Analysis of a Compromised System
Part 2: Dead (Offline) Analysis

License

Contents

Preparation

You cannot complete this lab (part 2) without having saved the evidence you collected during part 1 of the lab.

These tasks can be completed on the LinuxZ IMS image. To start, download the Backtrack VM using the download script. Note that within the Leeds Met environment you can choose to use a Live Disk version of Backtrack or the **Installed version (RECOMMENDED for this lab)**, which is a larger download but your changes will

persist when restarting the VM. Using the Installed version, you can choose to right click on the monitor icon in the system tray to adjust the screen size of the VM.

> Note that some students have reported difficulties when working from a live disk VM on this lab, presumably due to the amount of storage used, which a live disk OS will store in RAM. Working on evidence directly from USB storage may avoid the problem (although you would need to adjust the provided steps as necessary).

Start the Backtrack VM. If prompted, login (user:root, pass:toor). And start the graphical desktop environment: `startx`. Open a terminal window (such as start->System->Konsole).

This lab work could be completed on any system with Backtrack installed or available, and the evidence collected from the compromised VM in the previous lab.

Introduction to dead (offline) analysis

Once you have collected information from a compromised computer (as you have done in the previous lab), you can continue analysis offline. There are a number of software environments that can be used do offline analysis. We will be using Backtrack, which includes a number of forensic tools. Another popular toolset is the Helix incident response environment, which you may want to also experiment with.

This lab re-enforces what you have learned about integrity management and log analysis, and introduces a number of new concepts and tools.

Getting our data into the analysis environment

To start, we need to get the evidence that has been collected in the previous lab onto an analysis system (copy to /root/evidence) that has software for analysing the evidence. If you like, you can transfer the evidence using USB or a VM shared folder to /root/evidence on the Backtrack system.

If you are using the LinuxZ image, an SSH server is running on your lab machine and you can instead transfer the evidence straight out of /home/student to the Backtrack system using scp:

> **Note the IP addresses**: run /sbin/ifconfig, and also run ifconfig on the guest Backtrack VM. Make a note of the two IP addresses, which should be on the same subnet (starting the same).
>
> **On Backtrack**:

```
scp -r student@Your-Host-IP-Address:evidence evidence
```

Mounting the image read-only

It is possible to mount the partition image directly as a loop device, and access the files directly. However, doing so should be done with caution (and is generally a bad idea), since it may result in changes to your evidence, and you risk infecting the analysis machine with any malware on the system being analysed. However, this technique is worth exploring, since it does make accessing files particularly convenient.

Create a directory to mount our evidence onto:

```
mkdir /mnt/compromised
```

Mount the image that we previously captured of the state of the main partition on the compromised system:

```
mount -O ro -o loop evidence/sda1.img /mnt/compromised
```

Confirm that you can now see the files that were on the compromised system:

```
ls /mnt/compromised
```

Backtrack includes rkhunter, which is (as the name suggests) another rootkit detection program (similar to chkrootkit used in the previous lab). Now that we have access to the files from within Backtrack, you can easily run an offline rootkit check:

```
rkhunter --check --enable rootkits --rootdir /mnt/
compromised
```

Note that it detects many signs of rootkits, which clearly indicates this system has been compromised, although as you will see from the analysis you are about to undertake, the specific rootkits that are detected are not necessarily accurately identified.

Preparing for analysis of the integrity of files

Fortunately the “system administrator” of the Red Hat server had run a file integrity tool to generate hashes before the system was compromised. Start by fetching a copy of the hashes recorded of the system when it was in a clean state...

Ensure the Backtrack system has access to the Internet. Hint: check the VM network mode is set to bridged.

If you are in the Leeds Met labs, you need to tell wget to use our proxy:

```
export http_proxy=192.168.208.51:3128
```

Download the file:

```
wget http://old.honeynet.org/scans/scan29/linux-suspended-md5s.gz
```

Extract the file:

```
gunzip linux-suspended-md5s.gz
```

Note: if you receive an error message, check that the file is actually compressed. If not, just rename the file:

```
cp linux-suspended-md5s.gz linux-suspended-md5s
```

View the file, to confirm all went well:

```
less linux-suspended-md5s
```

(Q to quit)

As you have learned in the Integrity Management lab, this information can be used to check whether files have changed.

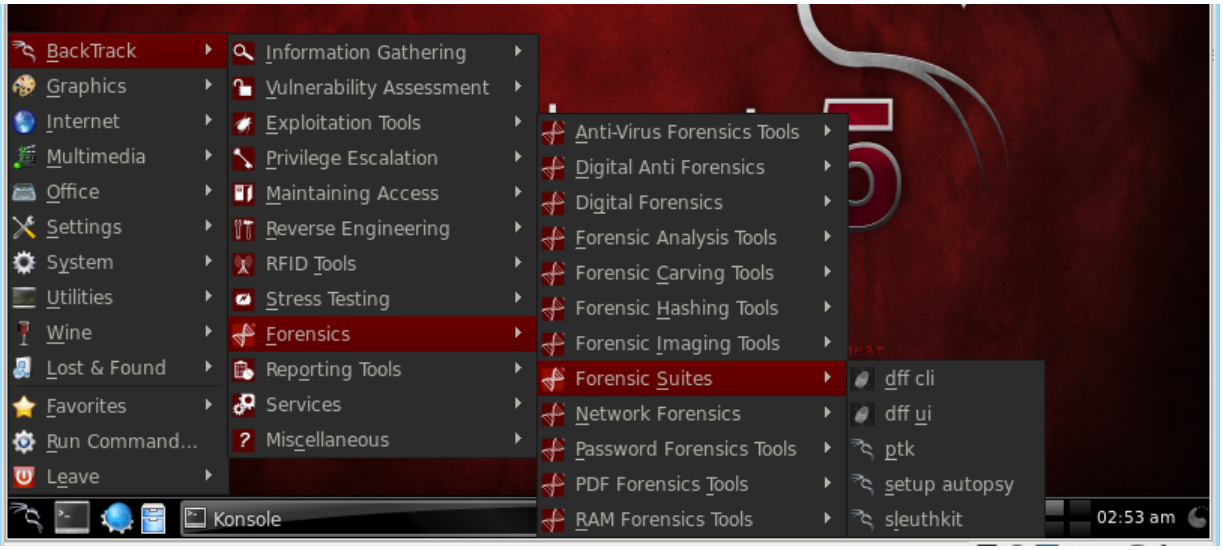
Starting Autopsy

Autopsy is a front-end for the Sleuth Kit™ (TSK) collection of forensic analysis command line tools.

Create a directory for storing the evidence files from Autopsy:

```
mkdir /root/evidence/autopsy
```

Setup Autopsy. You can do this using the program menu, as shown below, click “setup autopsy”



Answer “no” to using the NIST National Software Reference Library (NSRL).

When prompted, specify /root/evidence/autopsy as the evidence locker directory.

Once that is complete start Autopsy:

```
./autopsy
```

Open Firefox, and visit http://localhost:9999/autopsy

Click “New Case”.

Enter a case name, such as “RedHatCompromised”, and a description, such as “Compromised Linux server”, and enter your name. Click “New Case”.

Click “Add Host”.

In section “6. Path of Ignore Hash Database”, enter /root/linux-suspended-md5s

Click “Add Host”.

Click “Add Image”.

Click “Add Image File”.

For section “1. Location”, enter /root/evidence/sda1.img

For “2. Type”, select “Partition”.

For “3. Import Method”, select “Symlink”.

Click “Next”.

Click “Add”.

Click “Ok”.

File type analysis and integrity checking

Now that you have started and configured Autopsy with a new case and hash database, you can view the categories of files, while ignoring files that you know to be safe.

Click “Analyse”.

Click “File Type”.

Click “Sort Files by Type”.

Confirm that “Ignore files that are found in the Exclude Hash Database” is selected.

Click “Ok”, this analysis takes some time.

Once complete, view the “Results Summary”.

The output shows that over 16000 files have been ignored because they were found in the md5 hashes (“Hash Database Exclusions”). This is good news, since what it leaves us with are the interesting files that have changed or been created since the system was in a clean state. This includes archives, executables, and text files (amongst other categories).

Click “View Sorted Files”.

Copy the results file location as reported by Autopsy, and open the report in a new tab within Firefox:

> /root/evidence/autopsy/RedHatCompromised/host1/output/sorter-vol1/index.html

Click “compress”, to view the compressed files. You are greeted with a list of two compressed file archives, “/etc/opt/psyBNC2.3.1.tar.gz”, and “/root/sslstop.tar.gz”.

Figure out what psyBNC2.3.1.tar.gz is used for. Try using Google, to search for the file name, or part thereof. Browse the evidence (on the Backtrack system, using a file browser, such as Dolphin) and look at the contents of the archive (in /etc/opt, and you may find that the attacker has left an uncompressed version which you can assess in Autopsy). Remember, don’t execute any files from the compromised system on your analysis machine: you don’t want to end up infecting your analysis machine. For this

reason, it is safer to assess these files via Autopsy. Browse to the directory by opening another tab of Autopsy, and clicking “Analyse”, “File Analysis”, then browse to the files from there (in /etc/opt). Read the psyBNC README file, and note what this software package is used for:

Next, we investigate what sslstop.tar.gz is used for. A quick Google brings up a CGSecurity.org page, which reports that this script modifies httpd.conf to disable SSL support from Apache. Interesting... Why would an attacker would want to disable SSL support? This should soon become clear.

Return the page where “compress” was accessed (/root/evidence/autopsy/RedHatCompromised/host1/output/sorter-vol1/index.html), and click “exec”. This page lists a fairly extensive collection of new executables on our compromised server.

Make a list of all the executables that are likely trojanized. Hint: for now ignore the “relocatable” objects left from compiling the PSYBNC software, and focus on “executable” files, especially those in /bin/ and /usr/bin/.

Make a list of all the standard Linux executables on the system that are likely trojanized, as evidence that you have completed this part of the task.

Label it or save it as “DeadIR-1”.

Refer to your previously collected evidence to identify whether any of the new executables were those with open ports when live information was collected. Two of these have particularly interesting file names: “/usr/bin/smbd -D” and “/usr/bin/(swapd)”. These names are designed to be deceptive: for example, the inclusion of “-D” is designed to trick system administrators into thinking that any processes were started with the “-D” command line argument flag.

Note that /lib/.x/ contains a number of new executables, including one called “hide”. These are likely part of a rootkit.

Using Autopsy “File Analysis” mode, browse to “/lib/.x/”. **Explicit language warning: if you are easily offended, then skip this next step.** View the contents of “install.log”. This includes the lines:

```
##################################################
# SucKIT version 1.3b by Unseen <unseen@broken.org> #
##################################################
```

SuckIT is a rootkit that tampers with the Linux kernel directly via /dev/kmem, rather than the usual approach of loading a loadable kernel module (LKM). The lines in the log may indicate that the rootkit had troubles loading.

SuckIT and the rootkit technique is described in detail in Phrack issue 58, article 0x07 "Linux on-the-fly kernel patching without LKM"

http://www.textfiles.com/magazines/PHRACK/PHRACK58

Using Autopsy “File Analysis”, view the file /lib/.x/.boot

Make a note of the file’s access times, this will come in handy soon.

This shell script starts an SSH server (s/xopen), and sends an email to a specific email address to inform them that the machine is available. View the script, and determine what email address it will email the information to:

Return to the file type analysis (presumably still open in a Firefox tab), still viewing the “exec” category, also note the presence of “adore.o”. Adore is another rootkit (and worm), this one loads via an LKM.

Here is a concise description of a version of Adore:

http://lwn.net/Articles/75990/

This system is well and truly compromised, with multiple kernel rootkits installed, and various trojanized binaries.

Timeline analysis

It helps to reconstruct the timeline of events on the system, to get a better understanding. Software such as Sluthkit (either using the Autopsy frontend or the mactime command line tool) analyses the MAC times of files (that is, the most recent modification, most recent access, and most recent inode change) to reconstruct a sequence of file access events.

In another Firefox tab, visit http://localhost:9999/autopsy, click “Open Case”, “Ok”, “Ok”.

Click “File Activity Timelines”.

Click “Create Data File”.

Select “/1/ sda1.img-0-0 ext”.

Click “Ok”.

Click “Ok”.

For “5. Select the UNIX image that contains the /etc/passwd and /etc/group files”, select “sda1.img-0-0”.

Wait while a timeline is generated.

Click “Ok”.

Once analysis is complete, the timeline is presented. Note that the timeline starts with files accessed on Jan 01 1970.

Click “Summary”.

A number of files are specified as accessed on Jan 01 1970. What is special about this date?

Browse through the history. Note that it is very detailed, and it is easy to get lost (and waste time) in irrelevant detail.

The access date you previously recorded (for “/lib/.x/.boot”) was in August 2003, so this is probably a good place to start.

Browse to August 2003 on the timeline, and follow along:

Note that on the 6th of August it seems many files were accessed, and not altered (displayed as “.a..”). This is probably the point at which the md5 hashes of all the files on the system were collected.

On 9th August a number of config files were accessed including “/sbin/runlevel”, “/sbin/ipchains”, and “/bin/login”. This indicates that the system was likely rebooted at this time.

On 10th August, a number of files that have since been deleted were accessed.

Shortly thereafter the inode data was changed (displayed as "..c.") for many files. Then many files owned by the *apache* user were last modified before they were deleted. The apache user goes on to access some more files, and then a number of header files (.h) were accessed, presumably in order to compile a C program from source. Directly after, some files were created, including "/usr/lib/adore", the Adore rootkit.

At 23:30:54 /root/.bash_history and /var/log/messages were symlinked to /dev/null.

Next more header files were accessed, this time Linux kernel headers, presumably to compile a kernel module (or perhaps some code that tries to tamper with the kernel). This was followed by the creation of the SuckIT rootkit files, which we previously investigated.

Note that a number of these files are created are again owned by the "apache" user.

What does this tell you about the likely source of the compromise?

Further down, note the creation of the /root/sslstop.tar.gz file which was extracted (files created), then compiled and run. Shortly after, the Apache config file (/etc/httpd/conf/httpd.conf) was modified.

Why would an attacker, after compromising a system, want to stop SSL support in Apache?

Meanwhile the attacker has accidently created a /.bash_history, which has not been deleted.

Further down we see wget accessed and used to download the /etc/opt/psyBNC2.3.1.tar.gz file, which we investigated earlier.

This file was then extracted, and the program compiled. This involved accessing many header (.h) files. Finally, the "/etc/opt/psybnc/psybnc.conf" file is modified, presumably by the attacker, in order to configure the behaviour of the program.

Highlight the point in the timeline where you think the first suspicious activity started to happen, and take a screenshot, as evidence that you have completed this part of the task.

Label it or save it as "DeadIR-2".

Logs analysis

As you learned in the Log Management topic, the most common logging system on Unix systems is Syslog, which is typically configured in /etc/syslog.conf (or similar, such as rsyslog). Within the Autopsy File Analysis browser, navigate to this configuration file and view its contents. Note that most logging is configured to go to /var/log/messages. Some security messages are logged to /var/log/secure. Boot messages are logged to /var/log/boot.log.

Make a note of where mail messages are logged, you will use this later:

Within Autopsy, browse to /var/log. Note that you cannot view the messages file, which would have contained many helpful log entries. Click the inode number to the right (47173, as illustrated below):

l/l	messages	2003-08-10 23:30:54 (BST)	2013-04-09 16:19:06 (BST)	2003-08-10 23:30:54 (BST)	9	0	0	47173

As previously seen in the timeline, this file has been symlinked to /dev/null. If you are not familiar with /dev/null, search the Internet for an explanation.

For now, we will continue by investigating the files that are available, and later investigate deleted files.

Using Autopsy, view the /var/log/secure file, and identify any IP addresses that have attempted to log in to the system using SSH or Telnet.

Determine the country of origin for each of these connection attempts:

> On a typical Unix system we can look up this information using the command:
>
> whois *IP-address*
>
> (Where IP-address is the IP address being investigated).
>
> However, this may not be possible from within our lab environment, and alternatively there are a number of websites that be used:
>
> http://whois.domaintools.com/

http://whois.net/ip-address-lookup/

You may also run a traceroute to determine what routers lie between your system and the remote system.

Additionally, software and websites exist that will graphically approximate the location of the IP:

http://www.iplocationfinder.com/

Screenshots of an investigation into the country or location of the suspect IP addresses, as evidence that you have completed this part of the task.

Label it or save it as "DeadIR-3".

Within Autopsy, view the /var/log/boot.log file. At the top of this file Syslog reports starting at August 10 at 13:33:57. Given what we have learned about this system during timeline analysis, what is suspicious about Syslog restarting on August 10th? Was the system actually restarted at that time?

Note that according to the log, Apache fails to restart. Why can't Apache restart? Do you think the attacker intended to do this?

Open the mail log file, which you recorded the location of earlier. Identify the email addresses that messages were sent to.

Take a screenshot of a list of email addresses, as evidence that you have completed this part of the task.

Label it or save it as "DeadIR-4".

Another valuable source of information are records of commands that have been run by users. One source of this information is the .bash_history file. As noted during timeline analysis, the /root/.bash_history file was symlinked to /dev/null, meaning the history was not saved. However, the attacker did leave behind a Bash history file in the root of the filesystem ("/"). View this file.

Towards the end of this short Bash session the attacker downloads sslstop.tar.gz, then the attacker runs:

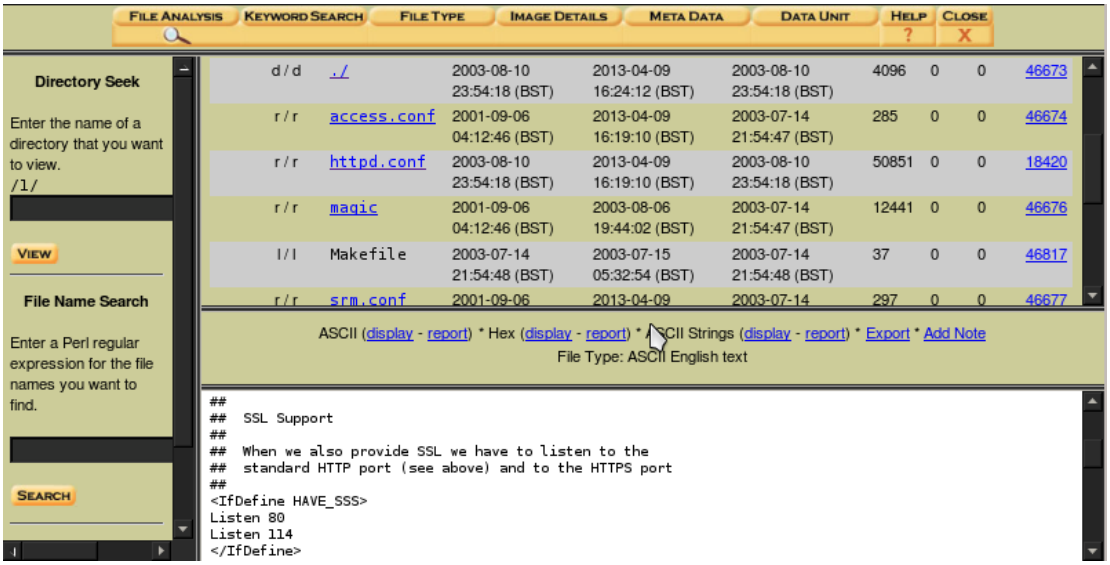
```
ps aux | grep apache

kill -9 21510 21511 23289 23292 23302
```

What is the attacker attempting to do with these commands?

Apache has clearly played an important role in the activity of the attacker, so it is natural to investigate Apache's configuration and logs.

Still in Autopsy, browse to /etc/apache/conf/, and view httpd.conf.

Note that the Apache config has been altered by sslstop, by changing the "HAVE_SSS" directive to "HAVE_SSL" (remember, this file was shown in the timeline to be modified after sslstop was run):

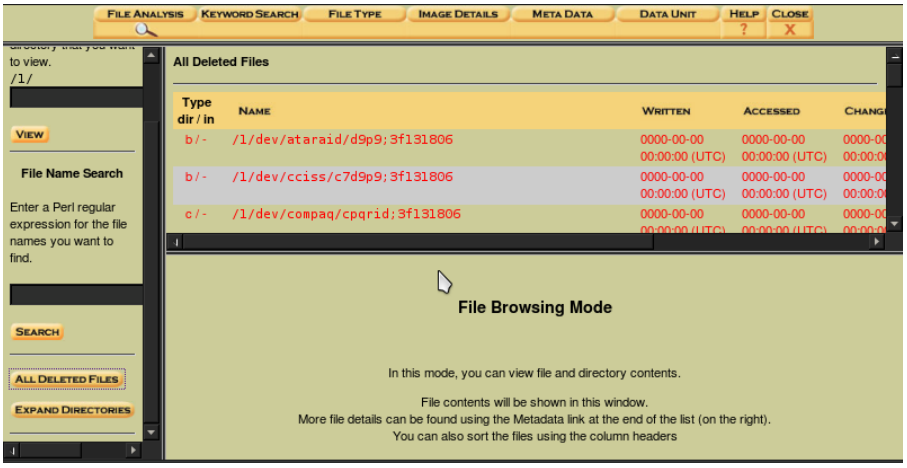


This configuration also specifies that Apache logs are stored in /etc/httpd/logs, and upon investigation this location is symlinked to /var/log/httpd/. This is a common Apache configuration.

Unfortunately the /var/log/httpd/ directory does not exist, so clearly the attacker has attempted to cover their tracks by deleting Apache's log files.

Deleted files analysis

Autopsy can be used to view files that have been deleted. Simply click "All Deleted Files", and browse through the deleted files it has discovered. Some of the deleted files will have known filenames, others will not.



However, this is not an efficient way of searching through content to find relevant information.

Since we are primarily interested in recovering lost log files (which are ASCII human-readable), one of the quickest methods is to extract all unallocated data from our evidence image, and search that for likely log messages. Autopsy has a keyword search. However, manual searching can be more efficient.

In a terminal console in Backtrack, run:

```
blkls -A evidence/sda1.img | strings > evidence/unallocated
```

This will extract all unallocated blocks from the partition, and run this through strings, which reduces it to text only (removing any binary data), and the results are stored in "evidence/unallocated".

Open the extracted information for viewing:

```
less evidence/unallocated
```

Scroll down, and find any deleted email message logs.

Hint: try pressing ":" then type "To:".

What sorts of information was emailed?

To get the list of all email recipients quit less (press 'q'), and run:

```
grep "To:.*@" evidence/unallocated
```

Once again, open the extracted deleted information for viewing:

```
less evidence/unallocated
```

Scroll down until you notice some Bash history. What files have been downloaded using wget? Quit less, and write a grep command to search for wget commands used to download files.

Take a screenshot of a list of wget commands used by the attacker to download files (along with the grep command used), as evidence that you have completed this part of the task.

Label it or save it as "DeadIR-5".

Write a grep command to search for commands used by the attacker to delete files from the system.

Once again, open the extracted deleted information for viewing:

```
less evidence/unallocated
```

Press ":" and type "/shellcode". There a quite a few exploits on this system, not all of which were used in the compromise.

Search for the contents of log files, that were recorded on the day the attack took place:

```
grep "Aug[/ ]10" evidence/unallocated
```

Note that there is an error message from Apache that repeats many times, complaining that it cannot hold a lockfile. This is caused by the attacker having deleted the logs directory, which Apache is using.

If things have gone extremely well the output will include further logs from Apache, including error messages with enough information to search the Internet for information about the exploit that was used to take control of Apache to run arbitrary code. If not, then at some point during live analysis you may have clobbered some deleted files. This is the important piece of information from unallocated disk space:

> [Sun Aug 10 13:24:29 2003] [error] mod_ssl: SSL handshake failed (server localhost.localdomain:443, client 213.154.118.219) (OpenSSL library error follows)
>
> [Sun Aug 10 13:24:29 2003] [error] OpenSSL: error:1406908F:SSL routines:GET_CLIENT_FINISHED:connection id is different

This may indicate the exploitation of this software vulnerability:

> OpenSSL SSLv2 Malformed Client Key Remote Buffer Overflow Vulnerability
>
> http://www.securityfocus.com/bid/5363

Take a screenshot of an investigation into the country or location of the attackers IP address, as evidence that you have completed this part of the task.

Label it or save it as “DeadIR-6”.

Unguided pen-test task

Restart Apache, and do a penetration test, to confirm that Apache/mod_ssl are vulnerable to attack. Start the compromised VM, repair the Apache service by recreating the log directory, re-enable SSL by editing /etc/apache/conf/httpd.conf, and restart httpd. You may download an exploit (for example, from the security advisory

above), or attempt to recreate the attack using Metasploit/Armitage.

Take screenshots showing an attack on the Apache server, as evidence that you have completed this part of the task.

Label it or save it as “DeadIR-A1”.

What to show your tutor for XP rewards

Show your tutor each of the above (in red) evidences. You may be asked to justify your decisions. This will be used to allocate XP for the module. Further details of the XP rewards and requirements are available on the *My XP* site.