Password Attack on Kerberos V and Windows 2000

Last updated: 9th of May 2003

The vulnerability was analyzed by:

- Kimmo Kasslin, kimmo.kasslin@hut.fi
- Antti Tikkanen, antti.tikkanen@hut.fi

Threat and Vulnerability

Kerberos V authentication protocol is described in more detail in [1]. The Windows 2000 implementation of Kerberos V protocol requires the use of the pre-authentication data in the KRB_AS_REQ message by default, which makes it harder to implement offline password attacks. If pre-authentication is not used, anyone can make a request for a TGT and launch an offline password attack against it. The default implementation of pre-authentication data consists of an encrypted timestamp and a cryptographic checksum created with a key derived from the user's password.

If an attacker is able to listen to the network traffic between the victim machine and the KDC server, a password attack becomes possible. This is based on the fact that the timestamp in the pre-authentication data is an ASCII-encoded string with the syntax YYYYMMDDHHMMSSZ before encryption. This information makes it possible to find a valid password by running a dictionary or brute force attack against it. The correctness of the result can be verified by calculating the checksum.

Similar password attack vulnerabilities are described in [2]. A theoretical study on this matter has already been conducted and is available from [3].

There is already a similar tool with only basic functionality available from http://ntsecurity.nu/toolbox/kerbcrack/. However the source code is not available which is a major shortcoming.

Preconditions for the Attack

The following assumptions will be made:

- The victim wishing to authenticate himself will be using a Windows 2000 Professional workstation.
- The KDC will be running on a Windows 2000 server.
- The attacker must be able to listen to the network traffic between the victim and the KDC.

The last condition will be accomplished by ARP spoofing which is described in [4].

Attack Environment Used

Our environment will consist of:

- Windows 2000 Professional SP3 (the victim)
- Windows 2000 Server SP3 (KDC)
- Red Hat Linux 8.0 (the attacker)

The network is switched. The client and attacker will be in the same logical network segment and the KDC server in another segment.

The servers will have basic configurations regarding Kerberos. No encryption will be used in the network layer (IPSEC).

Analysis of the Attack

The attack was conducted by listening to the network in promiscuous mode after we had successfully ARP spoofed the victim. We decided to use ethereal [5] to capture the packets because of its ability to dissemble the ASN.1 encoded Kerberos packets.

RFC 1510 [1] specifies the exact format of the KRB_AS_REQ message. The message fields of all relevant parts are:

```
KDC-REQ ::=        SEQUENCE {
           pvno[1]            INTEGER,
           msg-type[2]        INTEGER,
           padata[3]          SEQUENCE OF PA-DATA OPTIONAL,
           req-body[4]        KDC-REQ-BODY
}

PA-DATA ::=        SEQUENCE {
           padata-type[1]     INTEGER,
           padata-value[2]    OCTET STRING,
}
```

Below we have the packet decoded by ethereal:

```
Frame 1 (332 bytes on wire, 332 bytes captured)
Ethernet II, Src: 00:90:27:d6:59:ee, Dst: 00:03:47:92:88:f0
Internet Protocol, Src Addr: 193.167.5.29 (193.167.5.29), Dst Addr: 193.167.4.1
(193.167.4.1)
User Datagram Protocol, Src Port: 3013 (3013), Dst Port: kerberos (88)
Kerberos
    Version: 5
    MSG Type: AS-REQ
    Pre-Authentication
        Type: PA-ENC-TIMESTAMP
        Value: 303DA003020117A236043475E8111005...
        Type: PA-PAC-REQUEST
        Value: 3005A0030101FF
    Request
    Addresses

0000  00 03 47 92 88 f0 00 90 27 d6 59 ee 08 00 45 00   ..G.....'.Y...E.
0010  01 3e 44 10 00 00 80 11 69 32 c1 a7 05 1d c1 a7   .>D.....i2......
0020  04 01 0b c5 00 58 01 2a e8 7e 6a 82 01 1e 30 82   .....X.*.~j...0.
0030  01 1a a1 03 02 01 05 a2 03 02 01 0a a3 5f 30 5d   .............._0]
0040  30 48 a1 03 02 01 02 a2 41 04 3f 30 3d a0 03 02   0H......A.?0=...
0050  01 17 a2 36 04 34 75 e8 11 10 05 28 0b 58 8f 59   ...6.4u....(.X.Y
0060  25 84 6d 2b a7 32 c5 df 85 a8 f5 42 63 81 ab f5   %.m+.2.....Bc...
0070  53 bf f7 6b df b9 81 e0 5e db 3d 81 5f b6 bf 52   S..k....^.=._..R
0080  33 50 81 71 f4 4c da e6 23 5b 30 11 a1 04 02 02   3P.q.L..#[0.....
0090  00 80 a2 09 04 07 30 05 a0 03 01 01 ff a4 81 ac   ......0.........
```

```
00a0  30 81 a9 a0 07 03 05 00 40 81 00 10 a1 15 30 13   0.......@.....0.
00b0  a0 03 02 01 01 a1 0c 30 0a 1b 08 61 75 74 69 6b   .......0...autik
00c0  6b 61 6e a2 05 1b 03 57 49 4e a3 18 30 16 a0 03   kan....WIN..0...
00d0  02 01 02 a1 0f 30 0d 1b 06 6b 72 62 74 67 74 1b   .....0...krbtgt.
00e0  03 57 49 4e a5 11 18 0f 32 30 33 37 30 39 31 33   .WIN....20370913
00f0  30 32 34 38 30 35 5a a6 11 18 0f 32 30 33 37 30   024805Z....20370
0100  39 31 33 30 32 34 38 30 35 5a a7 06 02 04 64 c8   913024805Z....d.
0110  b4 3a a8 19 30 17 02 01 17 02 02 ff 7b 02 01 80   .:..0.......{...
0120  02 01 03 02 01 01 02 01 18 02 02 ff 79 a9 1d 30   ............y..0
0130  1b 30 19 a0 03 02 01 14 a1 12 04 10 54 45 4d 50   .0..........TEMP
0140  45 53 54 20 20 20 20 20 20 20 20 20               EST
```

We are interested in the pair:

```
Type: PA-ENC-TIMESTAMP
Value: 303DA003020117A236043475E8111005...
```

For some unknown reason ethereal does not fully decode the pre-authentication data. It is important to notice that the *Value*-field is still ASN.1 encoded. The full byte sequence of the *Value*-field is:

```
303da003020117a236043475e8111005280b588f5925846d2ba732c5df85a8f5426381a
bf553bff76bdfb981e05edb3d815fb6bf5233508171f44cdae6235b
```

We will use the HEX to ASN.1 Decoder [6] to get the following structure:

```
Decoding:
303da003020117a236043475e8111005280b588f5925846d2ba732c5df85a8f5426381a
bf553bff76bdfb981e05edb3d815fb6bf5233508171f44cdae6235b
Block: 0xa003020117
   Integer: 23
Block:
0xa236043475e8111005280b588f5925846d2ba732c5df85a8f5426381abf553bff76bd
fb981e05edb3d815fb6bf5233508171f44cdae6235b
   StringHex:
75e8111005280b588f5925846d2ba732c5df85a8f5426381abf553bff76bdfb981e05ed
b3d815fb6bf5233508171f44cdae6235b
```

From [7] we can verify that the integer 23 defines the encryption type RC4-HMAC. The encrypted timestamp that we have been looking for is the 52-bytes long block:

```
75e8111005280b588f5925846d2ba732c5df85a8f5426381abf553bff76bdfb981e05ed
b3d815fb6bf5233508171f44cdae6235b
```

The decryption operation of the encrypted timestamp is described in more detail in [7].

To make the attack a little bit easier to perform we created two programs: sniff_krb5_asreq_packet.c [8] and crack_krb5_preauth_data.c [9].

The first program listens to the network traffic in promiscuous mode and automatically extracts the encrypted timestamp from the pre-authentication data. The second program runs a dictionary attack against the data collected with the first tool.

Detection and tracing

This attack is accomplished by passively listening to the network traffic between the client and the KDC server. The only way to detect this is by monitoring the network for symptoms which might give us a hint that someone is running a sniffer on the network. More information can be found from [4].

Protection against the Attack

This attack will become computationally infeasible if a strong password policy is implemented. The Windows 2000 implementation of Kerberos V supports also another pre-authentication method in addition to the password-based. This public key based scheme, called PKINIT [10], does not suffer from the weakness we are describing here. Another effective way to prevent this attack is to encrypt the network traffic, for example by using IPSEC.

Test Results

The attack was conducted successfully. We were able to collect pre-authentication data from the switched network and perform a limited dictionary attack against these accounts. We did not have enough resources to generate a dictionary file extensive enough to perform an attack against strong passwords.

References

[1] J. Kohl, C. Neuman. The Kerberos Network Authentication Service (V5). Request for Comments 1510, September 1993.

[2] S. M. Bellovin, M. Merrit. Limitations of the Kerberos Authentication System. http://www.research.att.com/~smb/papers/kerblimit.usenix.pdf. Referenced 3.2.2003.

[3] F. O'Dwyer. Feasibility of attacking Windows 2000 Kerberos Passwords. http://www.brd.ie/papers/w2kkrb/feasibility_of_w2k_kerberos_attack.htm. Referenced 11.3.2003.

[4] K. Kasslin, A. Tikkanen. Hijacking a Network Connection on a Switched Network.

[5] Ethereal. http://www.ethereal.com. Referenced 11.3.2003.

[6] HEX to ASN.1 Decoder. http://kairos.dsa.uqam.ca/software/asndecode.html. Referenced 15.3.2003.

[7] M. Swift, J. Brezak. The Microsoft Windows 2000 RC4-HMAC Kerberos encryption type. Internet Draft draft-brezak-win2k-krb-rc4-hmac-04.txt, May 2002.

[8] K. Kasslin, A. Tikkanen. sniff_krb5_asreq_packet.c.

[9] K. Kasslin, A. Tikkanen. crack_krb5_preauth_data.c.

[10] B. Tung, C. Neuman, M. Hur, A. Medvinsky, S. Medvinsky, J. Wray, J. Trostle. Public Key Cryptography for Initial Authentication in Kerberos. Internet Draft draft-ietf-cat-kerberos-pk-init-16.txt, March 2002.