Systems Applications Proxy

Pwnage

ian@sensepost.com
about: us

Ian de Villiers
What we’re going to talk about

• Why this Talk?
• The history of decompressing SAP DIAG
• Understanding the fundamentals
• New Attacks
• Conclusion
Why this Talk?

- SAP systems carry business critical data
  - Root is nice, but it’s all about the data… 😊
- Any numbers of attacks against SAP systems
  - This talk is not about them…
- Fundamental security shortcoming in the SAP GUI (DIAG) protocol
  - Unencrypted. By Default
  - Compressed
  - This is old news…

[SensePost – 2011]
• SAP is a behemoth
• Very little documentation out there
  – service.sap.com require user accounts 😞
• Documentation for DIAG protocol requires NDA (apparently)… 😞
• Custom toolsets require development
• SAP Basis version used is reasonably outdated..
  – Fine for protocol analysis
  – Some attack scenarios may not be applicable
SensePost Assessments
  - Covered a lot of ground…
  - … but virtually impossible to do a complete job on something as complex
  - Research has been on an “as-time-allows” approach between projects

Releasing tools and research as-is…
  - Let’s see some SAP 0-day in the next couple of months… 😊

Lack of documentation means analysis is probably not spot-on
• Planned to present with SAP on second laptop
• Some technical issues yesterday
  – Running SAP in a VM
  – Laptop is a dog in terms of speed at the moment
What we’re going to talk about

• Why this Talk?
• The history of decompressing SAP DIAG
• Understanding the fundamentals
• New Attacks
• Conclusion
The History…

• Sniffing SAP GUI Passwords
  – Andreas Baus & René Ledosquet from Securon
  – Published 6\textsuperscript{th} July, 2009

• Dealt with playing back captured packets to SAP GUI

• Decompressed data obtained from SAP GUI memory with debugger
But wait…
There’s more…
The History…

• Dennis Yurichev
  – Published 2\textsuperscript{nd} June, 2010
• Discovered that similar compression method was employed in MaxDB
  – Open Source MaxDB code available
• Wrote utility for decompressing SAP traffic
  – Required manual reassembly of data segments over multiple packets
The History…

• Dennis’ research required:
  – Identification of SAP compressed packets by magic
    • 0x1f @ packet.data[17]
    • 0x9d @ packet.data[18]
  – Stringing together of subsequent packets without magic at 17 and 18
  – Once complete “message” had been assembled, we could decompress the data
    • (Decompression won’t work until we have the complete stream)
SAPDecompress – In Pictures
1f 9d == SAP Compressed Message Magic
At packet.data[17] and packet.data[18]
SAP Decompress – In Pictures
SAPDecompress – In Pictures
SAPDecompress – In Pictures
SAPDecompress – In Pictures
SAPDecompress – In Pictures

Compressed Message → SAPDecompress.exe → Decompressed Message
**SAPDecompress – In Pictures**

### Compressed Message

| 00: | 00 00 00 38 00 00 11 00 00 01 00 01 af 01 00 00 |
| 01: | 1f 9d 02 52 e5 f8 bd 24 0c 45 71 fc 5c 9d bf |
| 02: | d6 c8 23 0e a3 a7 6e b7 b7 e4 da ee 0c 34 70 ca |
| 03: | 30 13 1f ac b0 09 0f 0b 97 52 87 56 d4 7f |
| 04: | db 53 7f 47 e7 ae 2d f2 c2 e5 0b 9f f3 3d e2 fb |
| 05: | 65 ac 0a 07 0f 8b 17 40 08 d9 94 36 4c fd |
| 06: | bb 36 a0 56 81 7c cb b1 51 3b 85 1c cd aa 0c 10 |
| 07: | a0 04 fa bf 5d d4 34 28 83 49 14 a1 84 da 09 19 |
| 08: | cf 50 6b 90 d4 b1 c0 a0 01 05 80 9c 59 73 71 2e |
| 09: | 11 6a 7f 5b 46 59 9d ca 7f d3 b9 38 53 34 8a 70 |
| 0a: | 4e c3 cc 25 01 18 |
| 0b: | ce 8e be 99 40 65 |
| 0c: | a3 ed c6 15 f2 ca |
| 0d: | fe 39 be 14 bd ae |
| 0e: | a4 ab 73 de e9 3f |
| 0f: | 42 87 d3 51 c2 |
| 10: | 7f 4a 06 eb f0 75 |
| 11: | d6 cc 71 45 8b 44 |
| 12: | c7 26 43 40 06 65 |
| 13: | 4d 2a 59 aa 13 55 |

---

**Decompressed Message**

| 00: | 01 00 0f 00 00 00 01 00 00 00 0e 01 34 31 31 30 00 |
| 01: | 54 54 46 38 00 01 00 04 09 00 03 37 32 30 10 04 19 |
| 02: | 00 02 00 00 00 00 00 07 00 0a 0f 00 |
| 03: | 00 00 00 00 00 00 00 00 00 00 00 10 00 00 00 |
| 04: | 00 00 00 07 00 00 07 10 04 17 00 02 00 1f 10 04 16 |
| 05: | 00 00 00 00 00 00 00 00 00 00 00 10 00 00 00 |
| 06: | 00 00 13 10 05 01 00 16 00 05 00 00 02 14 11 |
| 07: | 00 00 10 5b 31 10 00 11 00 00 00 10 00 00 10 |
| 08: | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |
| 09: | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |
| 0a: | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |
| 0b: | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |
| 0c: | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |
| 0d: | 78 6d 6c 20 76 65 72 73 69 6f 6e 3d 22 31 2e 30 |
| 0e: | 22 20 65 6e 63 6f 64 69 6e 67 3d 22 73 61 70 2a |
| 0f: | 22 3f 3e 0a 3c 44 41 54 41 4d 41 4e 41 47 4f 52 |
| 10: | 3e 0a 20 20 3c 43 4f 50 59 20 69 64 3d 22 63 6f |
| 11: | 70 79 22 3e 0a 20 20 20 3c 47 55 49 20 69 64 |
| 12: | 3d 22 67 75 69 22 3e 0a 20 20 20 20 20 20 20 20 20 |
| 13: | 45 54 52 49 43 53 20 69 64 3d 22 63 6f 75 69 22 |
| 14: | 63 73 22 20 58 33 3d 22 31 39 32 30 22 20 20 20 |
| 15: | 3d 22 37 22 20 58 31 3d 22 37 32 30 30 32 22 20 20 |
| 16: | 32 38 33 22 20 59 33 3d 22 31 32 30 30 22 20 20 20 |
| 17: | 32 3d 22 32 30 20 3f 3e 0a 20 20 20 20 20 20 20 20 |
| 18: | 30 3d 22 32 38 33 22 2f 3e 0a 20 20 20 20 20 20 20 20 |
| 19: | 47 55 49 3e 0a 20 20 20 2c 43 4f 50 59 3e 0a 3c |
| 20: | 2f 44 41 54 41 4d 41 4e 41 47 4f 52 3e 0a 0c |

---

**Length:** 316 Bytes

---

**Length:** 431 Bytes

---

XML Version: 1.0

Encoding: UTF-8

```xml
<DOCUMENT>
  <COPY id="copy" type="py"/>
  <GUI id="gui"/>
  <METRICS id="metrics"/>
  <XREF id="refs"/>
</DOCUMENT>
```
The History…

• Dennis Yurichev’s work is *awesome*…
• My work is based very much on his discovery…
What we’re going to talk about

• Why this Talk?
• The history of decompressing SAP DIAG
• Understanding the fundamentals
• New Attacks
• Conclusion
The Fundamentals

• Understand the compression
• Understand the compressed protocol
  – Simplify the sniffing and decompression
• Recompression
• Understand the application protocol
  – What makes SAP GUI tick?
• Identify SAP attack vectors not previously considered…
The Compression Algorithm

• Variants of Lempel-Ziv
  – LZC
  – LZH
  – SAP Supports both (tried and tested)
The Compression Algorithm

• Variant of Lempel-Ziv
  – LZC
  – LZH
  – SAP Supports both (tried and tested)
  • Makes one believe that SAP and MaxDB share same compression code-base... 😊
The Compression Algorithm

• Variant of Lempel-Ziv
  – LZC
  – LZH
  – SAP Supports both (tried and tested)
    • Makes one believe that SAP and MaxDB share same compression code-base… 😊

• Version used per message is determined by the Compression Header…
  – This is described in a minute…
The Fundamentals

• Understand the compression
• Understand the compressed protocol
  – Simplify the sniffing and decompression
  – Recompression
• Understand the application protocol
  – What makes SAP GUI tick?
• Identify SAP attack vectors not previously considered…
The Core, Compressed Protocol

• Easy to parse…
The Core, Compressed Protocol

• Easy to parse…

• In the absence of documentation, I’ve had to make my own names…
The Core, Compressed Protocol

• Easy to parse…

• In the absence of documentation, I’ve had to make my own names…
  – SAP Header
  – Compression Header
  – Compressed Data
The Core, Compressed Protocol

- Easy to parse...
- Consist of:
  - "SAP Header"
  - "Compression Header"
  - Compressed Data

[SensePost – 2011]
The Core, Compressed Protocol

- Easy to parse...
- Consist of:
  - "SAP Header"
  - "Compression Header"
  - Compressed Data
The Core, Compressed Protocol

- Easy to parse...
- Consist of:
  - "SAP Header"
  - "Compression Header"
  - Compressed Data

<table>
<thead>
<tr>
<th>SAP Header</th>
<th>Compression Header</th>
<th>Compressed Data</th>
<th>Length: 316 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>00. 00 00 01 00 00 01 00 01</td>
<td>01. 12 1f 9d 02 5e 68 1c 45 71</td>
<td>0c 34 70 ca</td>
<td>03 13 1f 7f 5f 87 67 77 d7 3e 2c 8e 0c 9e 42 87 0c 5c 28 67 67</td>
</tr>
</tbody>
</table>
The SAP Header

The SAP Header

- Bytes [0] – Bytes [3]
  - Len(Sheader) + Len(Cheader) + Len(Cdata) - 4
The SAP Header

• Bytes [0] – [11]

– Bytes [0] – Bytes [3]
  • \( \text{Len(Sheader)} + \text{Len(Cheader)} + \text{Len(Cdata)} - 4 \)

\[
0x0000138 = 312
\]

316 bytes – 4 bytes == 312 bytes
The SAP Header

• Bytes [0] – [11]
  
  – Bytes [0] – Bytes [3]
    • \( \text{Len(Sheader)} + \text{Len(Cheader)} + \text{Len(Cdata)} - 4 \)

    • Unknown (Tampering makes *no* difference)
The Compression Header

The Compression Header

- **Bytes [12] – [19]**

- **Bytes [12] – Bytes [15]**
  - Length of decompressed stream
  - Little-Endian
The Compression Header

  - Length of decompressed stream
  - Little-Endian

  - Length of decompressed stream
  - Little-Endian

0x00001af == 431
The Compression Header


    - Length of decompressed stream
    - Little-Endian

\[
\begin{array}{cccccccccccccccc}
00: & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 \\
01: & 12 & 1f & 9d & 02 & 52 & e5 & f8 & bd & 24 & 0c & 45 & 71 & 0c & 5e & 9d & b1 \\
02: & 5e & a3 & a7 & 6e & b7 & b7 & 1e & da & ee & 0c & 34 & 70 & ca & 0e & 4p & e \\
03: & 0e & 9f & 42 & 87 & oc & 0c & 52 & 87 & 56 & d4 & 7f & 0b & f3 & v0 & 0f \\
\end{array}
\]

Length: 316 Bytes

\[
\begin{array}{cccccccccccccccc}
00: & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 \\
01: & 12 & 1f & 9d & 02 & 52 & e5 & f8 & bd & 24 & 0c & 45 & 71 & 0f & 5e & 9d & b1 \\
02: & 5e & a3 & a7 & 6e & b7 & b7 & 1e & da & ee & 0c & 34 & 70 & ca & 0e & 4p & e \\
03: & 0e & 9f & 42 & 87 & oc & 0c & 52 & 87 & 56 & d4 & 7f & 0b & f3 & v0 & 0f \\
\end{array}
\]

Length: 431 Bytes

\[0x00001af == 431\]
The Compression Header


  – Bytes [16]
  • Version of compression (LZH / LZC)
  • LZC == byte & 0x0f = 0x00
  • LZH == byte & 0x0f = 0x02
The Compression Header

  - Version of compression (LZH / LZC)
    - LZC == byte & 0x0f = 0x00
    - MZH == byte & 0x0f = 0x02

```
#define CS_LZC 0x0 /* use lzc .......................*/
#define CS_LZH 0x2 /* use lzh .......................*/

int CsObjectInt::CsGetAlgorithm(SAP_BYTE * data)
{ /* Get Algorithm number of compressed data */
  return ((int) (data[4] & (unsigned char) 0x0F));
}
```
The Compression Header

  - Bytes [17] – Bytes [18]
    - Compression Magic
    - Always 1f 9d
The Compression Header


- Bytes [19]
  - MaxBits
## Compressed Data

- **Bytes [20] – [N]**
  - The compressed stream

```plaintext
<table>
<thead>
<tr>
<th>Length: 316 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>00 00 01 00 00 11 00 00 01 00 01 af 01 00 00</td>
</tr>
</tbody>
</table>
```

---

[SensePost – 2011]
As an Aside…

  - Length of decompressed stream
  - Little-Endian

  Field is user-controlled, but programmatic type is SAP_INT
  - Signed integer

- What if the original length was 0xffffffff ?
- (thanks Behrang Fouladi)
As an Aside…

  - Length of decompressed stream
  - Little-Endian
  - User input is user-controlled, but programmatic type is SAP_INT
  - What if the original length was 0xffffffff?
As an Aside…

- What if the original length was 0xffffffff?
As an Aside...

- What if the original length was 0xffffffff?
Sniffing SAP Traffic

• SAP traffic does not lend itself very well to WireShark dissectors…

• Answer was to write a custom protocol analysis tool in Java

• Used 3rd Party pCap JNI interface
  – Allows us to use standard pCap filters / dump files

• Use custom built JNI interface built from MaxDB code
SAPCap
SApCap

• Jpcap
  – JNI interface for pCap
  – Responsible for reading packets
    • pCap dump files
    • Sniffing
  – Filtering packets using standard pCap filters
  – Saving information as pCap dump files
SApCap

- SApCap
- Java
- Responsible for:
  - Parsing packet data
  - Decompressing messages
  - Queue management
SApCap

• SapCompress
  – JNI interface
  – Implements MaxDB functions for decompression
    • int[] doDecompress(int[])

[SensePost – 2011]
Demo: SApCap
The Fundamentals

• Understand the compression
• Understand the compressed protocol
  – Simplify the sniffing and decompression
• Recompression
• Understand the application protocol
  – What makes SAP GUI tick?
• Identify SAP attack vectors not previously considered…
Recompression?

• Core decompression functions are defined in vpa105CsObjInt.cpp
  – CsDecompr()

```c
int CsObjectInt::CsDecompr(SAP_BYTE * inbuf, /* ptr input .......*/
    SAP_INT inlen, /* len of input ....*/
    SAP_BYTE * outbuf, /* ptr output .........*/
    SAP_INT outlen, /* len output .........*/
    SAP_INT option, /* decompr. option */
    SAP_INT * bytes_read, /* bytes read .........*/
    SAP_INT * bytes_decompressed) /* bytes decompr. */
```

/* Decompress */
/* */
/* Adaptive Dictionary Compression */
/* Lempel-Zip */
Recompression ?

- But… vpa105CsObjInt.cpp also contains function for what would appear to be compression…
  - CsCompr()

```c
int CsObjectInt::CsCompr(SAP_INT sumlen,
                        SAP_BYTE * inbuf,
                        SAP_INT inlen,
                        SAP_BYTE * outbuf,
                        SAP_INT outlen,
                        SAP_INT option,
                        SAP_INT * bytes_read,
                        SAP_INT * bytes_written)

/*---------------------------------------------*/
/* Compress a memory segmented */
/* */
/* */
/* Adaptive Dictionary Compression */
/* Lempel-Zip */
/* */
```
Recompression?

- We modify our JNI library to make use of MaxDB code
  - doCompress()
  - doDecompress()

```c
#include <jni.h>
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>

#include "SapLib.h"
#include "hpa101saptype.h"
#include "hpa104CsObject.h"
#include "hpa106cs1zc.h"
#include "hpa107cs1zh.h"
#include "hpa105Cs0bjInt.h"

JNIEXPORT jintArray JNICALL Java_com_sensepost_SAPProx_jni_JniInterface__1doDecompress
  (JNIEnv * env, jobject obj, jintArray in) {...}

JNIEXPORT jintArray JNICALL Java_com_sensepost_SAPProx_jni_JniInterface__1doCompress
  (JNIEnv * env, jobject obj, jintArray in) {...}
```
Recompression?

• We now have programmatic interface to:
  – Decompress SAP traffic
    • doDecompress()
    • Useful for interception and sniffing
  – Compress SAP traffic
    • doCompress()
    • Useful for MiTM attacks
    • Useful for assessment of SAP Gui Applications
The Fundamentals

- Understand the compression
- Understand the compressed protocol
  - Simplify the sniffing and decompression
- Recompression
- Understand the application protocol
  - What makes SAP GUI tick?
- Identify SAP attack vectors not previously considered…
The Application Protocol

• Multiplexed
  – Single connection per-user per-location per-host
The Application Protocol

• Multiplexed
  – Single connection per-user per-location per-host
• Initial hand-shake is uncompressed
The Application Protocol

- Multiplexed
  - Single connection per-user per-location per-host
- Initial hand-shake is uncompressed
- Server response is compressed
The Application Protocol

- Multiplexed
  - Single connection per-user per-location per-host
- Initial hand-shake is uncompressed
- Server response is compressed
- Uncompressed component is static
  - Terminal name
  - Options change depending on capabilities of SAP GUI (support bits)
The Application Protocol

- Multiplexed - Single connection per-user per-location per-host
- Initial hand-shake is uncompressed
- Server response is compressed
- Handshake is static (apart from "terminal name")
Message Types

• Two basic Types of Messages
  – Simple Messages
  – Complex Messages

• Message structures differ in terms of direction
  – GUI -> Server
  – Server -> GUI
Simple Messages : GUI -> SAP

• Header
  – “OK Code”
  – Character Encoding
  – GUI Version

• Data
  – Input values
  – XML Stream defining screen metrics
## Simple Messages : GUI -> SAP

### Header
- **OK Code**
- **Character Encoding**
- **GUI Version**

### Data
- **Input values**
- **XML Stream defining screen metrics**

```xml
<?xml version="1.0" encoding="utf-8"?>
<DATAMANAGER>
<Gui>
<metrics />
</Gui>
</DATAMANAGER>
```
Simple Messages : SAP -> GUI

- Header
- Data
- “TH”
Simple Messages : SAP -> GUI

• Header
  – Encoding

[Image of a table showing hexadecimal values and encoding information]
Simple Messages : SAP -> GUI

- Header
  - Encoding
  - Transaction ID
Simple Messages : SAP -> GUI

- Header
  - Encoding
  - Transaction ID
  - System & Hostname
Simple Messages : SAP -> GUI

• Header
  – Encoding
  – Transaction ID
  – System Name
  – Host name
  – Theme
## Simple Messages: SAP -> GUI

- **Data**
  - SAP Program Context & SAP Screen

```plaintext
| 069: | 53 68 69 66 74 00 10 06 06 00 02 00 c8 10 06 07 | Shift E |
| 08a: | 00 24 53 30 30 30 20 20 20 20 20 20 20 20 20 | $S000 |
| 06b: | 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 | T |
| 05c: | 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 | T |
| 06d: | 00 00 00 00 00 16 54 10 0c 07 00 10 00 00 00 | T |
| 06e: | 00 00 00 04 00 00 00 00 00 04 01 04 10 04 1c | T |
| 06f: | 01 2c 10 04 1b 00 01 45 10 04 1c 00 0c 20 20 | E |
| 070: | 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 | , |
| 071: | 30 10 0c 0a 00 14 53 41 30 20 52 2f 33 20 28 | SAP R/3 |
| 072: | 29 20 20 4e 53 30 20 20 20 20 20 20 20 20 20 | (S) |
| 073: | 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 | APMSYST |
| 074: | 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 | 0020 |
| 075: | 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 | (SAPMSYST |
| 076: | 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 | 0020 |
| 077: | 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 | V |
| 078: | 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 | |
| 079: | 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 | |
| 07a: | 00 04 30 30 32 30 12 06 01 00 00 00 56 00 1c | 00 |
| 07b: | 00 00 00 16 00 01 00 00 00 00 00 00 00 00 00 | 0 |
| 07c: | 00 00 53 73 65 00 05 50 00 00 00 00 00 00 00 00 | User U |
| 07d: | 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | Sys |
| 07e: | 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | Item y |
| 07f: | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | Help |
| 080: | 00 00 00 12 00 00 00 00 00 00 00 00 00 00 00 00 | H (E |
| 081: | 02 64 01 01 00 00 00 00 00 00 00 00 00 00 00 00 | Log o |
| 082: | 02 64 01 01 00 00 00 00 00 00 00 00 00 00 00 00 | ge on L |
| 083: | 01 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | $ |
| 084: | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | New |
| 085: | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | password N |
| 086: | 00 00 00 12 0f 01 03 00 00 00 00 00 00 02 06 02 | Log off O & |
| 087: | 00 00 00 02 01 00 00 00 00 00 00 00 00 00 00 00 | C |
| 088: | 72 65 61 73 73 73 73 73 73 73 73 73 73 73 73 6e | Create Session E |
| 089: | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | # A |

(SensePost – 2011)
Simple Messages : SAP -> GUI

• Data
  – SAP Program Context & SAP Screen
  – Menus & Keyboard Accelerators
Simple Messages : SAP -> GUI

• Data
  – SAP Program Context & SAP Screen
  – Menus & Keyboard Accelerators
  – Input dialogs
Simple Messages : SAP -> GUI

- Data
  - SAP Program Context & SAP Screen
  - Menus & Keyboard Accelerators
  - Input dialogs
  - Screen Data
Simple Messages : SAP -> GUI

• “TH”
  – System Name
  – Transaction
  – Transaction ID
Dialogs

• All input fields accept strings
  – No client-side validation
  – Data is validated on the server

• Input field lengths can be manipulated
Dialogs

• All input fields accept strings
  – No client-side validation
  – Data is validated on the server

• Input field lengths can be manipulated
Dialogs

• All input fields accept strings
  – No client-side validation
  – Data is validated on the server
• Input field lengths can be manipulated

SAP

New password

Client 001
User
Password
Language

Information
SAP NetWeaver 7.01 ABAP Version
Simply an almost default install for test and PoC purposes.
For any information, please contact ian@sensepost.com.
Dialogs

• All input fields accept strings
  – No client-side validation
  – Data is validated on the server
• Input field lengths can be manipulated
Dialogs

• All input fields accept strings
  – No client-side validation
  – Data is validated on the server
• Input field lengths can be manipulated
Dialogs

• Length of submitted strings can be adjusted in a similar fashion...
Dialogs

• Length of submitted strings can be adjusted in a similar fashion…
Complex Messages

- Contain the same structures as simple messages...
Complex Messages

• Contain the same structures as simple messages…
• … But include XML structure:
  – <SVARS>
Complex Messages

- Contain the same structures as simple messages...
- ... But include XML structure:

  \[
  \langle \text{SVARS} \rangle
  \]

[SensePost – 2011]
Complex Messages

• Contain the same structures as simple messages…
• … But include XML structure:
  – <SVARS>
• Include compressed streams:
  – PARAMS
Complex Messages

- Contain the same structures as simple messages...
- ... But include XML structure:
  - `<SVARS>`
- Include compressed streams:
  - `PARAMS`

---

[SensePost – 2011]
Complex Messages

• Contain the same structures as simple messages...

• … But include XML structure:
  – <SVARS>

• Include compressed streams:
  – PARAMS
  – RFC_QUEUE
Complex Messages

• Contain the same messages...
• ... But include XML structures:
  – <SVARS>
• Include compressed streams:
  – PARAMS
  – RFC_QUEUE
Complex Messages

• Contain the same structures as simple messages…

• … But include XML structure:
  – <SVARS>

• Include compressed streams:
  – PARAMS
  – RFC_QUEUE
  – VERBS
Complex Messages

- Contain the same structures as simple messages...
- ... But include XML structure:
  - `<SVARS>`
- Include compressed streams:
  - `PARAMS`
  - `RFC_QUEUE`
  - `VERBS`
Complex Messages

• Contain the same structures as simple messages…
• … But include XML structure:
  – <SVARS>
• Include compressed streams:
  – PARAMS
  – RFC_QUEUE
  – VERBS
  – VARS
Complex Messages

- Contain the same structures as simple messages...
- ... But include XML structure:
  - `<SVARS>`
- Include compressed streams:
  - `PARAM`  
  - `RFC_QUEUE`  
  - `VERBS`  
  - `VARS`
PARAMS, VARS and VERBS

• Work in conjunction
• Define actions to be performed on / by objects on the GUI / Server
• Lists of “indexed data” are decompressed and parsed by ABAP to various fixed-length data structures
PARAMS, VARS and VERBS

- Decompressed, they look as follows:
  - PARAMS:
PARAMS, VARS and VERBS

- Decompressed, they look as follows:
  - PARAMS:
  - VERBS:
PARAMS, VARS and VERBS

- Decompressed, they look as follows:
  - PARAMS:
  - VERBS:
  - VARS:
PARAMS, VARS and VERBS

- Parsed by ABAP into structured variables
- CASE ABAP PARAM-TYP.
  - WHEN ‘S’:
    - Set Value Of
  - WHEN ‘G’:
    - Get Value Of
  - WHEN ‘C’:
    - Call Method Of
- Thoughts of eval() spring to mind…😊
PARAMS, VARS and VERBS

- Graphic example:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>31</td>
<td>20</td>
<td>43</td>
<td>72</td>
<td>65</td>
<td>61</td>
<td>74</td>
<td>65</td>
</tr>
<tr>
<td>4f</td>
<td>62</td>
<td>6a</td>
<td>65</td>
<td>63</td>
<td>74</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

Length: 159 Bytes

1 Create

Object

C 5 C

createObject

C

8 ShellExecute

C

[SensePost – 2011]
PARAMS, VARS and VERBS

- Graphic example:
PARAMS, VARS and VERBS

- Graphic example:

```plaintext
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>31</td>
<td>20</td>
<td>43</td>
<td>72</td>
<td>65</td>
<td>61</td>
<td>74</td>
<td>65</td>
</tr>
<tr>
<td>1</td>
<td>4f</td>
<td>62</td>
<td>6a</td>
<td>65</td>
<td>63</td>
<td>74</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e</td>
</tr>
<tr>
<td>0</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>31</td>
<td>20</td>
<td>43</td>
<td>72</td>
<td>65</td>
<td>61</td>
<td>74</td>
<td>65</td>
</tr>
<tr>
<td>1</td>
<td>4f</td>
<td>62</td>
<td>6a</td>
<td>65</td>
<td>63</td>
<td>74</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e</td>
</tr>
<tr>
<td>0</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>31</td>
<td>20</td>
<td>43</td>
<td>72</td>
<td>65</td>
<td>61</td>
<td>74</td>
<td>65</td>
</tr>
<tr>
<td>1</td>
<td>4f</td>
<td>62</td>
<td>6a</td>
<td>65</td>
<td>63</td>
<td>74</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>
```

Method to call

- Length: 159 Bytes
  - 1 Create Object
- Length: 159 Bytes
  - 1 Create Object
- Length: 159 Bytes
  - 1 Create Object

- `createObject`
PARAMS, VARS and VERBS

• Graphic example:

8th element in VARS structure provides the argument...
PARAMS, VARS and VERBS

• Details on these structures can be found in ABAP code…

• Refer to ABAP Structures && where used:
  – OLE_PA
  – OLE_VERBS
  – SWCBCCONT
RFC_QUEUE

- Contains META and internal table data in use by the current application / screen
- Only ever seems to appear in SAP responses
  - This assumption may be incorrect
RFC_QUEUE

• Contains META and internal table data in use by the current application / screen
• Only ever seems to appear in SAP responses
– This assumption may be incorrect

Length: 46,757 Bytes
00000001
0192.168.1.10
-192.168.1.
00
3
701
701

winxpsap_NSP

00
SAPGUI

QUEUEEEE á ~

KN: "LCNDPN:

Ö ~ BCUSER

EÖ 

EÖ 4 "X "
e

dp_put_client.t

ABLE45A

12935CE05DFBF

1F9A4FO ĖC29DE8

1*TH*
The Fundamentals

- Understand the compression
- Understand the compressed protocol
  - Simplify the sniffing and decompression
- Recompression
- Understand the application protocol
  - What makes SAP Gui tick?
- Identify SAP attack vectors not previously considered…
SAPProx
SAPProx

- SapCompress
  - JNI interface
  - Implements MaxDB functions for decompression & compression
    - int[] doDecompress(int[])
    - Int[] doCompress(int[])
SAPProx

- Java
- Responsible for:
  - Parsing packet data
  - Decompressing messages
  - Interception
  - Compressing modified messages
  - Queue management
Demo: SAPProx
Attack API

- Users can write their own exploits
- In a scripting language of their choice…
  - Jython
  - Groovy
  - Jruby
  - *
- Script locations specified in configuration
- Allow for canned exploits
- (thanks Willem Mouton)
print "START : COMMAND EXEC SCRIPT"
#
# The saved packet
b = [16, 6, 17, 0, 32, 255, 127, 254, 45, 2,
0, 0, 0, 16, 6, 35, 0, 15, 0, 0, 16, 14, 1,
105, 99, 111, 100, 101, 76, 105, 116, 116,
56, 57, 48, 54, 70, 49, 56, 49, 65, 52, 70,
119, 105, 110, 120, 112, 115, 97, 112, 16,
224, 139, 63, 241, 32, 164, 252, 0, 12, 41,
126, 92, 224, 140, 114, 241, 34, 164, 252,
2, 0, 200, 16, 6, 7, 0, 20, 83, 69, 56, 48,
0, 0, 0, 49, 255, 16, 12, 7, 0, 16, 0, 0, 0,
12, 66, 67, 85, 83, 69, 82, 32, 32, 32, 32,
32, 78, 83, 80, 32, 32, 32, 32, 32, 16, 6,
32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32,
32, 32, 32, 32, 32, 32, 32, 32, 16, 6, 13, 0,
32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32,
55, 0, 0, 16, 6, 39, 0, 32, 0, 0, 16, 7, 2,
114, 107, 101, 100, 0, 16, 2, 5, 0, 223, 4,
print " + SET Message:1"
c = api.getStringInput("C:\PATH\COMMAND")
print " + GET Command : " + c
What we’re going to talk about

• Why this Talk?
• The history of decompressing SAP DIAG
• Understanding the fundamentals
• New Attacks
• Conclusion
New (Old) Attacks?

- We now have a proxy for SAP GUI
  - WebScarab for SAP
- For what I believe is the first time, we get an unprecedented view into SAP GUI applications...
- ... and we know where that left us with web applications ...
New (Old) Attacks?

- Authorisation
- Authentication
Demo: Auth*
New (Old) Attacks?

- Authorisation
- Authentication
- State Management
- Business Logic
Demo: State & Business Logic
New (Old) Attacks?

- Authorisation
- Authentication
- State Management
- Business Logic
- Validation
Demo: Validation
New (Old) Attacks?

- Authorisation
- Authentication
- State Management
- Business Logic
- Validation
- Replay
Demo: Replay
New (Old) Attacks?

- Authorisation
- Authentication
- State Management
- Business Logic
- Validation
- Replay
- Client-Side attacks
Client-Side Attacks

- Many business cases require the execution of applications on the client.
  - Provided for by ABAP
- Deprecated: GUI_RUN or WS_EXECUTE
- Current: cl_gui_frontend_services
- Newer clients still support old methods
  - Backwards compatibility
  - Do prompt when applications execute
- (thanks Steve Lord)
Client-Side Attacks

- WS_EXECUTE / GUI_RUN
### Client-Side Attacks

#### WS_EXECUTE / GUI_RUN

<table>
<thead>
<tr>
<th>.0</th>
<th>.1</th>
<th>.2</th>
<th>.3</th>
<th>.4</th>
<th>.5</th>
<th>.6</th>
<th>.7</th>
<th>.8</th>
<th>.9</th>
<th>.a</th>
<th>.b</th>
<th>.c</th>
<th>.d</th>
<th>.e</th>
<th>.f</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>06</td>
<td>0d</td>
<td>00</td>
<td>28</td>
<td>53</td>
<td>41</td>
<td>50</td>
<td>4c</td>
<td>47</td>
<td>52</td>
<td>41</td>
<td>50</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>2c</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>04</td>
<td>30</td>
<td>31</td>
<td>30</td>
<td>30</td>
<td>10</td>
<td>06</td>
<td>23</td>
<td>00</td>
<td>0b</td>
<td>00</td>
<td>00</td>
<td>04</td>
<td>67</td>
<td>01</td>
<td>31</td>
</tr>
<tr>
<td>31</td>
<td>32</td>
<td>37</td>
<td>00</td>
<td>00</td>
<td>10</td>
<td>06</td>
<td>27</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>10</td>
<td>07</td>
<td>02</td>
<td>34</td>
<td>4</td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>33</td>
<td>00</td>
<td>55</td>
<td>6e</td>
<td>69</td>
<td>63</td>
<td>6f</td>
<td>64</td>
<td>65</td>
<td>4c</td>
<td>69</td>
<td>74</td>
<td>74</td>
<td>6c</td>
</tr>
<tr>
<td>65</td>
<td>2e</td>
<td>6b</td>
<td>65</td>
<td>64</td>
<td>00</td>
<td>10</td>
<td>02</td>
<td>05</td>
<td>00</td>
<td>df</td>
<td>04</td>
<td>1V3.4 LE B D,</td>
<td>B</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Text content: (SAPLGRAP 0100 # g 1 127 4 103 UnicodeLittle eUnmarked 0 1V3.4 LE B D,)
- Text content: Xc:\windows\system32\calc.exe
Demo: Client-Side Attacks
Client-Side Attacks

- `cl_gui_frontend_services` – Makes use of OLE
Client-Side Attacks

• `cl_gui_frontend_services`
  – Makes use of OLE
Client-Side Attacks

- SAP GUI provides number of COM libraries with potentially exploitable functions
  - Saved by the fact that the controls are not marked “Safe for Scripting”
Client-Side Attacks

- SAP GUI provides a number of COM libraries with potentially exploitable functions.
  - Saved by the fact that the controls are not marked "Safe for Scripting".
Client-Side Attacks

• SAP GUI provides number of COM libraries with potentially exploitable functions
  – Saved by the fact that the controls are not marked “Safe for Scripting”
• With SAPProxy we can potentially instantiate diverse COM objects
New (Old) Attacks?

- Authorisation
- Authentication
- State Management
- Business Logic
- Validation
- Replay
- Client-Side attacks
- DoS
Demo: DoS
New (Old) Attacks?

- Authorisation
- Authentication
- State Management
- Business Logic
- Validation
- Replay
- Client-Side attacks
- DoS
- *
What we’re going to talk about

• Why this Talk?
• The history of decompressing SAP DIAG
• Understanding the fundamentals
• New Attacks
• Conclusion
Conclusion

• A couple of factors have been common security knowledge for years…
  – Plain-text communication == #fail
  – Security by obscurity == #fail
• We now have a toolset and programmatic interface into SAP DIAG protocol
  – Game Changer
  – Change the way we look at ABAP
  – Happy Haxoring
Conclusion

• SAP provides encryption for client components in the form of Secure Network Communications
  – Provided by 3rd Parties
  – Provided by SAP
• SAP Clients should ensure the use of SNC is enabled and enforced
Questions?

- www.sensepost.com/blog

ian@sensepost.com