**<http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2009-0388>**

**Vulnerability Summary for CVE-2009-0388**

**Original release date:**02/04/2009

**Last revised:**02/17/2009

**Source:** US-CERT/NIST

**Overview**

Multiple integer signedness errors in (1) UltraVNC 1.0.2 and 1.0.5 and (2) TightVnc 1.3.9 allow remote VNC servers to cause a denial of service (heap corruption and application crash) or possibly execute arbitrary code via a large length value in a message, related to the (a) ClientConnection::CheckBufferSize and (b) ClientConnection::CheckFileZipBufferSize functions in ClientConnection.cpp.

[**http://www.securityfocus.com/bid/33568/exploit**](http://www.securityfocus.com/bid/33568/exploit)

**Multiple VNC Clients Multiple Integer Overflow Vulnerabilities**  
  
An attacker may exploit these issues by enticing a victim into connecting to a malicious server.  
  
Core Security Technologies has developed a working commercial exploit for its CORE IMPACT product. This exploit is not otherwise publicly available or known to be circulating in the wild.  
  
The following proofs of concept are available:

* /data/vulnerabilities/exploits/33568.py
* /data/vulnerabilities/exploits/33568-desi.py

**http://www.securityfocus.com/archive/1/500632**

-----BEGIN PGP SIGNED MESSAGE-----  
Hash: SHA1  
  
Core Security Technologies - CoreLabs Advisory  
http://www.coresecurity.com/corelabs/  
  
VNC Multiple Integer Overflows  
  
1. \*Advisory Information\*  
  
Title: VNC Multiple Integer Overflows  
Advisory ID: CORE-2008-1009  
Advisory URL: http://www.coresecurity.com/content/vnc-integer-overflows  
Date published: 2009-02-03  
Date of last update: 2009-02-03  
Vendors contacted: UltraVNC, TightVNC  
Release mode: Coordinated release  
  
2. \*Vulnerability Information\*  
  
Class: Integer overflow  
Remotely Exploitable: Yes  
Locally Exploitable: No  
Bugtraq ID: 33568  
CVE Name: CVE-2009-0388  
  
3. \*Vulnerability Description\*  
  
Multiple integer overflow vulnerabilities have been discovered in  
UltraVNC [1] and TightVNC [2], two (open source) remote control  
applications derived from the popular VNC [3] software.  
  
The vulnerabilities cause a miscalculation of a buffer size on the heap,  
allowing an attacker to corrupt a VNC client heap and can probably allow  
code execution (exploitation is very likely).  
  
4. \*Vulnerable packages\*  
  
. UltraVNC - 1.0.2  
. UltraVNC - 1.0.5  
. TightVnc - 1.3.9  
. Older versions are probably affected too, but they were not tested  
  
5. \*Non-vulnerable packages\*  
  
. UltraVNC - 1.0.5.4  
. TightVNC - 1.3.10  
  
6. \*Vendor Information, Solutions and Workarounds\*  
  
VNC users connecting to untrusted servers should update their VNC  
viewers/clients.  
  
The UltraVNC team has released patched binaries [4] for its viewer.  
Additional information can be found in the UltraVNC Forum  
(http://forum.ultravnc.info/).  
  
The TightVNC team has released patched source code in [5]. TightVNC  
1.3.10 will be released by Feb 10th 2009.  
  
7. \*Credits\*  
  
These vulnerabilities were discovered and researched by Ariel  
Futoransky, Fernando Russ and Alfredo Ortega from Core Security  
Technologies.  
  
8. \*Technical Description / Proof of Concept Code\*  
  
Multiple integer overflow vulnerabilities have been discovered in  
UltraVNC and TightVNC. The vulnerable functions are located in  
'ClientConnection.cpp', and they are:  
  
. 'ClientConnection::CheckBufferSize'  
. 'ClientConnection::CheckFileZipBufferSize'  
  
These functions are used in UltraVNC - 1.0.2 (and previous versions):  
  
. 'ClientConnection::ReadServerCutText() : 3859'  
. 'ClientConnection::Authenticate() : 1701'  
  
And in TightVNC - 1.3.9 (and previous versions):  
  
. 'ClientConnection::ReadServerCutText() : 2951'  
. 'ClientConnection::ReadFailureReason() : 3066'  
  
Other versions may be vulnerable too.  
  
Multiple VNC clients are affected, as they share the vulnerable code.  
The integer overflow follows this pattern:  
  
/-----------  
  
unsigned int len; /\* note the \*unsigned int\* \*/  
  
// read len from the net  
len = network.read\_placeholder();  
  
// check the size to ensure the network related read buffer is of the  
bigger as need  
CheckBufferSize( len ); // or CheckZipBufferSize(len);  
  
// use network related red buffer  
// ...  
  
- -----------/  
  
where 'CheckBufferSize' looks like:  
  
/-----------  
  
(ClientConnection.cpp)  
  
4185: // Makes sure netbuf is at least as big as the specified size.  
4186: // Note that netbuf itself may change as a result of this call.  
4187: // Throws an exception on failure.  
4188: void ClientConnection::CheckBufferSize(int bufsize)  
4189: {  
4190: if (m\_netbufsize > bufsize) return;  
...  
...  
  
- -----------/  
  
and 'CheckZipBufferSize' looks like:  
  
/-----------  
  
(ClientConnection.cpp)  
  
4238: void ClientConnection::CheckFileZipBufferSize(int bufsize)  
4239: {  
4240: unsigned char \*newbuf;  
4241:  
4242: if (m\_filezipbufsize > bufsize) return;  
...  
...  
  
- -----------/  
  
Also, other functions like 'CheckFileZipBufferSize()' and  
'CheckFileChunkBufferSize()' follow the same vulnerable pattern. The  
integer overflow will ensue a heap corruption in the function  
'ReadString()', often called after the bug in 'CheckBufferSize()'. This  
is not a comprehensive list of possible memory corruptions caused by  
this bug, as the vulnerable function is used in many places.  
  
The integer overflow is caused because the data types of the argument  
'bufsize' (signed int) and the buffers size member (unsigned long),  
'm\_netbufsize' and 'm\_filezipbufsize'. Both are 'unsigned long', so:  
  
'(unsigned long)-1 > (int)42 == TRUE'  
  
because all the comparison was "casted" to unsigned long... (0xFFFFFFFF  
> 0x2a).  
  
Steps to reproduce:  
  
The quickest way to reproduce this bug is by modifying the VNC server to  
send crafted evil packets as:  
  
/-----------  
  
(from the TightVNC vncClient.cpp sourcecode...)  
  
358: BOOL vncClientThread::SendTextStringMessage(const char \*str)  
359: {  
360: CARD32 len = Swap32IfLE(strlen(str));  
361: if (!m\_socket->SendExact((char \*)&len, sizeof(len)))  
362: return FALSE;  
363: if (!m\_socket->SendExact(str, strlen(str)))  
364: return FALSE;  
365:  
366: return TRUE;  
367: }  
...  
  
- -----------/  
  
modifying the line 360, a crafted length like 0xFFFFFFFF triggers an  
exception in the following functions:  
  
. In the case of UltraVNC, in 'ClientConnection::Authenticate()'  
. In the case of TightVNC, in 'ClientConnection::ReadFailureReason()'  
  
To trigger the bug in the function 'ClientConnection::CheckBufferSize'  
located in the file 'ClientConnection.cpp' (both vendors):  
  
/-----------  
  
(vncClient.cpp)  
  
1848: void vncClient::UpdateClipText(LPSTR text)  
1849: {  
..  
..  
1858: rfbServerCutTextMsg message;  
1860: message.length = Swap32IfLE(strlen(text));  
1861: if (!SendRFBMsg(rfbServerCutText, (BYTE \*) &message, sizeof(message)))  
1862: {  
1863: Kill();  
1864: return;  
1865: }  
1866: if (!m\_socket->SendQueued(text, strlen(text)))  
1867: {  
1868: Kill();  
1869: return;  
1870: }  
1871: }  
..  
  
- -----------/  
  
In line 1860 the 'message.length' structure must be modified to some  
evil value like 0xFFFFFFFF.  
  
9. \*Report Timeline\*  
  
. 2009-01-09:  
Core notifies the TightVNC team of the vulnerability.  
  
. 2009-01-09:  
Core notifies the UltraVNC team of the vulnerability.  
  
. 2009-01-10:  
The UltraVNC team asks Core for a technical description of the  
vulnerability.  
  
. 2009-01-12:  
Core notifies the TightVNC team of the vulnerability. The previous email  
sent by Core was rejected by the vendor email service.  
  
. 2009-01-12:  
Technical details sent to UltraVNC team by Core.  
  
. 2009-01-14:  
The TightVNC team asks Core for a technical description of the  
vulnerability.  
  
. 2009-01-14:  
Technical details sent to TightVNC team by Core.  
  
. 2009-01-21:  
TightVNC team notifies Core that a fix has been produced, but the  
release of the fixed version (TightVNC 1.3.10) will be available early  
February. TightVNC team releases the fix for its SVN users [5].  
  
. 2009-01-26:  
Core asks TightVNC if the fixed version will be available on  
02-Feb-2009. No reply received.  
  
. 2009-01-26:  
Core asks UltraVNC team if a fixed version is available.  
  
. 2009-01-26:  
UltraVNC team notifies Core that a fixed version will probably be  
available on Feb 1st 2009.  
  
. 2009-01-30:  
Core notifies TightVNC and UltraVNC teams the advisory will be released  
on Feb 3rd 2009, given that the vulnerability was already made public [5].  
  
. 2009-02-02:  
UltraVNC team notifies Core that a fix has been produced and will be  
available to the users on Tuesday, Feb 3rd.  
  
. 2009-02-02:  
TightVNC team notifies Core that a patched version will be available to  
the users on Tuesday, Feb 10th.  
  
. 2009-02-03:  
CORE-2008-1009 advisory is published.  
  
10. \*References\*  
  
[1] http://www.uvnc.com.  
[2] http://www.tightvnc.com.  
[3] http://www.realvnc.com.  
[4] UltraVNC binary patches:  
http://support1.uvnc.com/download/vncviewer\_1054\_w32.zip and  
http://support1.uvnc.com/download/vncviewer\_1054\_X64.zip.  
[5]  
http://vnc-tight.svn.sourceforge.net/viewvc/vnc-tight?view=rev&revision=  
3564.  
  
11. \*About CoreLabs\*  
  
CoreLabs, the research center of Core Security Technologies, is charged  
with anticipating the future needs and requirements for information  
security technologies. We conduct our research in several important  
areas of computer security including system vulnerabilities, cyber  
attack planning and simulation, source code auditing, and cryptography.  
Our results include problem formalization, identification of  
vulnerabilities, novel solutions and prototypes for new technologies.  
CoreLabs regularly publishes security advisories, technical papers,  
project information and shared software tools for public use at:  
http://www.coresecurity.com/corelabs.  
  
12. \*About Core Security Technologies\*  
  
Core Security Technologies develops strategic solutions that help  
security-conscious organizations worldwide develop and maintain a  
proactive process for securing their networks. The company's flagship  
product, CORE IMPACT, is the most comprehensive product for performing  
enterprise security assurance testing. CORE IMPACT evaluates network,  
endpoint and end-user vulnerabilities and identifies what resources are  
exposed. It enables organizations to determine if current security  
investments are detecting and preventing attacks. Core Security  
Technologies augments its leading technology solution with world-class  
security consulting services, including penetration testing and software  
security auditing. Based in Boston, MA and Buenos Aires, Argentina, Core  
Security Technologies can be reached at 617-399-6980 or on the Web at  
http://www.coresecurity.com.  
  
13. \*Disclaimer\*  
  
The contents of this advisory are copyright (c) 2009 Core Security  
Technologies and (c) 2009 CoreLabs, and may be distributed freely  
provided that no fee is charged for this distribution and proper credit  
is given.  
  
14. \*PGP/GPG Keys\*  
  
This advisory has been signed with the GPG key of Core Security  
Technologies advisories team, which is available for download at  
http://www.coresecurity.com/files/attachments/core\_security\_advisories.a  
sc.  
-----BEGIN PGP SIGNATURE-----  
Version: GnuPG v1.4.7 (MingW32)  
Comment: Using GnuPG with Mozilla - http://enigmail.mozdev.org  
  
iD8DBQFJiKUCyNibggitWa0RAvpmAJ0ckztpZ9PyAmA+YE03PNo3O9YCegCeO1HD  
8LdXEbiysMMH42Q4sAQMJqA=  
=CRlF  
-----END PGP SIGNATURE-----