Test Plan

Skullduggery

Greg Baker - Robert Koeninger

Advisor: Prof. Wilsey

Feb 22, 2010

Test Plan Overview

The testing environment will consist of two virtual phones using the Android Debug Bridge. Two virtual phones will be launched using a Skullduggery modified copy of Android. A call will be made from one phone to the other, during which the Skullduggery code will be run and the data will be encrypted as it is transferred over the virtual cell “network.” Once the call has ended, the radio debug data is dumped from the emulation environment and can be examined. A group of control calls will also be made without Skullduggery so normal GSM-encoded call voice data can be collected.

The debug data will contain GSM or GSM + Skullduggery encoded voice data that needs to be extracted from the raw debug file as it is surrounded by packet headers. Once the data has been extracted, the open source GSM decoding software from GNU Radio Project will be used to decode the calls into voice data. The voice data is then imported into a media player and heard by a tester. The voice data from the control call should be perfectly audible but the Skullduggery enabled test call will still be encrypted and will be heard as noise.

Test Case Description

1. AES Encryption - Validate AES (unit, black-box)

    1.1. Validate AES library against standard Java Library

    1.2. Encrypt several different strings with a known valid AES library.

    1.3. Example inputs:

        1.3.1. '\0\0\0...\0'

        1.3.2. 'abcd....'

        1.3.3. 'abcd...\n'

        1.3.4. 'ab\0d...\n'

    1.4. Outputs should match between our AES (Crypto++) and Java AES

    1.5. Abnormal test cases should be spotted with null input, eol inputs.

2. AES Decryption - Validate AES (unit, black-box)

    2.1. Validate AES library against standard Java Library

    2.2. Encrypt several different strings with a known valid AES library.

    2.3. Example inputs:

        2.3.1. AES('\0\0\0...\0')

        2.3.2. AES('abcd....')

        2.3.3. AES('abcd...\n')

        2.3.4. AES('ab\0d...\n')

    2.4. Outputs should match between our AES (Crypto++) and Java AES

    2.5. Abnormal test cases should be spotted with null input, eol inputs.

3. Playing of voice data (integration, white-box)

    3.1. Ensure that we're able to play voice data on the speaker

    3.2. Play arbitrary voice data

    3.3. Example inputs:

        3.3.1. single tone

        3.3.2. two-tone

        3.3.3. square wave

    3.4. Output of the test should be what we wanted to output (tone, waves, etc)

    3.5. Abnormal test cases include static, no sound, errors, crashes, etc.

4. Capturing of voice packets (integration, white-box)

    4.1. Ensure that we're catching the voice data

    4.2. Capture voice packets, play them back immediately

    4.3. Example inputs:

        4.3.1. "This is my voice"

    4.4. Outputs should be the matching voice phrase

    4.5. Abnormal test cases include mangling of voice, no sound, loud sound

5. Encryption of voice packets (integration, white-box)

    5.1. Ensure that we're encrypting the voice data

    5.2. Capture voice packets, encrypt voice packets, output encrypted, raw voice

    5.3. Example inputs:

        5.3.1. "This is my encrypted voice"

    5.4. Outputs should be encrypted voice followed by raw voice

    5.5. Abnormal test cases include no mangling of voice, no sound, loud sound

6. Decryption of voice packets (integration, white-box)

    6.1. Ensure that we're encrypting the voice data

    6.2. Capture voice packets, encrypt voice packets, output encrypted, raw voice

    6.3. Example inputs:

        6.3.1. "This is my decrypted voice"

6.4. Outputs should be encrypted voice followed by decrypted voice

6.5. Abnormal test cases include mangling of voice, no sound, loud sound

7. Test that key is sent (integration, black-box)

7.1. Ensure that the key we're using to encrypt the voice data is used in a dialed call

7.2. Capture data packets, check for the key

7.3. Example Inputs:

7.3.1. AES Key (128b), Public key (1024b)

7.4. Output should encrypt the AES key with the private key, decrypt-able with public key.

7.5. Abnormal test cases include sending raw AES key, all-0 bytes, invalid encryption, etc.

8. Test that key is received (integration, black-box)

8.1. Ensure that the key we're using to encrypt the voice data is received in a received call

8.2. Examine the state of the phone while receiving a call

8.3. Example inputs:

8.3.1. Sent data over line

8.3.1.1. Should include encrypted AES

8.3.1.2. Should include RSA Public key

8.4. Output should be the correct AES key sent

8.5. Abnormal test cases include receiving the wrong AES key, using all-0 bytes or non-initialized bytes, etc.

9. Test that key is used (integration, white-box)

9.1. Ensure that the key we receive is used correctly.

9.2. Examine the state of the phone while receiving a call

9.3. Example inputs:

9.3.1. Sent data over line (as 8.3.1)

9.3.2. Encrypted voice

9.4. Output should be unencrypted voice

9.5. Abnormal test cases include not-receiving the correct key, etc.

10. Test that key is stored (integration, white-box)

    10.1. Ensure that the key we receive is stored between packets.

    10.2. Hold an extended phone call (> 5 minutes)

    10.3. Example inputs:

        10.3.1. Sent data over line (as 8.3.1)

        10.3.2. Encrypted voice

    10.4. Output should be unencrypted voice.

    10.5. Abnormal test cases include dropped packets, cut-out voice, abrupt call-end, etc.

11. Test a encryption to encryption call (integration, black-box)

    11.1. Ensure that it all works

    11.2. Hold an extended phone call (> 5 minutes)

    11.3. Example inputs:

        11.3.1. Sent data over line (as 8.3.1)

        11.3.2. Encrypted voice

    11.4. Output should be unencrypted voice.

    11.5. Abnormal test cases include dropped packets, cut-out voice, abrupt call-end, etc.

12. Test a encryption to sans call (integration, black-box)

    12.1. Ensure that the call is unencrypted

    12.2. Hold an extended phone call (> 5 minutes)

    12.3. Example inputs:

        12.3.1. Sent data over line (as 8.3.1)

        12.3.2. "Vanilla" phone

12.4. Outputs include:

12.4.1. Unencrypted voice

12.4.2. Warning on the encryption-enabled phone.

12.5. Abnormal test cases include dropped packets, cut-out voice, abrupt call-end, etc.

13. Test a sans to sans call (integration, black-box)

13.1. Ensure that the call is unencrypted with Skullduggery off.

13.2. Hold an extended phone call (> 5 minutes)

13.3. Example inputs:

13.3.1. Sent data over line (as 8.3.1)

13.3.2. Unencrypted voice

13.4. Output should be unencrypted voice

13.5. Abnormal test cases include dropped packets, cut-out voice, abrupt call-end, etc.

14. Test a sans to encryption call (integration, black-box)

14.1. Ensure that the call is unencrypted

14.2. Hold an extended phone call (> 5 minutes)

14.3. Example inputs:

14.3.1. Sent data over line (as 8.3.1)

14.4. Output should include:

14.4.1. Unencrypted voice

14.4.2. Warning on encryption-enabled phone.

14.5. Abnormal test cases do not include dropped packets, cut-out voice, abrupt call-end, etc.

15. Test public-key generation (unit, black-box)

15.1. Ensure that public-key generation works

15.2. Generate a public key for use in transmitting session-keys

15.3. Example inputs:

15.3.1. Key generation "entropy"

15.4. Output should be a valid public/private key pair.

15.5. Abnormal test cases include a key pair which does not encrypt, decrypt.

16. Test public-key usage (integration, white-box)

16.1. Ensure that the key we're using to encrypt the AES key is used

16.2. Capture data packets, check for Public key, encrypted AES key

16.3. Example Inputs:

16.3.1. AES Key (128b), Public key (1024b)

16.4. Outputs include:

16.4.1. Receiver should send public key

16.4.2. Caller should send AES key, encrypted with public key

16.5. Abnormal test cases include sending raw AES key, all-0 bytes, invalid encryption, encryption with incorrect key, etc.

Test Case Matrix

Test ID	Normal Abnormal Boundary	Blackbox White-box	Functional Performance	Unit Integration
1	AES = Java AES AES != Java AES	Blackbox	Functional	Unit
2	AES = Java AES AES != Java AES	Blackbox	Functional	Unit
3	Voice Plays No Voice	White-box	Functional	Integration
4	Voice plays No voice	White-box	Functional	Integration
5	Encrypted Voice No Encrypted Voice	White-box	Functional	Integration
6	Decrypted Voice No Decrypted Voice	White-box	Functional	Integration
7	AES Key No AES Key	Blackbox	Functional	Integration
8	AES Key No AES Key	Blackbox	Functional	Integration
9	Unencrypted Voice No Unencrypted voice	White-box	Functional	Integration
10	Unencrypted Voice	White-box	Functional	Integration

	No unencrypted voice			
11	Unencrypted Voice No unencrypted voice	Blackbox	Functional	Integration
12	Unencrypted voice, warning No unencrypted voice, no warning	Blackbox	Functional	Integration
13	Unencrypted voice, warning No unencrypted voice, no warning	Blackbox	Functional	Integration
14	Unencrypted voice, Warning No unencrypted voice, no warning	Blackbox	Functional	Integration
15	Public/private key pair No Key Pair	Blackbox	Functional	Unit
16	Key sent Key not sent	White-box	Functional	Integration