Hewlett Packard Enterprise Intelligent Management Center dbman Opcode 10008 Command Injection Remote Code Execution Vulnerability

Vulnerability Data:

CVE	CVE-2017-5816
CVSS#2	10, (AV:N/AC:L/Au:N/C:C/I:C/A:C)
ZDI ID	ZDI-17-340

Table of Contents:

Overview: **2**
Details: **3**
References: **6**

Author: Chris Lyne

Overview:

Hewlett Packard Enterprise Intelligent Management Center (iMC) is affected by a command injection vulnerability in the “dbman” executable. The dbman service does not sanitize user-supplied input prior to passing it to the system() function as an argument. By sending a specially crafted dbman opcode 10008 message containing a serialized ASN.1 object, an unauthenticated attacker may execute arbitrary commands remotely with SYSTEM privileges.

The following tools/software were required for research:

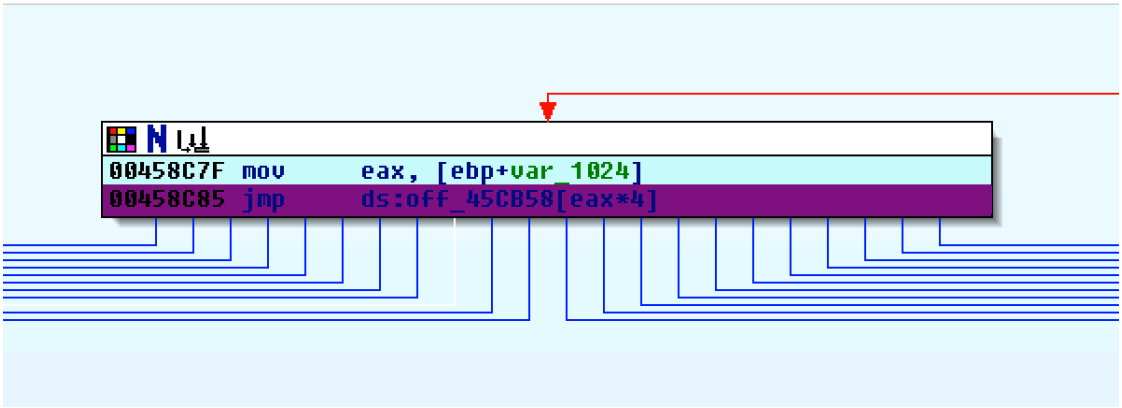
- IDA Pro (Free)
- Java Decompiler (JD) Eclipse plugin
- Python 2.7

Details:

Dbman communications occur over TCP port 2810 by default; a client/server architecture is employed. A specific message format is utilized to facilitate communication. Messages consist of an integer opcode and a corresponding ASN.1 encoded data structure. An opcode represents an operation type to be performed by the dbman executable. Each operation requires particular parameters, which are contained in the ASN.1 data structure.

The dbman message format is quite simple. It consists of an integer opcode followed by ASN.1 encoded data.

When dbman receives a message, it first interprets the opcode and utilizes a switch statement to determine which operation to perform. This can be visualized using IDA Pro.:



If the execution flow is traced further, a new thread is spawned to perform the “RestartDB” operation. It can be concluded that opcode 10008 is used to restart the backend database. Inside the RestartDB function, the ASN.1 buffer is passed to another function which processes the buffer. I have called this function the “do_dbrestart_op”.

In order to trigger the vulnerability, an ASN.1 object containing the proper data elements must be sent. It was previously found that the “HP Deployment Monitoring Agent” application communicates with the dbman service to configure database backups and restores. This application is launched by running “deploy.jar”, which is delivered as part of the iMC software package.

Decompilation of deploy.jar, using the JD Eclipse plugin, revealed a class named “AsnDataBaseRestartPara”. It contains the following class members.

```
public byte[] dbIp;
public BigInteger iDBType;
public byte[] dbInstance;
public byte[] dbSaUserName;
public byte[] dbSaPassword;
```

```
public byte[] strOraDbIns;
```

The class also contains an “encode” method, which is defined as such:

```
public void encode(ASN1Encoder arg0) throws ASN1Exception {
        int arg1 = arg0.encodeSequence();
        arg0.encodeOctetString(this.dbIp);
        arg0.encodeInteger(this.iDBType);
        arg0.encodeOctetString(this.dbInstance);
        arg0.encodeOctetString(this.dbSaUserName);
        arg0.encodeOctetString(this.dbSaPassword);
        arg0.encodeOctetString(this.strOraDbIns);
        arg0.endOf(arg1);
}
```

Using this format, an ASN.1 object can be crafted to trigger the vulnerability. After a few debugging sessions with IDA, it was found that in the “do_dbrestart_op” function, the “dbInstance” argument value is incorporated into a command that is subsequently processed by the msvcr90.dll system() function. This is where the vulnerability exists.

```
00416B69 push    offset aNet_exeStopMss ; "net.exe stop \"MSSQL$"
00416B6E lea     eax, [ebp+var_68]
00416B71 push    eax
00416B72 call    ds:??$?HDU?$char_traits@D@std@@@V?$allocator@D@1@@std@@VA?AV?$basic_string@D
00416B78 add     esp, 0Ch
00416B7B mov     [ebp+var_100], eax
00416B81 mov     ecx, [ebp+var_100]
00416B87 mov     [ebp+var_104], ecx
00416B8D mov     byte ptr [ebp+var_4], 4
00416B91 push    offset asc_4AD3A0 ; "\""
00416B96 mov     edx, [ebp+var_104]
00416B9C push    edx
00416B9D lea     eax, [ebp+var_84]
00416BA3 push    eax
00416BA4 call    ds:??$?HDU?$char_traits@D@std@@@V?$allocator@D@1@@std@@VA?AV?$basic_string@D
00416BAA add     esp, 0Ch
00416BAD mov     [ebp+var_108], eax
00416BB3 mov     ecx, [ebp+var_108]
00416BB9 mov     [ebp+var_10C], ecx
00416BBF mov     byte ptr [ebp+var_4], 5
00416BC3 mov     edx, [ebp+var_10C]
00416BC9 push    edx
00416BCA lea     ecx, [ebp+var_30]
00416BCD call    ds:??4?$basic_string@DU?$char_traits@D@std@@@V?$allocator@D@2@@std@@QAEAAV01
00416BD3 mov     byte ptr [ebp+var_4], 4
00416BD7 lea     ecx, [ebp+var_84]
00416BDD call    ds:??1?$basic_string@DU?$char_traits@D@std@@@V?$allocator@D@2@@std@@QAE@XZ ;
00416BE3 mov     byte ptr [ebp+var_4], 3
00416BE7 lea     ecx, [ebp+var_68]
00416BEA call    ds:??1?$basic_string@DU?$char_traits@D@std@@@V?$allocator@D@2@@std@@QAE@XZ ;
00416BF0 lea     ecx, [ebp+var_30]
00416BF3 call    ds:?c_str@?$basic_string@DU?$char_traits@D@std@@@V?$allocator@D@2@@std@@QBEP
00416BF9 push    eax             ; char *
00416BFA call    ds:system
```

To demonstrate this, the screenshot below shows the command that is constructed and passed to system(). A value of “AAAA” was sent as the “dbInstance” value.

```
debug025:00976EF8 aNet_exeStopMssqlAaaa db 'net.exe stop "MSSQL$AAAA"',0
```

The string ‘net.exe stop “MSSQL$’ is prepended to the value of dbInstance and then a trailing double quote is appended. In this example, the full command to be executed by system() is ‘net.exe stop “MSSQL$AAAA”’. In order to execute arbitrary commands, a trivial case of CLI-fu is required to construct a syntactically valid command.

In this case, to demonstrate proof of concept, the following command was constructed.

```
debug025:0098E678 aNet_exeStopMssqlAWhoamiC10008_txt db 'net.exe stop "MSSQL$a"& whoami > C:\10008.txt &"',0
```

The “malicious” command “whoami > C:\10008.txt” is injected and executed. This command can be replaced with an arbitrary value. Arbitrary code execution has been achieved.

In the references section, links have been added for relevant an IDA Pro IDB file and PoC code.

References:

[1] https://github.com/lynerc/exploits/blob/master/hp/dbman.idb
[2] https://github.com/lynerc/exploits/blob/master/hp/hp_imc_7_2_10008_rce.py
[3] http://snmplabs.com/pyasn1/pyasn1/contents.html
[4] http://www.zerodayinitiative.com/advisories/ZDI-17-340/