

# Oops, I hacked my PBX

## Why auditing proprietary protocols matters

pt

28c3

December 29, 2011

# "The enemy knows the system"

Claude Shannon

- 1 Introduction
- 2 Reverse engineering the protocol
- 3 Actual results
- 4 Conclusion

# A few words in beforehand

- Don't laugh too loud, YOU could have made this mistakes too!
- A real world example is used but slightly obfuscated

# Why did I hack the PBX?

- I didn't want to, seriously!
- Phones with PBX integration can be customized
- Client has >50 of them
- 5 Minutes/Phone to read-modify-write, non scriptable!
- They restructured  $\Rightarrow$  Reconfigure all phones
- $\Rightarrow$  Massive acceptance problems with the admin

# Why did I hack the PBX?

- I didn't want to, seriously!
- Phones with PBX integration can be customized
- Client has >50 of them
- 5 Minutes/Phone to read-modify-write, non scriptable!
- They restructured  $\Rightarrow$  Reconfigure all phones
- $\Rightarrow$  Massive acceptance problems with the admin

# Why did I hack the PBX?

- I didn't want to, seriously!
- Phones with PBX integration can be customized
- Client has >50 of them
- 5 Minutes/Phone to read-modify-write, non scriptable!
- They restructured  $\Rightarrow$  Reconfigure all phones
- $\Rightarrow$  Massive acceptance problems with the admin

# Why did I hack the PBX?

- I didn't want to, seriously!
- Phones with PBX integration can be customized
- Client has >50 of them
- 5 Minutes/Phone to read-modify-write, non scriptable!
- They restructured  $\Rightarrow$  Reconfigure all phones
- $\Rightarrow$  Massive acceptance problems with the admin

# Why did I hack the PBX?

- I didn't want to, seriously!
- Phones with PBX integration can be customized
- Client has >50 of them
- 5 Minutes/Phone to read-modify-write, non scriptable!
- They restructured  $\Rightarrow$  Reconfigure all phones
- $\Rightarrow$  Massive acceptance problems with the admin



# Why did I hack the PBX?

- I didn't want to, seriously!
- Phones with PBX integration can be customized
- Client has >50 of them
- 5 Minutes/Phone to read-modify-write, non scriptable!
- They restructured  $\Rightarrow$  Reconfigure all phones
- $\Rightarrow$  Massive acceptance problems with the admin

# Things you will need

- The original software
- Some PBX hardware to tinker with
- Wireshark
- Your brain
- Too much (client) time on your hands

## Things you will need

- The original software
- Some PBX hardware to tinker with
- Wireshark
- Your brain
- Too much (client) time on your hands



# Things you will need

- The original software
- Some PBX hardware to tinker with
- Wireshark
- Your brain
- Too much (client) time on your hands



# Things you will need

- The original software
- Some PBX hardware to tinker with
- Wireshark
- Your brain
- Too much (client) time on your hands



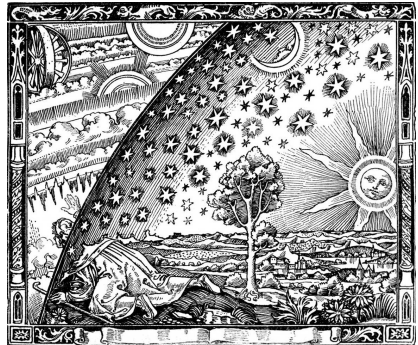
# Things you will need

- The original software
- Some PBX hardware to tinker with
- Wireshark
- Your brain
- Too much (client) time on your hands



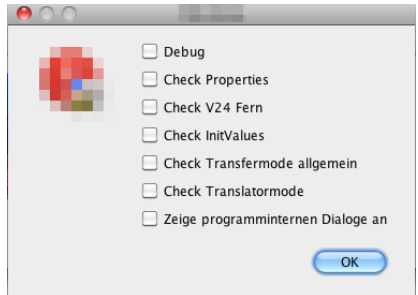
# Understand the original software

- Poke around the interface
- You might find gems ;)
- Try to think behind the GUI



# Understand the original software

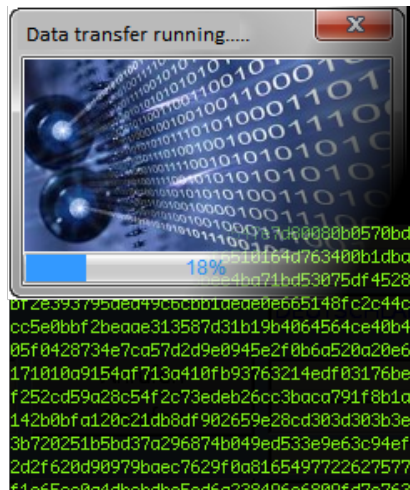
- Poke around the interface
- You might find gems ;)
- Try to think behind the GUI





## Understand the original software

- Poke around the interface
- You might find gems ;)
- Try to think behind the GUI



# Dump the communication

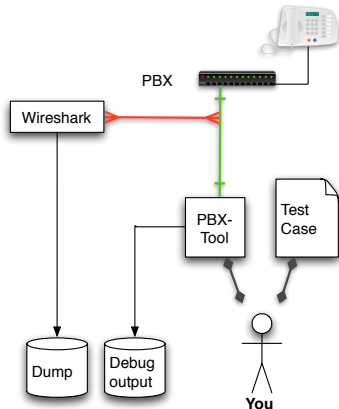
- Define test cases
- Enable debug output
- Repeat test cases while sniffing
- File cleanly

## Simple test case

- 1 Launch Software
- 2 Click on „Load”
- 3 Click on „From phone”
- 4 Select phone
- 5 Enter password
- 6 Watch download

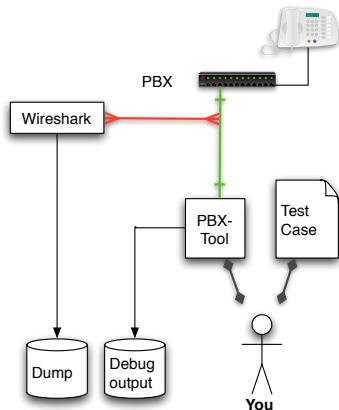
# Dump the communication

- Define test cases
- Enable debug output
- Repeat test cases while sniffing
- File cleanly



# Dump the communication

- Define test cases
- Enable debug output
- Repeat test cases while sniffing
- File cleanly



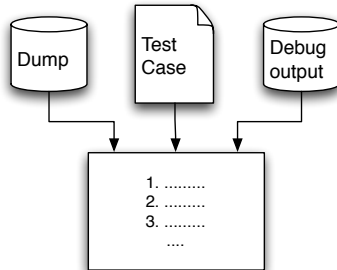
# Dump the communication

- Define test cases
- Enable debug output
- Repeat test cases while sniffing
- File cleanly



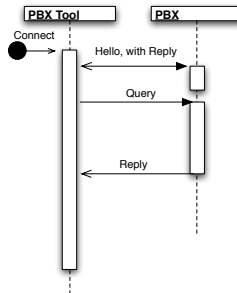
# Analyze your dumps

- Correllate debug data and dump with test case
- Make sense of data flow
- Look at hexdumps
- Look for known data



# Analyze your dumps

- Correlate debug data and dump with test case
- Make sense of data flow
- Look at hexdumps
- Look for known data



# Analyze your dumps

- Correlate debug data and dump with test case
- Make sense of data flow
- Look at hexdumps
- Look for known data

Handwritten annotations on a hex dump:

- A bracket groups the bytes 04 79 01 02 00 at address 00000000.
- An arrow labeled '+1' points from the 01 byte at address 00000000 to the 7a byte at address 00000001.
- A bracket groups the bytes 01 02 00 00 00 a0 24 at address 00000005.
- An arrow labeled '+1' points from the 01 byte at address 00000005 to the 22 byte at address 00000006.

```
00000000  04 79 01 02 00
00000001  04 7a 01 02 02
00000005  09 21 01 02 00 00 00 a0 24 07
00000006  10 22 01 02 02 00 00 a0 24
$. . . . .
00000015  00
```



# Analyze your dumps

- Correlate debug data and dump with test case
- Make sense of data flow
- Look at hexdumps
- Look for known data

Handwritten annotations on a hex dump:

- A bracket above the first line (00000000 04 79 01 02 00) groups the last four bytes (79 01 02 00).
- An arrow labeled "+1" points from the 01 byte of the first line to the 7a byte of the second line (00000000 04 7a 01 02 02).
- A bracket above the second line (00000000 04 7a 01 02 02) groups the last four bytes (7a 01 02 02).
- An arrow labeled "+1" points from the 02 byte of the second line to the 01 byte of the third line (00000005 09 21 01 02 00 00 00 a0 24 07).
- A bracket above the third line (00000005 09 21 01 02 00 00 00 a0 24 07) groups the last eight bytes (01 02 00 00 00 00 a0 24 07).
- The fourth line (00000005 10 22 01 02 02 00 00 a0 24) is preceded by a "\$....." symbol.
- The fifth line (00000015 00) is preceded by a "\$....." symbol.

# Protocol basics

- Header contains packet length
- Each packet to PBX triggers a reponse
- Packet type of a positive ACK is the one of the request +1
- Has virtual channels
- Has an idle timeout!

Client		PBX
05 21 00 00 00 00		05 22 00 00 12 e0
09 21 00 00 00 00 00 60 00 16		1f 22 00 00 12 00 00 60 00 16 40 43 90 14 e0 00 00 00 00 00 00 00 00 00 00 90 00 00 00 00 00 00
08 81 01 01 00 63 02 00 00		07 82 01 01 02 6b 01 7c

# Protocol basics

- Header contains packet length
- Each packet to PBX triggers a reponse
- Packet type of a positive ACK is the one of the request +1
- Has virtual channels
- Has an idle timeout!

Client		PBX
05 21 00 00 00 00	⇒	05 22 00 00 12 e0
09 21 00 00 00 00 00 60 00 16	⇒	1f 22 00 00 12 00 00 60 00 16 40 43 90 14 e0 00 00 00 00 00 00 00 00 00 00 90 00 00 00 00 00 00
08 81 01 01 00 63 02 00 00	⇒	07 82 01 01 02 6b 01 7c

# Protocol basics

- Header contains packet length
- Each packet to PBX triggers a reponse
- Packet type of a positive ACK is the one of the request +1
- Has virtual channels
- Has an idle timeout!

Client		PBX
05 21 00 00 00 00		05 22 00 00 12 e0
09 21 00 00 00 00 00 60 00 16		1f 22 00 00 12 00 00 60 00 16 40 43 90 14 e0 00 00 00 00 00 00 00 00 00 00 90 00 00 00 00 00 00
08 81 01 01 00 63 02 00 00		07 82 01 01 02 6b 01 7c

# Protocol basics

- Header contains packet length
- Each packet to PBX triggers a reponse
- Packet type of a positive ACK is the one of the request +1
- Has virtual channels
- Has an idle timeout!

Client		PBX
05 21 00 00 00 00		05 22 00 00 12 e0
09 21 00 00 00 00 00 60 00 16		1f 22 00 00 12 00 00 60 00 16 40 43 90 14 e0 00 00 00 00 00 00 00 00 00 00 90 00 00 00 00 00 00
08 81 01 01 00 63 02 00 00		07 82 01 01 02 6b 01 7c

# Protocol basics

- Header contains packet length
- Each packet to PBX triggers a reponse
- Packet type of a positive ACK is the one of the request +1
- Has virtual channels
- Has an idle timeout!

Client		PBX
05 21 00 00 00 00		05 22 00 00 12 e0
09 21 00 00 00 00 00 60 00 16		1f 22 00 00 12 00 00 60 00 16 40 43 90 14 e0 00 00 00 00 00 00 00 00 00 00 90 00 00 00 00 00 00
08 81 01 01 00 63 02 00 00		07 82 01 01 02 6b 01 7c

# Communication flow

- 1 Find out packet types
- 2 Explore the communication sequence
- 3 Find the authentication sequence.

Name	Value
HELLO	0x21
READ_NVRAM	0x31
WRITE_NVRAM	0x33
CHAN_OPEN	0x81
CHAN_CLOSE	0x85
INQUIRE_HARDWARE	0x87
PING	0x79

## Note

These apply to all devices in the system, Phones and PBXe

# Communication flow

- 1 Find out packet types
- 2 Explore the communication sequence
- 3 Find the authentication sequence.

Count	Chan	Type
1x	0	HELLO
1x	0	READ_NVRAM
1x	1	CHAN_OPEN
20x	1	INQUIRE_HW
1x	1	READ_NVRAM
1x	1	INQUIRE_HW
1x	1	PING
1x	2	CHAN_OPEN
1x	2	INQUIRE_HW
<i>nx</i>	2	READ_NVRAM
1x	2	CHAN_CLOSE
1x	1	CHAN_CLOSE



# Communication flow

- 1 Find out packet types
- 2 Explore the communication sequence
- 3 Find the authentication sequence.

Count	Chan	Type
1x	0	HELLO
1x	0	READ_NVRAM
1x	1	CHAN_OPEN
20x	1	INQUIRE_HW
1x	1	READ_NVRAM
1x	1	INQUIRE_HW
1x	1	PING
1x	2	CHAN_OPEN
1x	2	INQUIRE_HW
<i>nx</i>	2	READ_NVRAM
1x	2	CHAN_CLOSE
1x	1	CHAN_CLOSE

# Communication flow

- 1 Find out packet types
- 2 Explore the communication sequence
- 3 Find the authentication sequence.

Wait: No auth done?!

Count	Chan	Type
1x	0	HELLO
1x	0	READ_NVRAM
1x	1	CHAN_OPEN
20x	1	INQUIRE_HW
1x	1	READ_NVRAM
1x	1	INQUIRE_HW
1x	1	PING
1x	2	CHAN_OPEN
1x	2	INQUIRE_HW
<i>nx</i>	2	READ_NVRAM
1x	2	CHAN_CLOSE
1x	1	CHAN_CLOSE

# Where is the authentication gone?

- ❶ Launch Software
- ❷ Click on "Load"
- ❸ Click on "From phone"
- ❹ Select phone
- ❺ Enter password ("012345")
- ❻ Watch download

Step	Count	Chan	Type
3	1x	0	HELLO
	1x	0	READ_NVRAM
	1x	1	CHAN_OPEN
	20x	1	INQUIRE_HW
4	1x	1	READ_NVRAM
	1x	1	INQUIRE_HW
5	1x	1	PING
6	1x	2	CHAN_OPEN
	1x	2	INQUIRE_HW
	<i>nx</i>	2	READ_NVRAM
	1x	2	CHAN_CLOSE
	1x	1	CHAN_CLOSE

# Where is the authentication gone?

- 1 Launch Software
- 2 Click on "Load"
- 3 Click on "From phone"
- 4 Select phone
- 5 Enter password ("012345")
- 6 Watch download

Step	Count	Chan	Type
3	1x	0	HELLO
	1x	0	READ_NVRAM
	1x	1	CHAN_OPEN
	20x	1	INQUIRE_HW
4	1x	1	READ_NVRAM
	1x	1	INQUIRE_HW
5	1x	1	PING
6	1x	2	CHAN_OPEN
	1x	2	INQUIRE_HW
	<i>nx</i>	2	READ_NVRAM
	1x	2	CHAN_CLOSE
	1x	1	CHAN_CLOSE

# Where is the authentication gone?

- 1 Launch Software
- 2 Click on "Load"
- 3 Click on "From phone"
- 4 Select phone
- 5 Enter password ("012345")
- 6 Watch download

Step	Count	Chan	Type
⇒3	1x	0	HELLO
	1x	0	READ_NVRAM
	1x	1	CHAN_OPEN
	20x	1	INQUIRE_HW
4	1x	1	READ_NVRAM
	1x	1	INQUIRE_HW
5	1x	1	PING
6	1x	2	CHAN_OPEN
	1x	2	INQUIRE_HW
	<i>nx</i>	2	READ_NVRAM
	1x	2	CHAN_CLOSE
	1x	1	CHAN_CLOSE

# Where is the authentication gone?

- 1 Launch Software
- 2 Click on "Load"
- 3 Click on "From phone"
- 4 **Select phone**
- 5 Enter password ("012345")
- 6 Watch download

Step	Count	Chan	Type
3	1x	0	HELLO
	1x	0	READ_NVRAM
	1x	1	CHAN_OPEN
	20x	1	INQUIRE_HW
⇒4	1x	1	READ_NVRAM
	1x	1	INQUIRE_HW
5	1x	1	PING
6	1x	2	CHAN_OPEN
	1x	2	INQUIRE_HW
	<i>nx</i>	2	READ_NVRAM
	1x	2	CHAN_CLOSE
	1x	1	CHAN_CLOSE

# Where is the authentication gone?

- ① Launch Software
- ② Click on "Load"
- ③ Click on "From phone"
- ④ Select phone
- ⑤ Enter password ("012345")
- ⑥ Watch download

Step	Count	Chan	Type
3	1x	0	HELLO
	1x	0	READ_NVRAM
	1x	1	CHAN_OPEN
	20x	1	INQUIRE_HW
4	1x	1	READ_NVRAM
	1x	1	INQUIRE_HW
⇒5	1x	1	PING
6	1x	2	CHAN_OPEN
	1x	2	INQUIRE_HW
	<i>nx</i>	2	READ_NVRAM
	1x	2	CHAN_CLOSE
	1x	1	CHAN_CLOSE

# Where is the authentication gone?

- ① Launch Software
- ② Click on "Load"
- ③ Click on "From phone"
- ④ Select phone
- ⑤ Enter password ("012345")
- ⑥ Watch download

Step	Count	Chan	Type
3	1x	0	HELLO
	1x	0	READ_NVRAM
	1x	1	CHAN_OPEN
	20x	1	INQUIRE_HW
4	1x	1	READ_NVRAM
	1x	1	INQUIRE_HW
5	1x	1	PING
⇒6	1x	2	CHAN_OPEN
	1x	2	INQUIRE_HW
	<i>nx</i>	2	READ_NVRAM
	1x	2	CHAN_CLOSE
	1x	1	CHAN_CLOSE



# Where is the authentication gone?

- A short READ\_NVRAM
- Reads some binary gibberish
- Original software shows an auth-window
- Or was it...

# Where is the authentication gone?

- A short READ\_NVRAM
- Reads some binary gibberish
- Original software shows an auth-window
- Or was it...

*Client :*    09   31   01   02   00 00 00 A0 24   06  
              Len  Type Chan               Addr   Len  
*PBX :*       10   32   01   02   02 00 00 A0 24   06   86 87 84 85 82 83  
              Len  Type Chan               Addr   Len       Surprise

# Where is the authentication gone?

- A short READ\_NVRAM
- Reads some binary gibberish
- Original software shows an auth-window
- Or was it...

*Client :*    09   31   01   02   00 00 00 A0 24   06  
              Len   Type   Chan                      Addr            Len

*PBX :*       10   32   01   02   02 00 00 A0 24   06   86 87 84 85 82 83  
              Len   Type   Chan                      Addr            Len            Surprise

# Where is the authentication gone?

- A short READ\_NVRAM
- Reads some binary gibberish
- Original software shows an auth-window
- Or was it...

*Client :*    09   31   01   02   00 00 00 A0 24   06  
              Len  Type Chan                    Addr        Len

*PBX :*       10   32   01   02   02 00 00 A0 24   06   86 87 84 85 82 83  
              Len  Type Chan                    Addr        Len        Surprise

# Where is the authentication gone?

- A short READ\_NVRAM
- Reads some binary gibberish
- Original software shows an auth-window
- Or was it...

Client : 09 31 01 02 00 00 00 A0 24 06  
           Len Type Chan                      Addr            Len

PBX : 10 32 01 02 02 00 00 A0 24 06 86 87 84 85 82 83  
           Len Type Chan                      Addr            Len            Surprise

	86	87	84	85	82	83
XOR	B6	B6	B6	B6	B6	B6
=	30	31	32	33	34	35



## The story so far – But how could it happen?

- Authentication neither necessary nor useful
- No privilege system implemented
- Many commands useful for debugging
- $\Rightarrow$  Maybe a developers interface?

## The story so far – But how could it happen?

- Authentication neither necessary nor useful
- No privilege system implemented
- Many commands useful for debugging
- $\Rightarrow$  Maybe a developers interface?



## The story so far – But how could it happen?

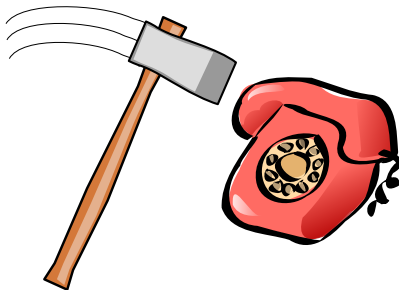
- Authentication neither necessary nor useful
- No privilege system implemented
- Many commands useful for debugging
- $\Rightarrow$  Maybe a developers interface?

## The story so far – But how could it happen?

- Authentication neither necessary nor useful
- No privilege system implemented
- Many commands useful for debugging
- $\Rightarrow$  Maybe a developers interface?

# What could happen..

- Read/Write any phone
- Reset PBX password
- Really bad stuff:  
Read/Write  
firmware



# What could happen..

- Read/Write any phone
- Reset PBX password
- Really bad stuff:  
Read/Write  
firmware

# What could happen..

- Read/Write any phone
- Reset PBX password
- Really bad stuff:  
Read/Write  
firmware



# And now?

- Contact the vendor
- Be nice, they will be too!
- Help them improve!
- Carry on and find more bugs!

## And now?

- Contact the vendor
- Be nice, they will be too!
- Help them improve!
- Carry on and find more bugs!

## And now?

- Contact the vendor
- Be nice, they will be too!
- Help them improve!
- Carry on and find more bugs!



## And now?

- Contact the vendor
- Be nice, they will be too!
- Help them improve!
- Carry on and find more bugs!

## Lessons learned

- Anchor authentication/encryption in protocol
- Do not use debugging interfaces in production
- Audit your codebase once in a while
- Shannon was right..
- Vendors are happy to be informed (at least good ones)

## Lessons learned

- Anchor authentication/encryption in protocol
- Do not use debugging interfaces in production
- Audit your codebase once in a while
- Shannon was right..
- Vendors are happy to be informed (at least good ones)

## Lessons learned

- Anchor authentication/encryption in protocol
- Do not use debugging interfaces in production
- Audit your codebase once in a while
- Shannon was right..
- Vendors are happy to be informed (at least good ones)

## Lessons learned

- Anchor authentication/encryption in protocol
- Do not use debugging interfaces in production
- Audit your codebase once in a while
- Shannon was right..
- Vendors are happy to be informed (at least good ones)

## Lessons learned

- Anchor authentication/encryption in protocol
- Do not use debugging interfaces in production
- Audit your codebase once in a while
- Shannon was right..
- Vendors are happy to be informed (at least good ones)

# Thank you for your attention

Any Questions?