped

Virus Total

https://www.virustotal.com/gui/-file/496a46a07ae436b82b87bef642afbb3b06d9dbf0e0fae0199f6389f312fa4e57/detection

Malware Bazaar

https://malshare.com/sample.php?action=detail&hash=5e1cbb6566f677da1d920c9d22f59bd7

Malicious File Summary

The malware written in golang is intended to create a botnet of infected systems that can later on carry out http based Denials Of Service attacks. The malware is dynamically linked and has not been stripped of debug symbols during compilation thus providing actual names of functions and libraries that were used.

When run, malware will first try to connect to its command and control at port 420 and confirm initial connection and then will wait for further instruction. The malware has only two DDOS functions i.e SimpleGet and SimplePost.
First the main.main function is run whose job is to connect to the command and control framework and if that fails try again. Once a successful connection is established **main.AwaitCommands** is run.

```c
1Var2 = *(long *)(in_FS_OFFSET + 0xffffffff8);
if (*(undefined *)[16])*(long *)((long *)1Var2 + 0x10) <= register0x00000020 &&
    *(undefined *)[16]! = *(undefined *)[16]!*(long *)!1Var2 + 0x10)) {
    main.Connect(param_1,param_2,param_3,1Var2,param_5,param_6,
                  (undefined *)[16])CONCAT88(in_stackfffffffffffffffcc0,in_stackfffffffffffffffcb8),
                  (undefined *)[16])CONCAT88(in_stackfffffffffffffffdd0,in_stackfffffffffffffffcc8));
    puVar1 = in_stackfffffffffffffffcc0;
    uVar3 = extraout_RDX;
    if (in_stackfffffffffffffffcc0 != 0) {
        local_18 = CONCAT88(0x72ca10,0x66dd20);
        puVar1 = local_18;
        log.Println(param_1,param_2,extraout_RDX,puVar1,param_5,param_6,(undefined *)[16])puVar1,1,1);
        main.main(param_1,param_2,extraout_RDX_01,puVar1,param_5,param_6);
        uVar3 = extraout_RDX_02;
    }
    local_18 = CONCAT88(0x72ca10,0x66dd20);
    log.Println(param_1,param_2,puVar3,puVar1,param_5,param_6,(undefined *)[16])local_18,1,1);
    main.AwaitCommands(param_1,param_2,extraout_RDX_00,puVar1,param_5,param_6,
                       in_stackfffffffffffffffcb8,in_stackfffffffffffffffcc0);
    return;
}
runtime.morestack_noctxt
    (param_1,param_2,param_3,1Var2,param_5,param_6,
     (void *)[16])CONCAT71(in_stack00000001,in_stack00000000);
main.main(param_1,param_2,extraout_RDX_03,1Var2,param_5,param_6);
return;
```

Image: main.main Function

Function call tree of the main function is as follows.

Image: Function Call Tree for main.main
The main.main function has calls to 2 important functions i.e main.Connect and mainAwaitCommands.

Image: main.Connect function is called from main.main

Image: Command and Control IP string defined

Image: ip address is pop from stack and net.Dial function is called
At the time of analysis the command and control was not listening on port 420 thus no new zombie computer could connect thus the following error message was being printed.

![Malware During Execution](image)

On successful connection to the command and control framework the malware prints a success message and then begins to run the `main.AwaitCommands`. The `main.AwaitCommands` is a controller for the malware and will take instructions from the c2 and then execute them.

![String printed on successful connection](image)

Most working of the malware is handled by `main.AwaitCommands` functions. It ensures that the connection is successful and patiently listens for incoming commands from the c2 and when a command is received it parses the command and executes the appropriate function.
**main.AwaitCommands** function starts with an infinite while loop inside which various features of malware are called. The malware offers http.get and http.post flood ddos attacks. The check routines checks if the value of puVar3 is equal to 0x8 and *p1Var2 contains the string 0x7465672d70747468 i.e (http.get). Once that is done malware proceeds to further spawn several goroutines each one calls function simpleGet and it initiates a ddos attack on the specified website.

Malware has good logging facilities and prints on screen "HTTP FLOOD | Starting for"

![Code snippet](image1)

Image : checking if the c2 command includes http.get

Exactly the same approach is followed when the http.post method is required in order to conduct the ddos attack. The only difference is this time the check is for puVar3 is equal to 0x9 and *p1Var2 contains the string http.post.

![Code snippet](image2)

Image : checking if the c2 command includes http.post
Also the malware contains a third case which if comes true, it prints "Ping ReceivedReset Content" on the screen.

```golang
if (in_stack_fffffffffffffff0 < 4) {
    local_18 = CONCAT88(0x72ca30, 0x66dd20);
    log.Println(param_1,(sigaction *)CONCAT71(uVar8,uVar7),local_18,
              &PTR_s_Ping_ReceivedReset_Content_0072ca30,(sigaction *)CONCAT71(uVar10,uVar9),
              (sigaction *)CONCAT71(uVar12,uVar11),(undefined8 *)local_18,1,1);
}
```

Further the main.AwaitCommands function is supported by other functions as shown and they do exactly what name.

The functions such as `randomDigit`, `randomString`, `randomToken` uses golang’s `rand` functions to seed the current time and then generate random integers. The implementation is trivial.

Functions such as `SimpleGet` and `SimplePost` are where actual http get/post requests are made using the golang net/http module.
Functions Making Network Connections

<table>
<thead>
<tr>
<th>Function</th>
<th>Purpose Of Connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>main.Connect()</td>
<td>Makes an Connection to the C2 Server</td>
</tr>
<tr>
<td>simpleGet()</td>
<td>Makes http get request in order to DDOS</td>
</tr>
<tr>
<td>simplePost()</td>
<td>Makes http post request in order to DDOS</td>
</tr>
</tbody>
</table>

The malware was nicely written and had good logging capabilities with limited functionalities for a ddos botnet malware. The Malware Author didn’t bother to statistically compile the binary thus the malware will not work all linux distributions also lead to it having debug symbols attached to the binary making the analysis easy. The command and control server for this botnet was hosted in Germany. We believe newer and more sophisticated variants of this malware will pop up in future and removing them will not be so trivial thus patching of linux based systems is recommended.

Indicator of Compromise

IP’s Reached Out

- 51.75.68.215

Currently Registered Domain: yumeko.sh

Domain Registration History:

<table>
<thead>
<tr>
<th>Domain</th>
<th>Last Resolved Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>alejandrojopinocc</td>
<td>2020-04-04</td>
</tr>
<tr>
<td>commonstatedevelopment.com</td>
<td>2021-06-07</td>
</tr>
<tr>
<td>ealingdistrystore.co.uk</td>
<td>2021-06-12</td>
</tr>
<tr>
<td>ealingheating.co.uk</td>
<td>2021-06-12</td>
</tr>
<tr>
<td>ealingplumbingssupplies.co.uk</td>
<td>2021-06-12</td>
</tr>
<tr>
<td>fivsfivebrand.com</td>
<td>2021-06-07</td>
</tr>
<tr>
<td>governmentre.com</td>
<td>2021-06-07</td>
</tr>
<tr>
<td>gsadeals.com</td>
<td>2021-06-07</td>
</tr>
<tr>
<td>lastchanceinteractive.com</td>
<td>2021-06-07</td>
</tr>
<tr>
<td>londongasinstallers.co.uk</td>
<td>2021-06-12</td>
</tr>
<tr>
<td>theartofluxuriating.com</td>
<td>2021-06-07</td>
</tr>
<tr>
<td>themetalblog.com</td>
<td>2021-06-07</td>
</tr>
</tbody>
</table>
GeoLocation:

ViewDNS.info > Tools > IP Location Finder

This tool will display geographic information about a supplied IP address including city, country, latitude, longitude and more.

IP Address:

IP Location Results for 51.75.68.215

City:
Zip Code: 0
Region Code:
Region Name:
Country Code: DE
Country Name: Germany
Latitude: 51.2993
Longitude: 9.491
GMT Offset:
DST Offset:

Abuse Reports

Confidence of Abuse is 85. This IP might be dangerous.

<table>
<thead>
<tr>
<th>Date</th>
<th>Comment</th>
<th>Categories</th>
</tr>
</thead>
<tbody>
<tr>
<td>2021-06-09</td>
<td>1 probe(s) @ TCP(2204)</td>
<td>Port Scan</td>
</tr>
<tr>
<td>2021-06-10</td>
<td>Blocked by Sophos UTM Network Protection / proto=6. srcport=22. dstport=20021. (2988)</td>
<td>Hacking, Brute-Force</td>
</tr>
<tr>
<td>2021-06-06</td>
<td>1 probe(s) @ TCP(49765)</td>
<td>Port Scan</td>
</tr>
<tr>
<td>2021-06-05</td>
<td>PortscanM</td>
<td>Port Scan</td>
</tr>
</tbody>
</table>

Active Ports

```
# nmap 7.91 scan initiated Tue Jun 15 12:49:56 2021 as: nmap -p -oA nmap/all_ports_51.75.68.215 -vvv 51.75.68.215
nmap scan report for yumeko.sh (51.75.68.215)
Host is up, received syn-ack (0.17s latency).
Scanned at 2021-06-15 12:49:56 IST for 601s
Not shown: 65528 closed ports
Reason: 65528 conn-refused
PORT STATE SERVICE REASON
22/tcp open  ssh  syn-ack
80/tcp open  http  syn-ack
443/tcp open  https  syn-ack
992/tcp open  telnetsyn-ack
999/tcp open  garcon  syn-ack
1194/tcp open  openvpn  syn-ack
5355/tcp open  freeciv  syn-ack
```
Mitigation Strategy and techniques

This attack is based on installing the malware on linux based systems and keep it hidden for long period of times thus for mitigations:

Specific Recommendations

Use The Following Yara Rule For Malicious Excel Document Detection

```cpp
private rule is_executable
{
  condition:
    uint16(0) == 0x5A4D or uint32(0) == 0x464c457f or uint32(0) == 0xfeedface or uint32(0) == 0xcefaedfe or uint32(0) == 0xfeedfacf or uint32(0) == 0xcfaedfe or uint32(0) == 0xcfaedfe or uint32(0) == 0xc1e8babe or uint32(0) == 0xbebafe0a
}

rule crime ZZ_botnet_aicm
{
  meta:
    description = "DDoS Golang Botnet sample for linux called 'aicm"
    sha256 = "496a46a07ae436b82b87bef642afbb3b06d9dbf0e0fae0199f6389f312fa4e57"
}
```
Strings:

$a1 = "51.75.68.215:420"

$f1 = "main.Connect" fullword
$f2 = "main.AwaitCommands" fullword
$f3 = "Methods.randomString" fullword
$f4 = "Methods.randomDigit" fullword
$f5 = "Methods.randomToken" fullword
$f6 = "Methods.SimpleGet" fullword
$f7 = "Methods.SimplePost" fullword

$b1 = "/root/bot/Methods.userAgent\00"
$b2 = "/root/bot/bot.go\00"
$b3 = "Ping Received Reset Content"
$b4 = "[BOT] | Failed to connect, Retrying"
$b5 = "HTTP FLOOD | Starting for "

// Address 0x6409c2 in 'main.AwaitCommands'
$opcodes_1 = {48 83 ?? 09 0F [3] 00 00 48 B? 68 74 74 70 2D [3] 48 39 ?? 0F }

// Address 0x640cdf in 'main.AwaitCommands'
$weak_opcodes_1 = { 48 B? 68 74 74 70 2D 70 6F 73 48 B? 68 74 74 70 2D 67 65 74 }$weak_opcodes_2 = { 48 B? 68 74 74 70 2D 67 65 74 48 B? 68 74 74 70 2D 70 6F 73 }

// Appear in 'Methods.SimplePost' and 'Methods.SimpleGet'
$constant_1 = { 80 7F B1 D7 0D 00 00 00}
$constant_2 = { 00 00 1A 3D EB 03 B2 A1}
$constant_3 = { 00 09 6E 88 F1 FF FF FF}

$ua1 = "Mozilla/5.0 (Windows NT 6.2) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1061.1 Safari/536.3"
$ua2 = "Mozilla/5.0 (Windows NT 6.2) AppleWebKit/536.6 (KHTML, like Gecko) Chrome/20.0.1090.0 Safari/536.6"
$ua3 = "Mozilla/5.0 (Windows NT 6.0) AppleWebKit/535.2 (KHTML, like Gecko) Chrome/15.0.874.120 Safari/535.2"
$ua4 = "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.6 (KHTML, like Gecko) Chrome/20.0.1092.0 Safari/536.6"
$ua5 = "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.1 (KHTML, like Gecko) Chrome/22.0.1207.1 Safari/537.1"
condition:
  is_executable and (  
    $a1 or 6 of ($f*) or 3 of ($b*) or all of ($ua*) or (  
      any of ($b*) and (  
        3 of ($f*) or  
        $opcodes_1 or  
        (for all of ($constant*): (# > 2)) or  
        10 of ($ua*)  
      )  
    ) or  
    any of ($weak_opcodes*) and (  
      2 of ($f*) and (  
        $opcodes_1 or  
        2 of ($constant*)  
      )  
    ) or  
    14 of ($ua*)  
  )  
)
Block Outbound Connections to IP Mentioned in IOCs

A firewall rule that stops any computer inside the network to communicate with the mentioned IP will stop the process of receiving commands from c2 if the malware has already compromised the network. Thus creating such firewall rules will help in stopping the malware.

Generic Recommendations

Utilize a Next-Generation Antivirus Solution for Your Enterprise

What makes Emotet malware so difficult to kill is the way that it was intended to dodge conventional record based antivirus discovery. Hence, you should introduce a more powerful piece of programming that can stay aware of and secure your association against developed dangers.

Continuously Apply Software Updates and Patches When Released

Weaknesses seem when fundamental programming gets out of date. This makes openings in the framework that can become points of passage for a wide range of awful dangers. Along these lines, introducing vital updates when they are conveyed is a significant advance in keeping up legitimate digital cleanliness in your business organization.

There are a couple of sensible practices that ought to be imparted in your association's working environment culture consistently, for example:

1. not giving log-in certifications to dubious solicitations,
2. trying not to open messages from new sources,
3. distinguishing counterfeit marking in messages,
4. checking the objective of a connection prior to opening it, also, in every case twofold checking the legitimacy of different requests.

Square File Attachments That Are Associated with Malware

Another digital cleanliness practice that you can authorize for the online wellbeing of your business is to hinder email connections containing record augmentations that are most ordinarily connected with malware, to be specific .dll and .exe. Make it one stride further by obstructing augmentations that can't be as expected filtered by your antivirus, for example, .zip. This can forestall an Emotet malware contamination, as the Trojan depends on Dynamic Link Libraries in its assaults.
Appendix of Other Malicious Associated Domains

Due to COVID-19 pandemic many cybercriminals are using malicious domains for spreading malwares, and malicious email spam to make handsome money out of it. Suspiciously some domains were identified as

- gsadeals.com
- commonstatedevelopment.com
- fivfivebrand.com
- ealingdiystore.co.uk
- ealingplumbingsupplies.co.uk
- lastchanceinteractive.co.uk
- theartofluxuriating.com
- ealingheating.co.uk
- lastchanceinteractive.com
- londongasinstallers.co.uk
- governmentre.com
- themetalblog.com
- thelocalhardwareshop.co.uk
References

1. https://analyze.intezer.com/-files/496a46a07ae436b82b87bef642afbb3b06d9dbf0e0fae0199f6389f312fa4e57

2. https://twitter.com/IntezerLabs/status/1401869234511175683

3. https://bazaar.abuse.ch/sample/496a46a07ae436b82b87bef642afbb3b06d9dbf0e0fae0199f6389f312fa4e57/

4. https://tria.ge/210607-hjyzz3h5ej

Revision Table:

<table>
<thead>
<tr>
<th>VERSION</th>
<th>DESCRIPTION</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Initial Public Release</td>
<td>Final</td>
</tr>
</tbody>
</table>

Issued by

Research wing: CyberPeace Foundation

Research Wing, Autobot Infosec Private Ltd.