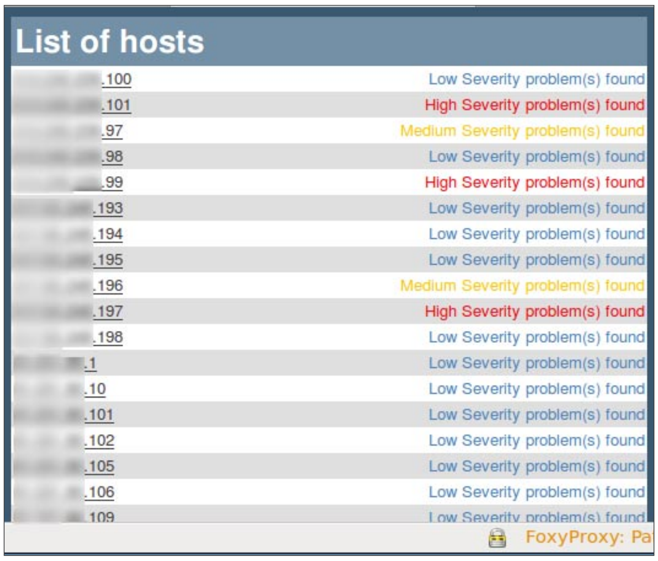
Taming Vulnerability Data

Vulnerability scanners are indispensable both for vulnerability assessments and penetration tests. One of the first things a tester does when faced with a network is fire up a network scanner or even several different ones. And, having scanned a large network with a lot of open ports and services he/she will end up with something like this: see Figure 1.

This is however only the first step. The next steps would usually be:

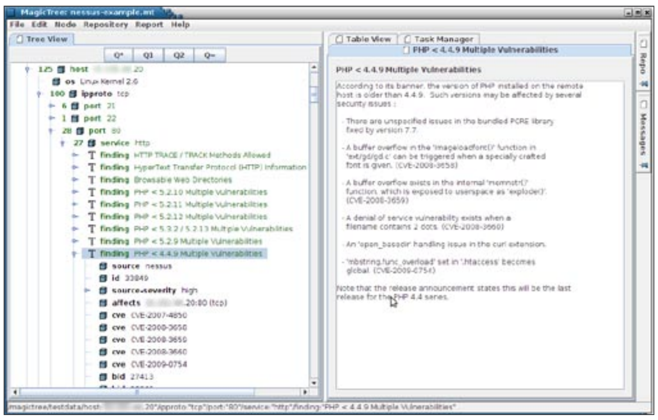


List of hosts

[redacted].100	Low Severity problem(s) found
[redacted].101	High Severity problem(s) found
[redacted].97	Medium Severity problem(s) found
[redacted].98	Low Severity problem(s) found
[redacted].99	High Severity problem(s) found
[redacted].193	Low Severity problem(s) found
[redacted].194	Low Severity problem(s) found
[redacted].195	Low Severity problem(s) found
[redacted].196	Medium Severity problem(s) found
[redacted].197	High Severity problem(s) found
[redacted].198	Low Severity problem(s) found
[redacted].1	Low Severity problem(s) found
[redacted].10	Low Severity problem(s) found
[redacted].101	Low Severity problem(s) found
[redacted].102	Low Severity problem(s) found
[redacted].105	Low Severity problem(s) found
[redacted].106	Low Severity problem(s) found
[redacted].109	Low Severity problem(s) found

Figure 1. *Nessus report. 404 hosts, 960 open ports, 3286 findings*

- Get a general idea of what is there. Are there any problems that look exploitable? What are common problems? What hosts are affected?
- Weed out the findings that are obviously bogus or unimportant.
- Attempt exploitation to verify the findings. In case of a penetration test this step will include exploiting, establishing pivot points, escalating privileges, gathering further information, and so on. In any case it will involve exploiting vulnerabilities and recording the results of the exploitation.
- Gather additional information if something is missing



MagicTree: nessus-example.mt
File Edit Node Repository Report Help
Tree View
Q* Q1 Q2 Q=
Table View
Task Manager
Repo
Messages

```
125 host [redacted] 20
    os Linux Kernel 2.6
  100 ipproto tcp
    6 port 21
    1 port 22
    28 port 80
      27 service http
        finding HTTP TRACE / TRACK Methods Allowed
        finding HyperText Transfer Protocol (HTTP) Information
        finding Browsable Web Directories
        finding PHP < 5.2.10 Multiple Vulnerabilities
        finding PHP < 5.2.11 Multiple Vulnerabilities
        finding PHP < 5.2.12 Multiple Vulnerabilities
        finding PHP < 5.3.2 / 5.2.13 Multiple Vulnerabilities
        finding PHP < 5.2.9 Multiple Vulnerabilities
        finding PHP < 4.4.9 Multiple Vulnerabilities
          source nessus
          id 33849
          source-severity high
          affects [redacted].20:80 (tcp)
          cve CVE-2007-4850
          cve CVE-2008-3658
          cve CVE-2008-3659
          cve CVE-2008-3660
          cve CVE-2009-0754
          bid 27413
```

PHP < 4.4.9 Multiple Vulnerabilities

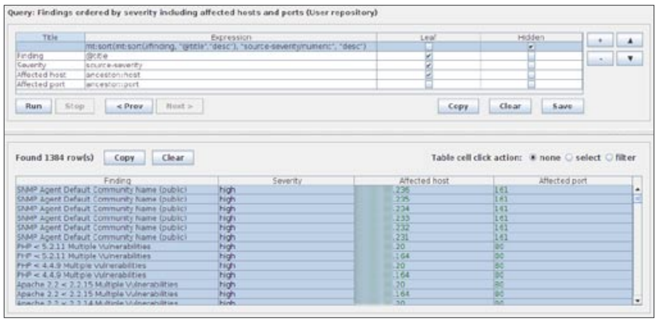
According to its banner, the version of PHP installed on the remote host is older than 4.4.9. Such versions may be affected by several security issues :

- There are unspecified issues in the bundled PCRE library fixed by version 7.7.
- A buffer overflow in the 'imageloadfont()' function in 'ext/gd/gd.c' can be triggered when a specially crafted font is given. (CVE-2008-3658)
- A buffer overflow exists in the internal 'memnstr()' function, which is exposed to userspace as 'explode()'. (CVE-2008-3659)
- A denial of service vulnerability exists when a filename contains 2 dots. (CVE-2008-3660)
- An 'open_basedir' handling issue in the curl extension.
- 'mbstring.func_overload' set in '.htaccess' becomes global. (CVE-2009-0754)

Note that the release announcement states this will be the last release for the PHP 4.4 series.

```
magictree/testdata/host:[redacted]20"/ipproto:"tcp"/port:"80"/service:"http"/finding:"PHP < 4.4.9 Multiple Vulnerabilities"
```

Figure 2. *MagicTree shows a host, ports, service and finding in Tree View*



Query: Findings ordered by severity including affected hosts and ports (User repository)

Title	Expression	Leaf	Hidden
	mt:sort(mt:sort(//finding, "@title","desc"), "source-severity/numeric", "desc")	☐	✓
Finding	@title	✓	☐
Severity	source-severity	✓	☐
Affected host	ancestor::host	✓	☐
Affected port	ancestor::port	☐	☐

Found 1384 row(s)

Table cell click action: [✓ none] select filter

Finding	Severity	Affected host	Affected port
SNMP Agent Default Community Name (public)	high	[redacted].236	161
SNMP Agent Default Community Name (public)	high	[redacted].235	161
SNMP Agent Default Community Name (public)	high	[redacted].234	161
SNMP Agent Default Community Name (public)	high	[redacted].233	161
SNMP Agent Default Community Name (public)	high	[redacted].232	161
SNMP Agent Default Community Name (public)	high	[redacted].231	161
PHP < 5.2.11 Multiple Vulnerabilities	high	[redacted].20	80
PHP < 5.2.11 Multiple Vulnerabilities	high	[redacted].164	80
PHP < 4.4.9 Multiple Vulnerabilities	high	[redacted].20	80
PHP < 4.4.9 Multiple Vulnerabilities	high	[redacted].164	80
Apache 2.2 < 2.2.15 Multiple Vulnerabilities	high	[redacted].20	80
Apache 2.2 < 2.2.15 Multiple Vulnerabilities	high	[redacted].164	80
Apache 2.2 < 2.2.14 Multiple Vulnerabilities	high	[redacted].20	80

Figure 3. *MagicTree TableView lists of findings by severity and type*

- Produce a human readable report including only the relevant findings. The *human readable* part is what makes it difficult. By that we mean a report that a human, such as a security officer or a system administrator will actually read it in full or easily find the parts that are relevant to him/her without missing anything important. A thousand page report where findings are listed by host with tens of findings for each of a hundred of hosts does not qualify as such.

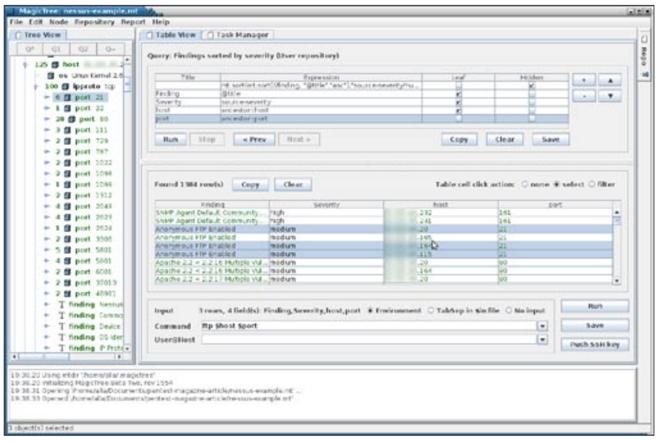
Unfortunately, vulnerability scanners, such as Nessus or OpenVAS, stop at producing the report. What you do with this data and how you do it is up to you.

Having faced these problems regularly, we came up with a solution. Enter MagicTree.

Making Sense of Data

In vulnerability assessments we deal with various kinds of objects such hosts, ports, services, applications, and vulnerabilities. Hosts have IP addresses, DNS names, operating systems, and various other attributes. They also have TCP/UDP ports, that may be open, closed or filtered. Ports have services or protocols, and also software listening on those ports. And all those might have vulnerabilities.

You will notice that these objects are naturally organized in a tree-like structure. This is the structure that MagicTree uses to represent the data: see Figure 2.



Query: Findings sorted by severity (User repository)

Found 1384 row(s)

Table cell click action: none [✓ select] filter

Finding	Severity	host	port
SNMP Agent Default Community...	high	[redacted].232	161
SNMP Agent Default Community...	high	[redacted].231	161
Anonymous FTP Enabled	medium	[redacted].20	21
Anonymous FTP Enabled	medium	[redacted].195	21
Anonymous FTP Enabled	medium	[redacted].164	21
Anonymous FTP Enabled	medium	[redacted].115	21
Apache 2.2 < 2.2.16 Multiple Vul...	medium	[redacted].20	80
Apache 2.2 < 2.2.16 Multiple Vul...	medium	[redacted].164	80
Apache 2.2 < 2.2.17 Multiple Vul...	medium	[redacted].20	80

Input: 3 rows, 4 field(s): Finding,Severity,host,port [✓ Environment] TabSep in $in file No input

Command: ftp $host $port

Figure 4. *Selecting systems that have anonymous FTP*

The data can come from a variety of sources, including vulnerability scanners, flat files and even manual entry. In this case we are concerned with vulnerability scanners. Figure 2 shows the data imported from an XML file generated by Nessus.

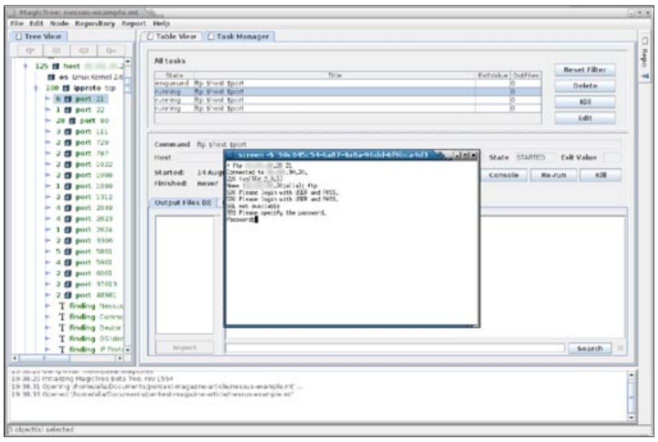
Another way to look at the vulnerability data is by type of the vulnerability. We want to see what kinds of vulnerabilities were found (ordered by severity) and which hosts are affected by each type of vulnerability. This will give us the general idea of how bad (or how good, depending on your point of view) things are, and what can be exploited. Having this kind of view we can proceed to exploiting individual vulnerabilities. MagicTree lets you do just that.

MagicTree allows querying data in the tree using XPath expressions. The data shown in Figure 3 is a result of such a query. A number of useful queries, such as this one, a pre-defined and stored in the query repository. Users can save queries they define to the repository and reuse them.

Using the Data

The tester's next step will usually be exploiting or verifying the discovered vulnerabilities. As an example let's consider anonymous FTP servers. In our scan, several FTP servers that allow anonymous access were discovered. We want to investigate those servers to see what files are accessible. In addition to that we want to run a password guessing attack on those servers to try to get authenticated FTP access. Let's also assume that the test scope was set, we were given a list of hosts that are critical and fragile, so no intrusive tests should be performed on them. One of our anonymous FTP servers is in the list, so we must exclude it from password brute forcing attack.

In the list of findings we can select all *Anonymous FTP Server*"findings. The table contains the sever IP addresses and port numbers. One of the servers is



All tasks

State	Title	ExitValue	OutFiles
enqueued	ftp $host $port		0
running	ftp $host $port		0
running	ftp $host $port		0
running	ftp $host $port		0

Command: ftp $host $port

State: STARTED

```
> ftp [redacted].20 21
Connected to [redacted].90.20.
220 (vsFTPd 2.0.5)
Name ([redacted].20:alla): ftp
530 Please login with USER and PASS.
530 Please login with USER and PASS.
KERBEROS_V4 rejected as an authentication type
331 Please specify the password.
Password:
```

Figure 4. *Running ftp for selected hosts and ports and opening FTP command console*

out of bounds for us, so we control click on the row corresponding to that server and de-select it.

First we are going to connect with FTP to each of the servers and have a look around. We want to avoid having to copy and paste the IP addresses and port numbers, and then having to copy and paste the output into the report.

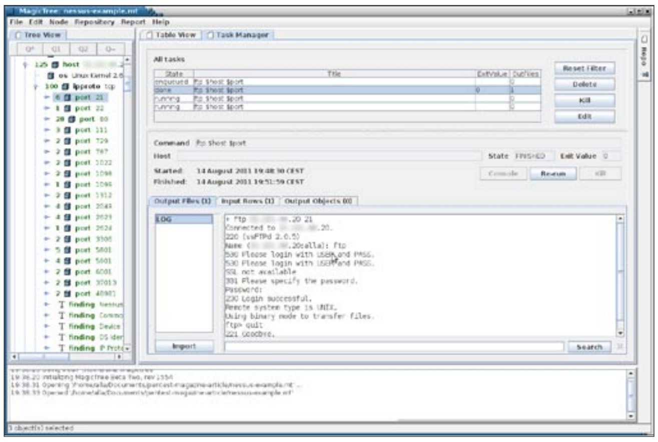
In MagicTree, under the table of results, there is an input field for the command line. We put in `ftp $host $port`, select *Environment* input mode and click Run. This will start one FTP command for each host and port in the list. Next we can open console for each running FTP session and look around.

When we terminate the FTP command, the console window closes, but the command output is saved. The output of all commands executed from MagicTree can be included in the generated report. Of course, commands that produced no useful output or were executed by mistake can be removed.

In a similar way we can run a password bruteforcer, such as hydra or medusa, feeding it the list of IP addresses, or running one instance per host. Again, all the output is retained by MagicTree and can be used in reports.

Advanced Commands Execution

The example above has shown some features related to the execution of external commands. We saw MagicTree launching a command-line FTP client, and passing target specification in environment variables, named after columns in the table view. The commands were executed locally, in parallel in small batches. Now we will consider slightly more complicated example. Suppose you have just scanned a bulk of class C networks using nmap in fast mode and now want to do full port scan, but only for hosts running web servers. We will want it to feed large bulk of data to nmap and execute the scanner on a remote host.



```
Started:   14 August 2011 19:48:30 CEST
Finished:  14 August 2011 19:51:59 CEST
```

```
+ ftp [redacted].20 21
Connected to [redacted].20.
220 (vsFTPd 2.0.5)
Name ([redacted].20:alla): ftp
530 Please login with USER and PASS.
530 Please login with USER and PASS.
SSL not available
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> quit
221 Goodbye.
```

Figure 6. *Browsing the FTP directories – screen log*

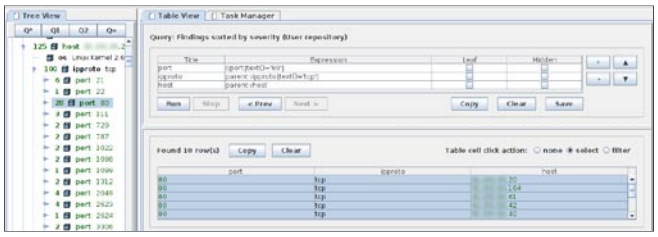


Figure 7. *Automatic query using Q2*

Preparing Input Data

First we need to mine our tree for IP addresses of hosts having HTTP services open. One way to do it is to manually find an one TCP/80 port and then running a query to find similar nodes in the tree. This is done by searching for *80* in the tree and click on Q2 button above the tree. This button activates Query Wizard which builds a query to select all nodes in the tree having the same values in the two last nodes in the path (*port=80 and ipproto=tcp*).

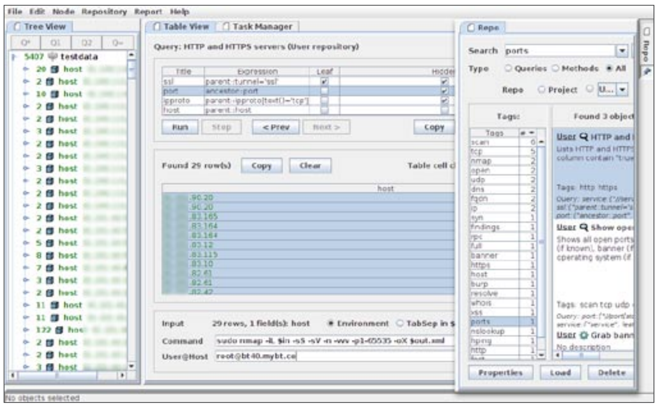
Another option is to find all ports where HTTP service has been detected, plain or wrapped in SSL. The query for doing the latter is provided in the repository, just click on *Repo* tab, and find *HTTP and HTTPS servers* query under *ports* tag. Double click on the query and it will be automatically loaded and executed into the table view.

Launching the Command

Command editor is located under the table containing the query results. In the command editor we switch input mode to *Tabsep in $in file*, and type the command:

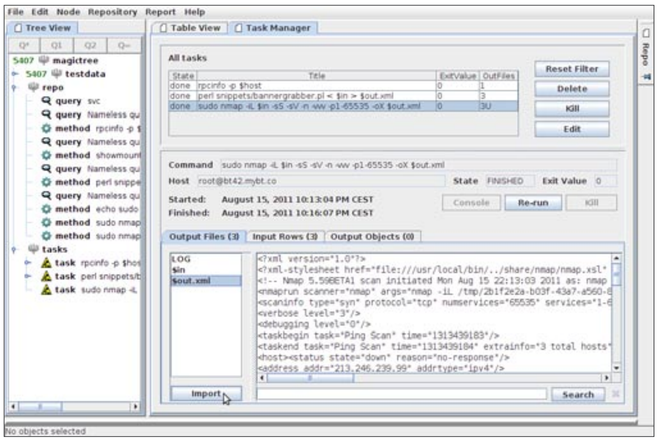
```
sudo nmap -iL $in -sS -sV -n -vvv -p1-65535 -oX $out.xml".
```

When we launch the command, MagicTree will create a temporary file with the list of IP addresses and pass it to nmap via `$in` parameter. The command will run in background, detached from MagicTree, but you will have access to the interactive console which comes handy if you need enter password for *sudo*.



```
Input      29 rows, 1 field(s): host    (•) Environment  ( ) TabSep in $
Command    sudo nmap -iL $in -sS -sV -n -vvv -p1-65535 -oX $out.xml
User@Host  root@bt40.mybt.ce
```

Figure 8. *Ready to run the full port scan of web servers*



All tasks

State	Title	ExitValue	OutFiles
done	rpcinfo -p $host	0	1
done	perl snippets/bannergrabber.pl < $in > $out.xml	0	3
done	sudo nmap -iL $in -sS -sV -n -vv -p1-65535 -oX $out.xml	0	3U

Command sudo nmap -iL $in -sS -sV -n -vv -p1-65535 -oX $out.xml
Host root@bt42.mybt.co State FINISHED Exit Value 0
Started: August 15, 2011 10:13:04 PM CEST
Finished: August 15, 2011 10:16:07 PM CEST

```
<?xml version="1.0"?>
<?xml-stylesheet href="file:///usr/local/bin/../share/nmap/nmap.xsl"
<!-- Nmap 5.59BETA1 scan initiated Mon Aug 15 22:13:03 2011 as: nmap
<nmaprun scanner="nmap" args="nmap -iL /tmp/2b1f2e2a-b03f-43a7-a560-6
<scaninfo type="syn" protocol="tcp" numservices="65535" services="1-6
<verbose level="3"/>
<debugging level="0"/>
<taskbegin task="Ping Scan" time="1313439183"/>
<taskend task="Ping Scan" time="1313439184" extrainfo="3 total hosts"
<host><status state="down" reason="no-response"/>
<address addr="213.246.239.99" addrtype="ipv4"/>
```

Figure 9. *Output files produced by command can be imported back into the tree*

After a command is launched, you can save the project, quit MagicTree and go home. Later you can open the project again and MagicTree will automatically check the status of commands it has launched and pick up its output files if the work is done.

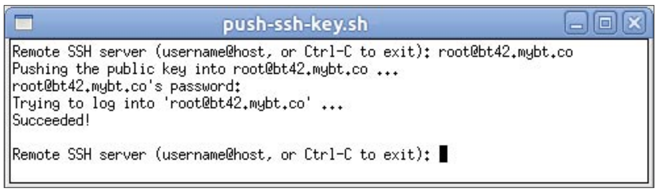
About Output Files

MagicTree will retain the data the command outputs to the console. Additionally, it provides special parameter `$out`. It can be used in output file name parameters. In the above example we have passed `-oX $out.xml` to nmap. This will let MagicTree pick up the file produced by Nmap and store it. You could have written `-oA $out` instead and have all kind of nmap output stored in the tree, not only XML.

When the command finishes, you can browse the list of output files and their content. For XML files, you can click on Import button to import it back into the tree.

Taking import of nmap XML as an example, newly discovered ports will appear in the tree next to the ones previously discovered by Nessus.

Some tools such as DnsEnum and WhatWeb natively produce XML files in MagicTree format. As of time of writing this article, MagicTree supported XML files generated by Nessus, Nmap, Nikto, Acunetix, Burp, W3AF, Qualys, and OpenVAS 3. Users can develop XSLT transforms to support other tools that generate XML. Output of commands incapable of producing XML by themselves, can be imported into MagicTree by creating a wrapper.



```
Remote SSH server (username@host, or Ctrl-C to exit): root@bt42.mybt.co
Pushing the public key into root@bt42.mybt.co ...
root@bt42.mybt.co's password:
Trying to log into 'root@bt42.mybt.co' ...
Succeeded!

Remote SSH server (username@host, or Ctrl-C to exit): █
```

Figure 10. *Configuring remote BackTrack server to run commands from MagicTree*

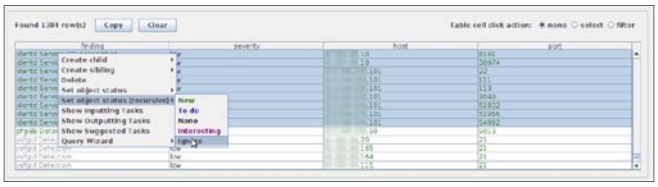


Figure 11. *Marking out findings that we don't want in the final report*

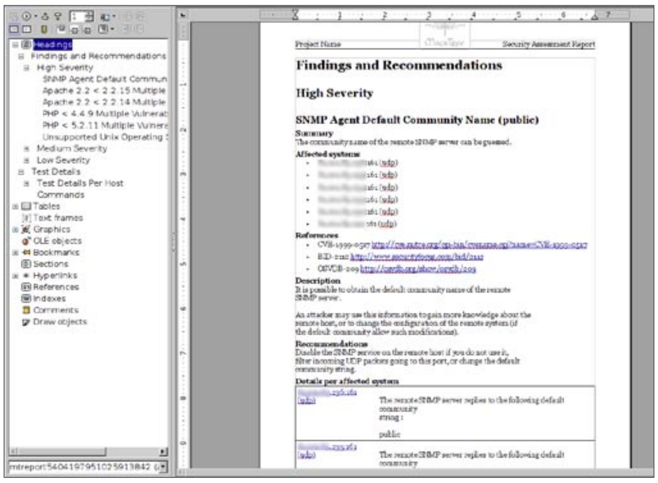
Remote Command Execution

Most likely you have already noticed *User@host* input fields beneath *Command*. It is related to another feature we would like to mention, remote command execution. Everything we have already said about running command from MagicTree – parameterization via environment variables or `$in`, output files pickup, XML file import, access to interactive console, detached execution – will work if a command is executed remotely. MagicTree uses no agents or third-party software installed on the remote host, it just needs to be able to login over SSH and find common *nix utilities.

To setup remote access, click on *Push SSH Key* button. A script running in newly open terminal window will generate a private SSH key (stored under `~/.magictree`) and push it into authorized keys of the specified remote SSH server. How it looks like when I configure MagicTree to work with remote BackTrack 5 VPS, from HackingMachines: (Figure 10). Pushing has succeeded, so we can just enter *root@bt42.mybt.co* in *User@host* and have the command executed remotely.

Preparing the Report

Another challenge in processing the vulnerability scanner results is weeding out the findings that are useless, unimportant or plain wrong. Those will include findings that have no security impact and are no use to the client (stuff like *Traceroute information* or *Nessus scan information*) and false positives. Again MagicTree can help.


Headings
Findings and Recommendations
High Severity
SNMP Agent Default Commun
Apache 2.2 < 2.2.15 Multiple
Apache 2.2 < 2.2.14 Multiple
PHP < 4.4.9 Multiple Vulnerat
PHP < 5.2.11 Multiple Vulnera
Unsupported Unix Operating
Medium Severity
Low Severity
Test Details
Test Details Per Host
Commands
Tables
Text frames
Graphics
OLE objects
Bookmarks
Sections
Hyperlinks
References
Indexes
Comments
Draw objects

Project Name Security Assessment Report

Findings and Recommendations

High Severity

SNMP Agent Default Community Name (public)

Summary
The community name of the remote SNMP server can be guessed.

Affected systems
- [redacted]:161 (udp)
- [redacted]:161 (udp)
- [redacted]:161 (udp)
- [redacted]:161 (udp)
- [redacted]:161 (udp)
- [redacted]:161 (udp)

References
- CVE-1999-0517 http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0517
- BID-2112 http://www.securityfocus.com/bid/2112
- OSVDB-209 http://osvdb.org/show/osvdb/209

Description
It is possible to obtain the default community name of the remote SNMP server.

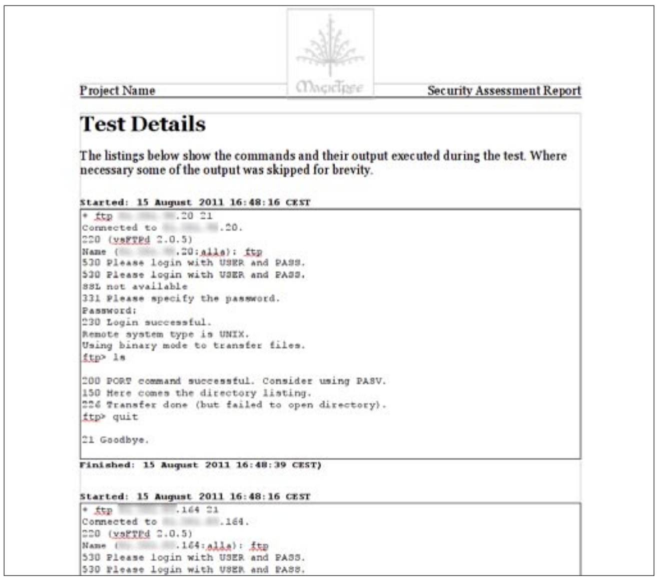
An attacker may use this information to gain more knowledge about the remote host, or to change the configuration of the remote system (if the default community allow such modifications).

Recommendations
Disable the SNMP service on the remote host if you do not use it, filter incoming UDP packets going to this port, or change the default community string.

Details per affected system

[redacted] (udp)	The remote SNMP server replies to the following default community string : public
[redacted] (udp)	The remote SNMP server replies to the following default community

Figure 12. *Generated report*



Project Name Security Assessment Report

Test Details

The listings below show the commands and their output executed during the test. Where necessary some of the output was skipped for brevity.

```
Started: 15 August 2011 16:48:16 CEST
* ftp [redacted].20 21
Connected to [redacted].20.
220 (vsFTPd 2.0.5)
Name ([redacted].20:alla): ftp
530 Please login with USER and PASS.
530 Please login with USER and PASS.
SSL not available
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls

200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
226 Transfer done (but failed to open directory).
ftp> quit

21 Goodbye.
Finished: 15 August 2011 16:48:39 CEST)

Started: 15 August 2011 16:48:16 CEST
* ftp [redacted].164 21
Connected to [redacted].164.
220 (vsFTPd 2.0.5)
Name ([redacted].164:alla): ftp
530 Please login with USER and PASS.
530 Please login with USER and PASS.
```

Figure 13. *Command listings in the report*

While deciding what is important and worth reporting and what isn't should be left to the human tester, the process of marking the findings as important or not and keeping only the important one in the final report, can and should be made easier.

Let's see how this can be done with MagicTree. We'll go back to the list of findings. Starting from the bottom, where all the low risk problems are, we will select findings we consider irrelevant and change their status to *ignore*.

The color of the table cells reflects the status. If we don't want to see findings marked as ignored in the list at all, all we need to do is change the query, adding a condition `[@status!='ignored']`, and re-run it.

Report Generation

After all this hard work sorting out and verifying the findings we want a report to show to the customer. The problem with the reports that vulnerability scanners, such as Nessus, generate is that they are difficult or impossible to edit and don't allow for much customization. MagicTree solves this problem, firstly by producing reports in Microsoft Office or OpenOffic Writer formats, so they are fully editable, and secondly providing fully customizable templates (also in Word and OpenOffice format), that can be modified to your liking. The templates use the same XPath expression syntax as the queries, so any data in the tree can be included in the report.

Let's start with one of the standard report templates that comes with MagicTree. Selecting *Report-> Generate Report* from the menu, choosing *summary-of-findings-cross-referenced.odt* template and clicking on the Generate Report button produces the report: (Figure 12).

The first section – Findings and Recommendations – lists the findings grouping them by type, so *Anonymous FTP* is one finding containing the list of affected hosts and Nessus plugin output for each host. The second section – Test Details – contains per host data including the list of findings per host cross-referenced to the first section, and the listing of executed commands. Our FTP commands and brute-forcing sessions end up in this last section: (Figure 13).

In many cases the output of the vulnerability scanner is used during the test but does not make it into the final report, that is custom-written. Still there is often a need to include bulky data, such as, for example, a list of hosts vulnerable to a certain problem, or those that were successfully exploited. Using MagicTree query to list the data and then copying the table output and pasting it into the report can come helpful in this case.

Conclusion

MagicTree simplifies analyzing and managing large bulks of data that vulnerability scanners produce. Automating the tasks that when done manually are boring and time-consuming, MagicTree leaves the penetration tester free to do what he or she is best at – real hacking.

ALLA BEZROUTCHKO AND ALEXANDRE BEZROUTCHKO

Alla Bezroutchko and Alexandre Bezroutchko are the founders of Gremwell and the developers of MagicTree. They have been working as penetration testers for the last 10 years. Alla specializes in web applications, client-side security and source code reviews. Alex specializes in Unix systems, networking and hardware hacking.

ABOUT MAGICTREE

MagicTree is developed by Gremwell. It is written in Java and was tested on Linux, Mac OS X and Windows. It is also included in BackTrack 5 Linux distribution. The latest version of MagicTree is available for download from Gremwell website. MagicTree can be freely downloaded from http://www.gremwell.com/download. Build 1559 includes fixes for bugs we noticed while writing this article.

Detailed documentation is available at http://www.gremwell.com/magictreedoc

A video tutorial may be found at http://www.gremwell.com/video_using_magictree_for_analysing_data

For more information about Gremwell and MagicTree see www.gremwell.com