RBS-2013-007

Rockwell Automation RSLinx Enterprise LogReceiver Service Missing Record Data Size Validation Remote Denial of Service

Table of Contents

About Risk Based Security

Risk Based Security offers clients fully integrated security solutions, combining real-time vulnerability and threat data, as well as the analytical resources to understand the implications of the data, resulting in not just security, but the right security.

Company History

Risk Based Security, Inc. (RBS) was established in early 2011 to better support the many users and initiatives of the Open Security Foundation - including the OSVDB and DataLossDB projects. RBS was created to transform this wealth of security data into actionable information by enhancing the research available, and providing a first of its kind risk identification and evidence-based security management service.

As a data driven and vendor neutral organization, RBS is able to deliver focused security solutions that are timely, cost effective, and built to address the specific threats and vulnerabilities most relevant to the organizations we serve. We not only maintain vulnerability and data breach databases, we also use this information to inform our entire practice.

Solutions

Cyber Risk Analytics - Extensive data breach database including interactive dashboards and breach analytics. Clients are able to gather and analyze security threat and data breach information on businesses, industries, geographies, and causes of loss.

VulnDB - Vulnerability intelligence, alerting, and third party library monitoring and tracking based on the largest and most comprehensive vulnerability database available today. Ability to integrate with products and services via an API or custom export.

YourCISO - Revolutionary service that provides organizations an affordable security solution including policies, vulnerability scans, awareness material, incident response, and access to high quality information security resources and consulting services.

Security Development Lifecycle (SDL) - Consulting, auditing, analysis, and independent verification specifically specialized in breaking code, which in turn greatly increases the security of software products.

Security Program Assessment Services / ISO/IEC 27001:2005 - Customized training, security assessments, security program audits, and gap analysis as well as pre-certification consulting services to both protect organizations with best practice security controls and to prepare for a smooth ISO/IEC 27001:2005 certification audit.

Vulnerable Program Details

Vendor:	Rockwell Automation
Product:	RSLinx Enterprise
Version:	5.50.04 CPR 9 SR 5
Component:	LogReceiver.exe
File version:	5.50.4.19
Platform:	Windows Server 2003 R2 Enterprise Edition

References

RBS:	RBS-2013-007
OSVDB:	94846[1], 94852[2]
CVE:	CVE-2013-2805 (Out-of-bounds read)
	CVE-2013-2806 ("End of Current Record" calculation integer overflow)
	CVE-2013-2807 ("Total Record Size" calculation integer overflow)
ICS-CERT:	ICSA-13-095-02A[3]

Credits

Carsten Eiram, Risk Based Security

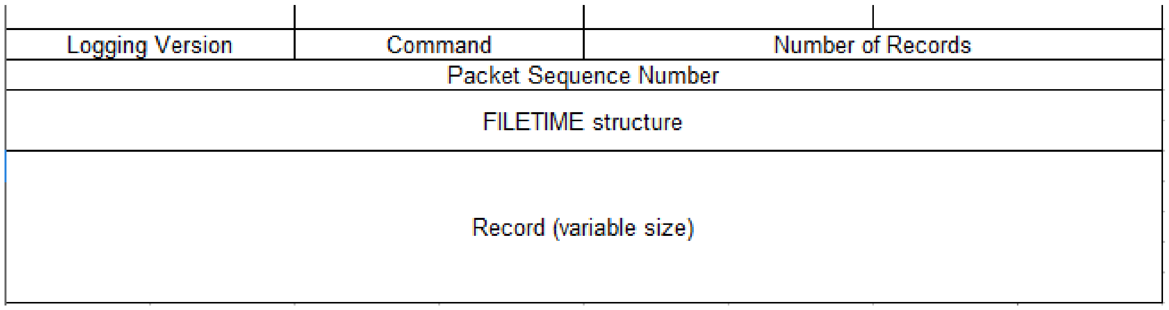
Twitter: @CarstenEiram
Twitter: @RiskBased

[1] http://osvdb.org/show/osvdb/94846
[2] http://osvdb.org/show/osvdb/94852
[3] http://ics-cert.us-cert.gov/advisories/ICSA-13-095-02A

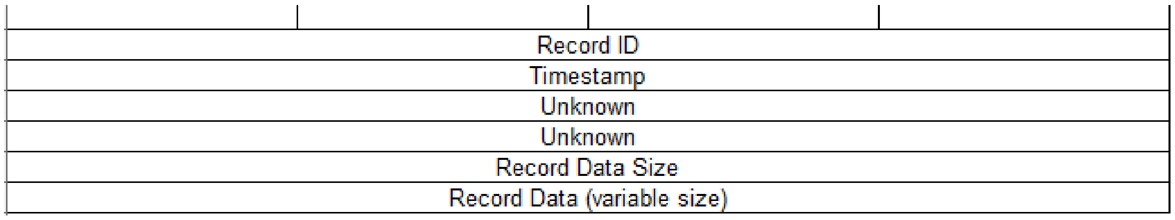
Vulnerability Details

RSLinx Enterprise provides the RSLinx Enterprise Network Event Log Service, `LogReceiver.exe`, which once it's started (disabled by default) binds to UDP port 4444 and listens for incoming datagrams to log network events.

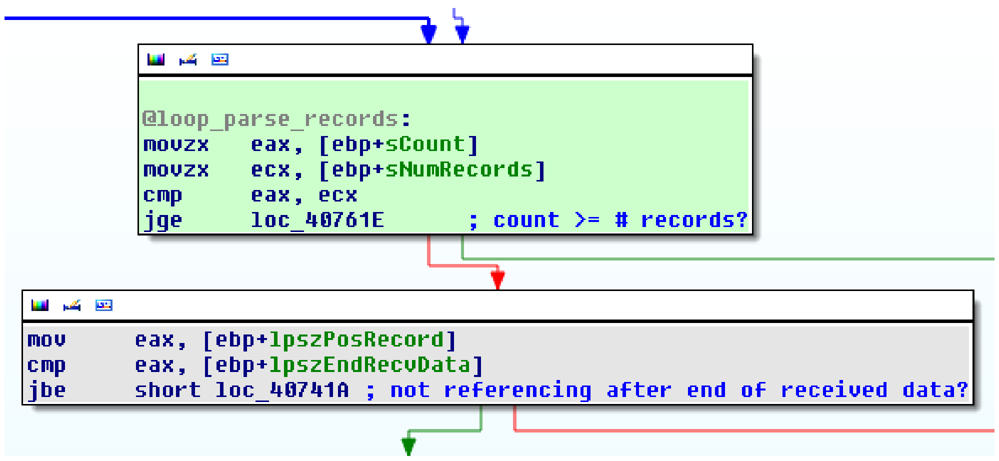
Once a network event message is received, the service starts parsing the datagram, which is expected to have a structure as depicted in the figure below with a variable number of records as specified by the "Number of Records" field.



Each record is expected to contain a 20 byte header optionally followed by a variable amount of data as specified by the "Record Data Size" field value.



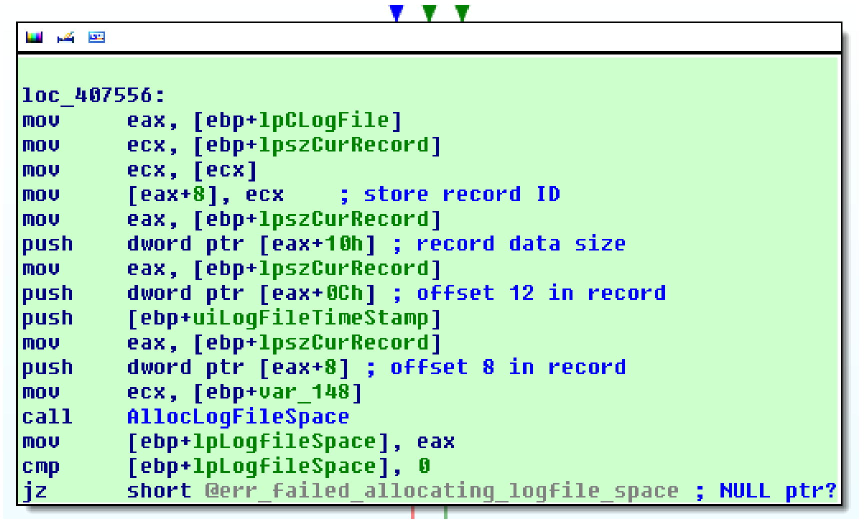
When parsing of the event message header is eventually complete, processing of the contained records commences by entering a loop that parses one record at a time. Processing stops once a total number of records matching the “Number of Records” field value has been processed, or the address of the record to parse references after the end of the received datagram.



```
@loop_parse_records:
movzx   eax, [ebp+sCount]
movzx   ecx, [ebp+sNumRecords]
cmp     eax, ecx
jge     loc_40761E      ; count >= # records?
```

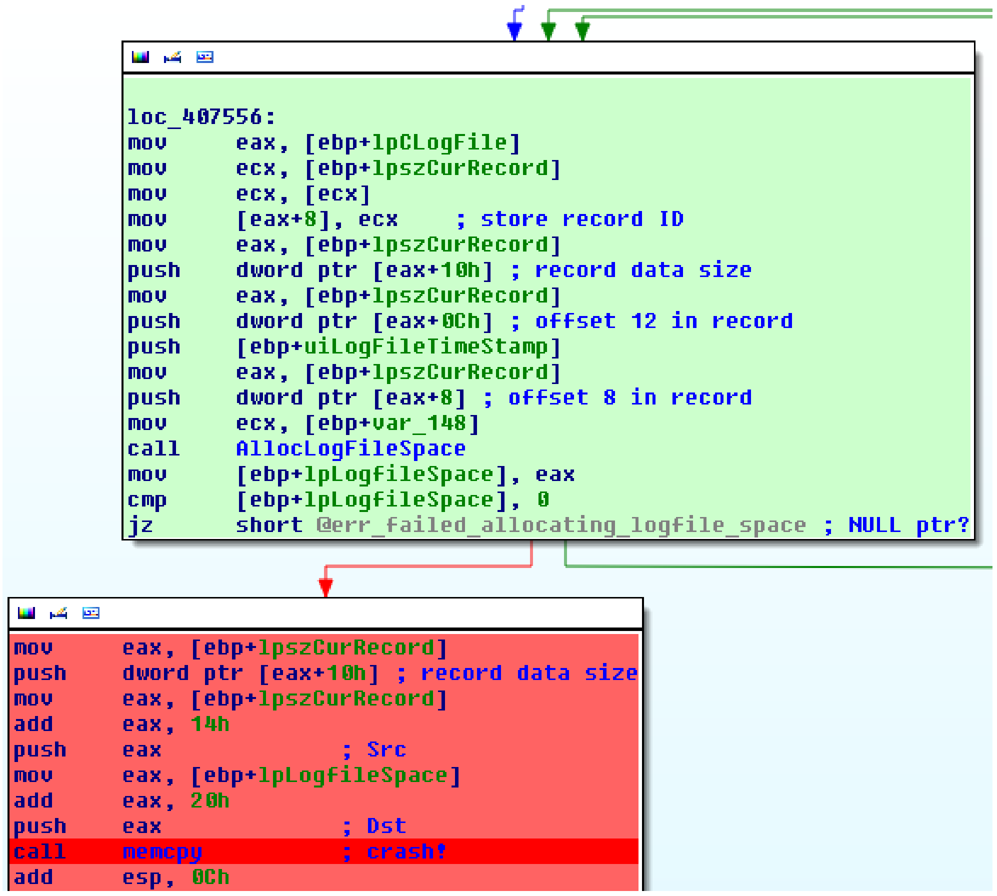
```
mov     eax, [ebp+lpszPosRecord]
cmp     eax, [ebp+lpszEndRecvData]
jbe     short loc_40741A ; not referencing after end of received data?
```

After various inconsequential processing, the function maps log file space to memory for the record.



```
loc_407556:
mov     eax, [ebp+lpCLogFile]
mov     ecx, [ebp+lpszCurRecord]
mov     ecx, [ecx]
mov     [eax+8], ecx     ; store record ID
mov     eax, [ebp+lpszCurRecord]
push    dword ptr [eax+10h] ; record data size
mov     eax, [ebp+lpszCurRecord]
push    dword ptr [eax+0Ch] ; offset 12 in record
push    [ebp+uiLogFileTimeStamp]
mov     eax, [ebp+lpszCurRecord]
push    dword ptr [eax+8] ; offset 8 in record
mov     ecx, [ebp+var_148]
call    AllocLogFileSpace
mov     [ebp+lpLogfileSpace], eax
cmp     [ebp+lpLogfileSpace], 0
jz      short @err_failed_allocating_logfile_space ; NULL ptr?
```

If successful, the record data is copied using the “Record Data Size” field value from the record header as size argument. Since the value at no point is validated to ensure it’s not larger than the actual amount of data received, this may lead of an out-of-bounds read access violation when copying the record data.



```
loc_407556:
mov     eax, [ebp+lpCLogFile]
mov     ecx, [ebp+lpszCurRecord]
mov     ecx, [ecx]
mov     [eax+8], ecx     ; store record ID
mov     eax, [ebp+lpszCurRecord]
push    dword ptr [eax+10h] ; record data size
mov     eax, [ebp+lpszCurRecord]
push    dword ptr [eax+0Ch] ; offset 12 in record
push    [ebp+uiLogFileTimeStamp]
mov     eax, [ebp+lpszCurRecord]
push    dword ptr [eax+8] ; offset 8 in record
mov     ecx, [ebp+var_148]
call    AllocLogFileSpace
mov     [ebp+lpLogfileSpace], eax
cmp     [ebp+lpLogfileSpace], 0
jz      short @err_failed_allocating_logfile_space ; NULL ptr?
```

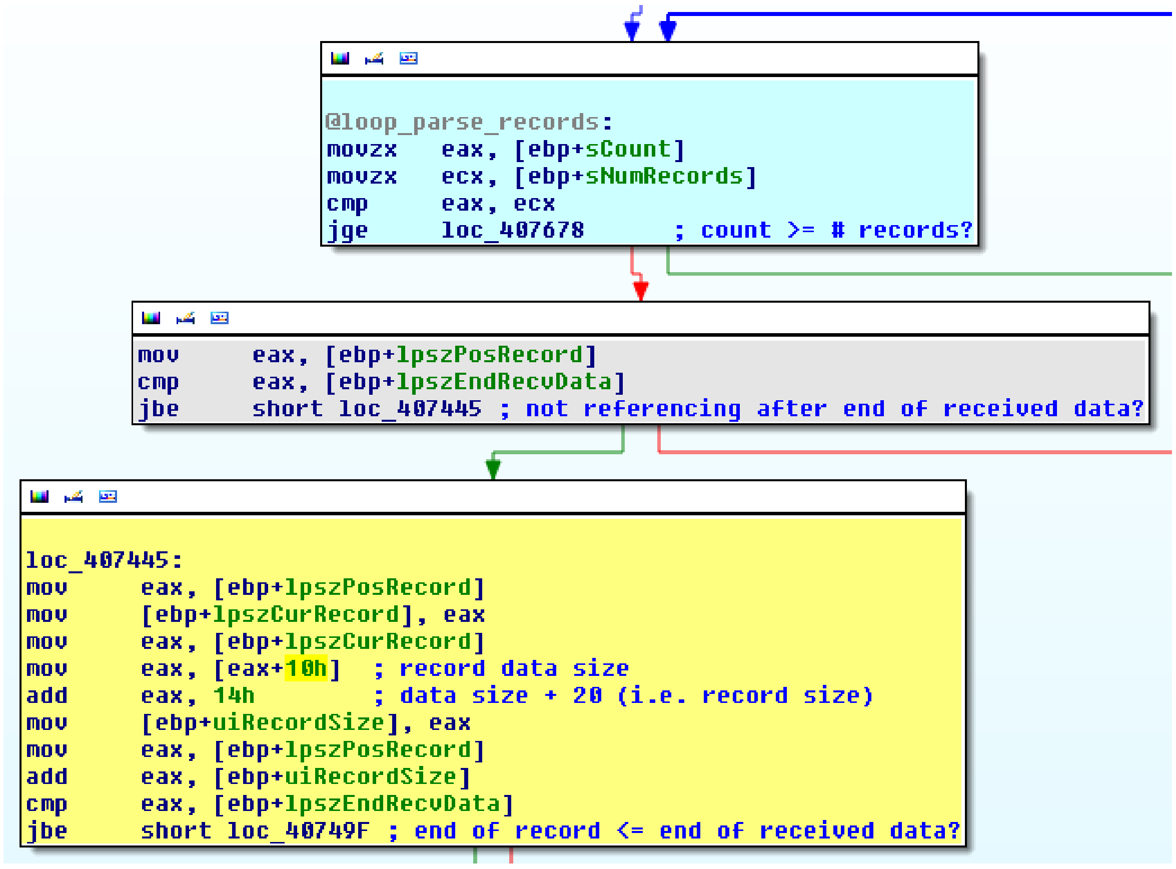
```
mov     eax, [ebp+lpszCurRecord]
push    dword ptr [eax+10h] ; record data size
mov     eax, [ebp+lpszCurRecord]
add     eax, 14h
push    eax                ; Src
mov     eax, [ebp+lpLogfileSpace]
add     eax, 20h
push    eax                ; Dst
call    memcpy             ; crash!
add     esp, 0Ch
```

By supplying a “Record Data Size” field value greater than the actual amount of data sent to the service, a remote attacker can cause the `LogReceiver.exe` service to crash.

Solution

Patched component:	LogReceiver.exe
Initial patched file version:	5.50.6.22, 5.60.2.12
Final patched file version:	5.50.7.23, 5.60.7.17

Rockwell Automation initially released patches (ID534705, ID537302, and ID535962), which addressed one part of the vulnerability by ensuring the size of a record does not exceed the end of the received data before copying the record data. This ensures that the function in most cases does not parse overly long records.



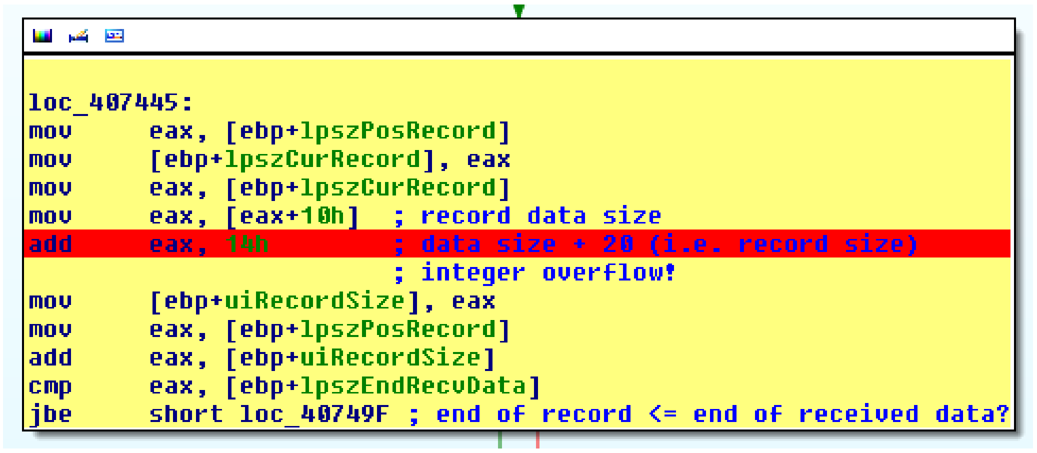
```
@loop_parse_records:
movzx   eax, [ebp+sCount]
movzx   ecx, [ebp+sNumRecords]
cmp     eax, ecx
jge     loc_407678       ; count >= # records?
```

```
mov     eax, [ebp+lpszPosRecord]
cmp     eax, [ebp+lpszEndRecvData]
jbe     short loc_407445 ; not referencing after end of received data?
```

```
loc_407445:
mov     eax, [ebp+lpszPosRecord]
mov     [ebp+lpszCurRecord], eax
mov     eax, [ebp+lpszCurRecord]
mov     eax, [eax+10h]  ; record data size
add     eax, 14h        ; data size + 20 (i.e. record size)
mov     [ebp+uiRecordSize], eax
mov     eax, [ebp+lpszPosRecord]
add     eax, [ebp+uiRecordSize]
cmp     eax, [ebp+lpszEndRecvData]
jbe     short loc_40749F ; end of record <= end of received data?
```

There are, however, two ways of bypassing these checks and still either trigger the original vulnerability or a variant thereof.

In the **first manner**, just prior to the added check, the total record size is calculated by adding 20 (i.e. the record header size) to the "Record Data Size" field value from the event message header. As this calculation does not perform sufficient checks of the result, an integer overflow may occur.



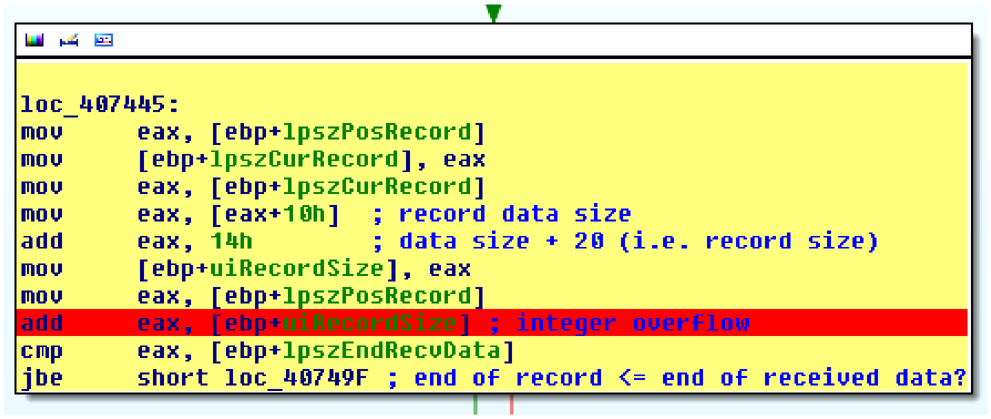
```
loc_407445:
mov     eax, [ebp+lpszPosRecord]
mov     [ebp+lpszCurRecord], eax
mov     eax, [ebp+lpszCurRecord]
mov     eax, [eax+10h]  ; record data size
add     eax, 14h        ; data size + 20 (i.e. record size)
                        ; integer overflow!
mov     [ebp+uiRecordSize], eax
mov     eax, [ebp+lpszPosRecord]
add     eax, [ebp+uiRecordSize]
cmp     eax, [ebp+lpszEndRecvData]
jbe     short loc_40749F ; end of record <= end of received data?
```

By sending a datagram with a value between FFFFFFECh and FFFFFFFFh (both values included) in the "Record Data Size" field, the calculation wraps around and yields a small result for the total record size. As this result is used to calculate the end of the record, which is compared to the end of the received data, the check can be bypassed, as the end of the record is considered to be within the bounds of the received data.

This leads to a similar out-of-bounds read crash as the original when copying the record data into the allocated log file space.

The **second manner** in which the initial patch can be bypassed is due to the check only ensuring that the end of the record does not reference outside the end of the received data; not whether it references prior to the beginning of the received data.

The check is performed by adding the total record size to the address of the beginning of the record and then comparing this result to the address of the end of the received data.

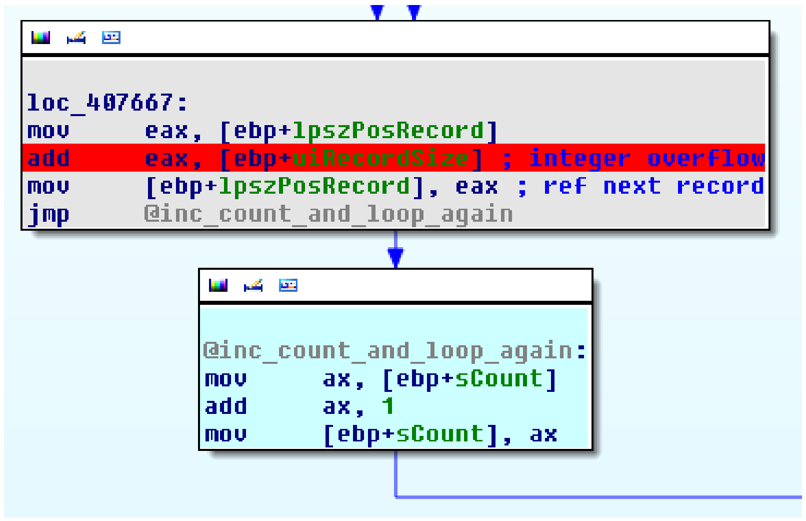


```
loc_407445:
mov     eax, [ebp+lpszPosRecord]
mov     [ebp+lpszCurRecord], eax
mov     eax, [ebp+lpszCurRecord]
mov     eax, [eax+10h]  ; record data size
add     eax, 14h        ; data size + 20 (i.e. record size)
mov     [ebp+uiRecordSize], eax
mov     eax, [ebp+lpszPosRecord]
add     eax, [ebp+uiRecordSize] ; integer overflow
cmp     eax, [ebp+lpszEndRecvData]
jbe     short loc_40749F ; end of record <= end of received data?
```

However, the addition may trigger an integer overflow, causing the calculated end of the current record to wrap-around and reference an address somewhere before the received data. This passes the check, as it only verifies that the record end is not after the end of the received data, and causes the function to start parsing the record.

As the supplied "Record Data Size" field value has to be very large in order to trigger the integer overflow, the function eventually fails to map sufficient log file space for the record.

This causes the function to log an error, after which it adds the total record size to the address of the currently parsed record in order to obtain a reference to the beginning of the following record. Similar to prior calculations, no integer overflow checks are performed.



```
loc_407667:
mov     eax, [ebp+lpszPosRecord]
add     eax, [ebp+uiRecordSize] ; integer overflow
mov     [ebp+lpszPosRecord], eax ; ref next record
jmp     @inc_count_and_loop_again
```

```
@inc_count_and_loop_again:
mov     ax, [ebp+sCount]
add     ax, 1
mov     [ebp+sCount], ax
```

At this point, the pointer to the next record may reference an arbitrary address somewhere before the data received by the service. As the loop iterates to parse the next record, all bounds checks are passed as the record is not considered to be outside the bounds of the received data. When trying to dereference the pointer during parsing, access violations may occur.

Rockwell Automation addressed these variants with new versions of the original patches (ID544798, ID545535, and ID545537).

Timeline

2013/03/27	Vulnerability discovered.
2013/03/27	Vulnerability reported to Rockwell Automation (RA).
2013/03/28	RA informs that a patch to address a vulnerability previously reported by RBS along with other internally discovered issues was just released a few days ago. RA provides the patch and asks RBS to test if it also addresses the new vulnerability.
2013/03/29	RBS provides details to Rockwell Automation that while the latest patch addresses certain aspects of the vulnerability, it can still be triggered due to other flaws in the code and insufficient checks in the fix.
2013/05/23	Updated patches released.
2013/06/28	Rockwell Automation advisory updated with details.
2013/07/05	Alerts published for OSVDB and RBS VulnDB[4].
2013/10/07	ICS-CERT publishes updated advisory.
2013/10/09	Publication of this vulnerability report.

[4] http://www.riskbasedsecurity.com/risk-data-analytics/vulnerability-database/