Part 2 of 2: The Integrity Project

WebJob

By Russ McRee – ISSA member, Puget Sound (Seattle), USA chapter

Prerequisites

VMWare Server 2.0 for the WebJob VM

Similar Projects

FTK and Autopsy
RAPIER 3.2

In this, the second discussion in our two-part series on tools from the Integrity Project, we'll investigate WebJob. The Integrity Project exists to "build high quality tools that meet the needs of both incident handlers and system administrators." We'll focus on security capabilities but remember that WebJob is also quite useful in roles such as performing administrative tasks for large content delivery networks or deploying FreeBSD, Linux, Solaris, and Windows packages.

Continued close contact with the project owners ensured content contributions from Bob Austin, who provided project information on behalf of WebJob developer Klayton Monroe, and FTimes as discussed last month. WebJob is actively used by Fortune 500 and government IT production environments to address a wide range of requirements such including:

- **eDiscovery** – WebJob supports enterprise-scale eDiscovery. WebJob, in concert with tools such as FTimes, is used to automate searches.
- **Security compliance** – WebJob is configured to automatically harvest security configurations to assess security posture and patch level.
- **Enterprise intrusion detection system management**, providing a more labor-efficient way to perform IDS system administrative tasks without sacrificing the availability needed for these security-critical devices. WebJob is used to support the following:
  - System health (load, disk utilization, etc) – reported back to the central console, where it is graphed, alerts generated, etc.
  - OS patching – OS patches are first vetted, and then placed in a queue to be automatically fetched and installed by a WebJob task.
  - Since WebJob was deployed, the operating systems on all sensors have been upgraded twice, essentially amounting to complete reinstalls of the systems each time. These upgrades were done by carefully written and tested WebJob tasks, with downtime limited to about ten minutes per host.
  - Software upgrades (of IDS software) are handled through a WebJob task, again requiring no interactive effort to manage over 240 IDS systems.
  - Configuration changes (such as changing `resolv.conf` on all systems, maintaining a common `/etc/hosts` file or user account management) are all done via WebJob tasks.
  - Add-on tools – have been deployed to the IDS sensors, which fetch their configuration, report their data, and can be upgraded as needed all via WebJob tasks.

Think of WebJob as is an open source framework that provides the automation and security to support a variety of security "jobs" such as system/integrity monitoring, enterprise search, configuration management, compliance verification, automated analysis, etc.

WebJob has also been used to accomplish the following:

- Automatically harvest argus, ifconfig, lsof, netstat, ndd, patch, ps, tcpdump, (name your utility), etc. data
- Automatically update cron tabs, DNS records, password files, snort rules, web sites, (name your application), etc.
- Automatically update system binaries when their MD5s do not match expected values
- Conduct massive searches for credit card numbers, social security numbers, and suspect hashes
- Harvest system information to perform security audits or compliance verification
- Implement a virtual evidence locker (VEL)
- Implement/maintain a distributed malware test harness
- Perform integrity monitoring with FTimes

The architecture is best described visually as shown in Figure 1 on the next page.

WebJob was initially written to assist incident response handlers by providing an efficient way to import and run known good diagnostic tools when investigating live systems, but has since evolved to perform a wide range of functions. Future plans for WebJob include:

- Add a cross-platform client-side scheduler
- Refine and enhance server-side capabilities (job queues/tracking, pipeline processing, and management)

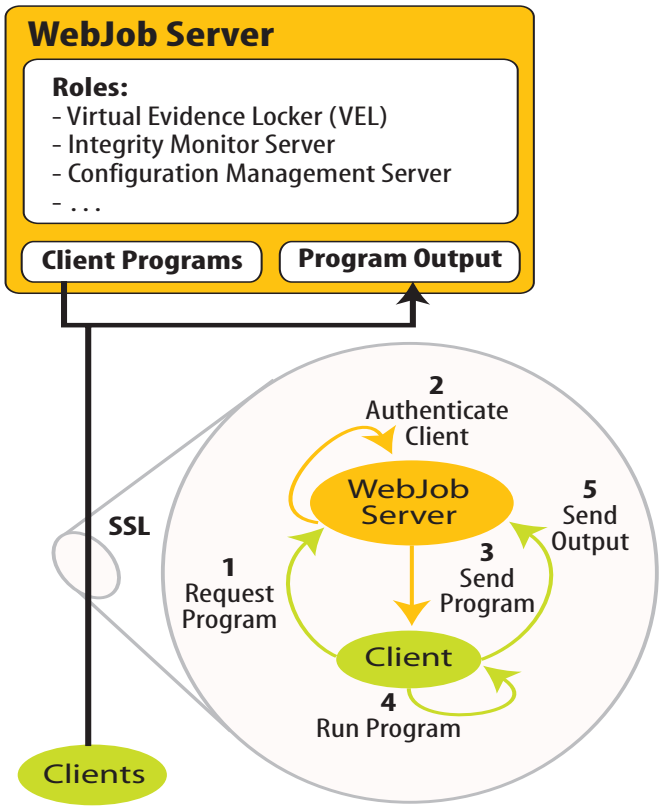

Figure 1 - WebJob Client-Server Architecture

- Refine and enhance client-side capabilities (embedded scripting, caching, and per-user configurations)
- Enhance and extend client-server communications and protocols.
- Add GUI support (e.g., dashboard, job submissions, etc.).

Using WebJob

Installation

Installing WebJob is as easy as downloading `webjob-1.8.0.tgz` and executing `tar zxvf webjob-1.8.0.tgz`. Change directories to the uncompressed WebJob install package, follow with the typical `./configure`, `make`, `sudo make install` and you're finished.

The single most important next step is committing to a thorough read of README.INSTALL, found in the root of the directory the WebJob tarball unzipped itself into.

There are significant configuration steps to take; ideally, consider the use of the pre-configured webjob-demo-appliance-freebsd-7.0[1] VM image as I did.

Like its FTimes partner, the best way to get started is to choose an entry from the WebJob cookbook,[2] which contains a series of "recipes" including details on compiling, testing, reporting, administration, management, collection, monitoring, compliance testing, patch analysis, and more. Each recipe attempts to solve a particular task or objective and is designed, where possible, to be scripted. Don't forget the man pages[3] for WebJob as well.

Understanding the inner workings

In the most common deployment, any number of participating systems run WebJob periodically (typically invoked by cron on UNIX, the Scheduler Service on Windows, etc). The WebJob client connects to a central Web server (typically Apache with SSL support enabled), and downloads a task or set of tasks. WebJob supports both client and server certificates. Using FTimes as an example, the "task list" might simply be to run FTimes and report back the map output. Often, there is one hourly job which records system-uptime stats and runs FTimes on critical system files, and then a daily job which kicks off regularly scheduled maintenance, rotates log files, does a full FTimes run, etc.

The job scripts and binaries typically reside on the WebJob server, and the participating systems automatically download each as they run them. This allows management of hundreds of servers, and consolidated lists indicating scenarios such as "every participating system runs job A, database servers run job B, application servers run job C, all machines with OS revision N run job D," etc. When a job needs to be changed, it needs to be changed only on the WebJob server; the next time any participating system runs that job, it will automatically run the updated version. Given that jobs are often written as scripts (shell, perl, etc), they can be easily tracked in a CVS repository on the WebJob server.

A large WebJob deployment can be hierarchical where participating systems connect to a WebJob server which is, itself, a client of an upper-tier WebJob server. Also, the participating systems can be configured to only execute jobs that have been digitally signed with a key that does not need to be stored on the server; the jobs can be generated on an isolated machine, signed, and then placed on the WebJob server(s). If an attacker were to compromise the WebJob server, he would not be able to take advantage of said compromise to control end clients. The attacker could not inject a malicious WebJob task, as he would not be able to generate signed job files.

The WebJob project developers indicate that when doing penetration-testing, they've seen many commercial or home-grown enterprise-management tools, and very few (if any) protect as well as WebJob does against an attacker who takes control of the central server and thus controls all the clients too.[4]

Real world scenario

One task of interest to anyone whose responsibilities include monitoring systems for security and availability is ensuring that someone or something is monitoring the monitoring systems themselves. It goes without saying that little or no

1 http://www.korelogic.com/Resources/Tools/webjob-demo-appliance-freebsd-7.0.tgz.

2 http://webjob.sourceforge.net/WebJob/Cookbook.shtml.

3 http://webjob.sourceforge.net/WebJob/Man+Pages/index.shtml.

4 Notes from Bob Austin.

Figure 2 – Confirm connectivity

time should go by where an important monitoring system goes down, and the appropriate administrator isn't made aware. WebJob offers a few readily available options such as monitoring ps, uptime, lsof, and df data. Most options can be preprocessed, loaded into RRD or MySQL, and rendered in HTML reports and graphs.

One way I prefer to monitor the monitors, aside from a typical heartbeat check, is to see who my monitors are talking to. Random high port connectivity to a Chinese IP would typically indicate "a very bad thing," while one of my monitors should only speak with known good database servers, perhaps a console, and a WebJob server. A very simple baseline should then be easy to establish for expected monitor behavior. Should it change in any way: red alert!

Following is a simplification of the "Harvest LSOF Socket Info"[5] recipe from the cookbook, as well as Example 1 from the WebJob man page.[6]

We'll assume a client running a Linux OS; my "monitor" is an Ubuntu server. All WebJob paths are in keeping with those you'll find on the VM. Even though we used the WebJob VM, WebJob needs to be installed on all clients as well. Just follow the directions above.

On the WebJob server I created a client account:

```
htpasswd /var/webjob/config/apache/ht-client
  toolsmith01
```

I then created a profile directory for toolsmith01:

```
mkdir -p -m 755 /usr/local/webjob/profiles/
  toolsmith01/commands
```

```
chown -R 0:0 /usr/local/webjob/profiles/toolsmith01
```

I also created a test script (testenv) for toolsmith01 to for use to test connectivity:

```
#!/bin/sh
echo "WEBJOB_CLIENTID=${WEBJOB_CLIENTID}"
echo "WEBJOB_HOSTNAME=${WEBJOB_HOSTNAME}"
```

It's recommended to change mode to 644 for the client profile because programs and scripts in the commands directory should not be run on the server.

```
chmod 644 /usr/local/webjob/profiles/toolsmith01/
  commands/testenv
```

On the client (my monitor server) I created a client config file for toolsmith01 (`webjob.cfg`):

```
ClientId=toolsmith01
  UrlGetUrl=https://192.168.248.104/cgi-client/nph-
   webjob.cgi
  UrlUsername=toolsmith01
  UrlPassword=testtest
  UrlAuthType=basic
  TempDirectory=/tmp
```

To test connectivity I executed `./webjob --execute --file /home/rmcree/webjob/webjob.cfg testenv` from the client (see Figure 2).

I then grabbed the netstat binary from my client and uploaded it to the WebJob server; specifically to `/var/webjob/profiles/toolsmith01/commands`. Remember, the binary you choose must be compiled for the system it's running on or you'll note errors.

On the client, I created `netstat.cfg` in the `/usr/local/webjob/etc` directory on my client:

```
ClientId=toolsmith01
URLGetURL=https://192.168.248.104/cgi-webjob/nph-
  webjob.cgi
#URLPutURL=https://192.168.248.104/cgi-webjob/nph-
  webjob.cgi
URLUsername=toolsmith01
URLPassword=testtest
URLAuthType=basic
RunType=snapshot
OverwriteExecutable=Y
UnlinkOutput=Y
UnlinkExecutable=Y
GetTimeLimit=0
RunTimeLimit=0
PutTimeLimit=0
URLDownloadLimit=10000000
TempDirectory=/usr/local/webjob/run
```

I tested functionality as follows, after I commented out the `URLPutURL` reference in the `netstat.cfg` file:

```
webjob -e -f netstat.cfg netstat --tcp
```

If all's well you'll likely see something like this:

```
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address      Foreign
  Address     State
tcp    0    0 hio-ubuntu-02.loc:35482
  emergingthreats.net:www ESTABLISHED
tcp6    1    0 localhost:8308     localhost:43091
  CLOSE_WAIT
tcp6    1    0 localhost:8308     localhost:43095
  CLOSE_WAIT
```

Finally, I ran WebJob so the client output would be uploaded to the WebJob server. To do so I uncommented the `URLPutURL` reference. The incoming directory on the WebJob server should contain four files that look approximately like:

```
toolsmith_20081208145123_netstat.env
toolsmith_20081208145123_netstat.err
toolsmith_20081208145123_netstat.out
toolsmith_20081208145123_netstat.rdy
```

The `.env` results are system and environment oriented, the `.out` file should look like our functionality test above. If the

5 http://webjob.sourceforge.net/Files/Recipes/webjob-harvest-lsof-socket-info.txt.

6 http://webjob.sourceforge.net/WebJob/Man+Pages/webjob.shtml.

WebJob run was successful the `.err` file should be empty. Note that the `.rdy` file is a file that contains server-side information about the upload, and it acts as a lock release. In other words, it indicates to other server-side tools that this job is "ready" for additional processing.

In the official webjob 1.8.0 release there is also a tool called webjob-create-profile. To create my toolsmith01 profile, I would have done the following:

```
webjob-create-profile -H /var/webjob toolsmith01
```

That, in turn, would create

```
/var/webjob/profiles/toolsmith01
   /var/webjob/profiles/toolsmith01/commands
   /var/webjob/profiles/toolsmith01/config
```

as well as a number of config files that could, in turn, be used on my client (i.e., my Ubuntu "monitor" server).

Hopefully, you get a sense of how extensive the options are, and realize that many a process can be automated with WebJob.

WebJob resources

—Basic Integrity Monitoring via WebJob (BIMVW).[7]

—All the Integrity Project tool goodness you'll ever need.[8]

—Some of WebJob's automation benefits.[9] This study makes a compelling case for WebJob's return-on-investment (ROI), and translates to almost any other common/repetitive IT task.

—Those among you who manage Snort farms may find the cookbook entry "Managing multiple Snort instances on many systems" extremely useful.[10]

In conclusion

In both our discussions regarding tools from the Integrity Project, we've barely touched on the endless uses for these tools. Again, be sure to read the cookbook and man pages for each. Both security practitioners and system administrators are well advised to consider multiple uses for WebJob.

Thanks to the Integrity Project for contributing mightily to this two-part series; I look forward to other offerings from this group in the future.

Cheers…until next month.

Acknowledgments

Bob Austin and Klayton Monroe of KoreLogic Security for significant contributions to this two-part series.

About the Author

Russ McRee, GCIH, GCFA, CISSP, is a security analyst working in the Seattle area. As an advocate of a holistic approach to information security, Russ' website is holisticinfosec.org. Contact him at russ@holisticinfosec.org.

7 http://webjob.sourceforge.net/Files/Recipes/ftimes-bimvw.txt.

8 http://www.korelogic.com/tools.html.

9 http://webjob.sourceforge.net/Files/Papers/webjob-breakeven-analysis-install-solaris-package.pdf.

10 http://webjob.sourceforge.net/Files/Recipes/webjob_manage_snort.txt.