All your private keys are belong to us

프로세스 메모리에서 RSA 개인키와 인증서 추출하기

Tobias Klein
tk@trapkit.de
Version 1.0, 2006/02/05.

번역 : P3tEr(www.wowhacker.org)

Abstract

본 문서는 신뢰성 있는 방법으로 프로세스 메모리로부터 RSA 개인키와 인증서를 찾아 추출하는 방법을 논의한다. 이 방법은 민감한 암호화된 자료를 가져오는데 이용되어질 수 있다. 개념 증명으로서 익스플로잇 페이로드와 동일한 IDA Pro 플러그인이 논의될 것이다.

1 Overview

본 문서는 매우 신뢰성 있는 방법으로서 프로세스 메모리의 밖으로 RSA 개인키와 인증서를 찾고 추출하기 위한 방법을 논의한다. 공격자는 이 방법을 사용하여 민감한 암호화된 자료를 가져올 수 있다. 개념 증명으로서 익스플로잇 페이로드와 마찬가지인 IDA Pro 플러그인이 설명될 것이다.

다음과 같은 시나리오를 생각해보자:

- 공격자는 웹 어플리케이션의 취약점이나 웹 서버 자체의 취약점을 통해 웹 서버에 접근할 수 있다. 그 결과, 공격자는 웹 서버 프로세스와 같은 보안 문맥상에 있게 된다. 공격자의 성공은 인증서와 동일한 SSL 개인키를 훔쳐오는 것이다. 공격자가 웹 서버 프로세스의 권한이 없는 문맥에 있을 때에는 공격자는 파일시스템의 접근 제한 때문에 이 정보에 접근 할 수 없다.

- 공격자는 웹 서버 시스템에 권한된 접근을 얻는다. 개인키는 passphrase에 의해 보호되어있다. 공격자의 성공은 cleartext에서 SSL 개인키를 훔쳐오는 것이다.

이러한 문제를 해결하기 위한 한가지 방법은 웹서버 프로세스 메모리의 밖으로 인증서와 마찬가지인 개인 키를 추출하는 것이다. 다음에 그 해결방법을 설명하고 있다.

2 메모리에서 개인키 와 인증서 찾기

이미 시스템 메모리[1]내의 암호화된 자료를 찾기 위한 다른 방법이 설명된 문서들이 있다. 본 문서에서 제안된 방법은 기존의 방법과 완전히 다른 접근이며, 따라서 Shamir et al의 업적과는 무관하다.

인증서와 마찬가지로 RSA 개인 키도 공통적으로 표준 포맷으로 표현된다.
RSA 개인키 정보의 문법은 PKCS #8 [2], SSL 인증서의 문법은 x509 v3 [3]에서 기술된다.
인증서 문법과 마찬가지로 개인키도 ASN.1로 표현된다.

PKCS #8: 개인키 문법 표준(Private-Key Information Syntax Standard)

개인키는 다음 문법을 따른다.(ASN.1):

```
PrivateKeyInfo ::= SEQUENCE {
version Version,
[…]
```

다음은 ASN.1 문법의 16진수형태로 나타낸 것이다.:

```
30 82 ?? ?? - SEQUENCE (30 82), length of the SEQUENCE (?? ??)
02 01 00 - integer (02), length (01), value (00)
```

모든 개인키는 이 문법 형태로 나타내어지며, 우리는 이러한 패턴을 찾아야 한다.

인터넷 X.509 공개키 기반 인증과 인증서 해제목록(Certificate Revocation List) Profile

인증서는 다음 문법을 따른다.(ASN.1):

SEQUENCE {
SEQUENCE {
[…]

다음은 ASN.1 문법의 16진수형태로 나타낸 것이다.:
:
30 82 ?? ?? - SEQUENCE (30 82), length of the SEQUENCE (?? ??)
30 82 ?? ?? - SEQUENCE (30 82), length of the SEQUENCE (?? ??)

모든 인증서는 이러한 문맥으로 나타내어지며 우리는 이러한 패턴을 찾으면 된다.

개인 키와 인증서를 찾기 위해서는 단지 이러한 패턴만 필요하다.

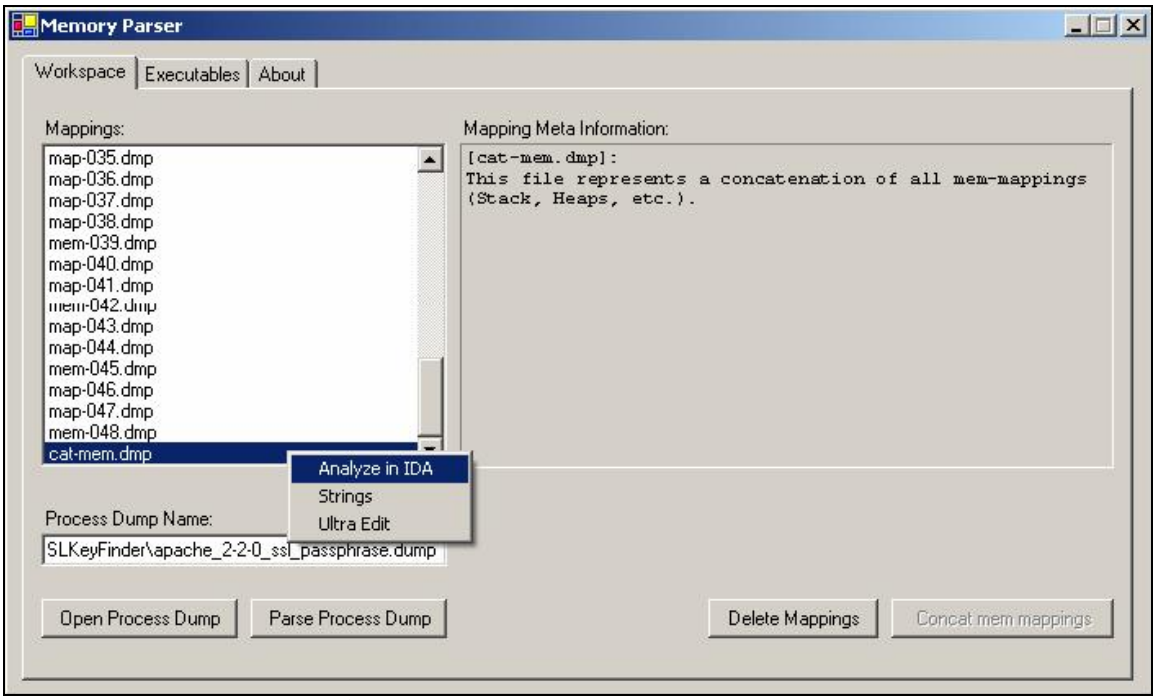
하나의 패턴은 키 혹은 인증서의 길이를 얻기 위해 적당한 SEQUENC 길이를 해석함으로서 키 혹은 인증서의 신뢰성 있는 추출이 가능하도록 한다.

3 실행

다음의 두 개의 개념 증명 툴은 프로세스 메모리로부터 인증서와 동일한 개인키를 추출하는 것이 가능하다는 것을 보여준다.

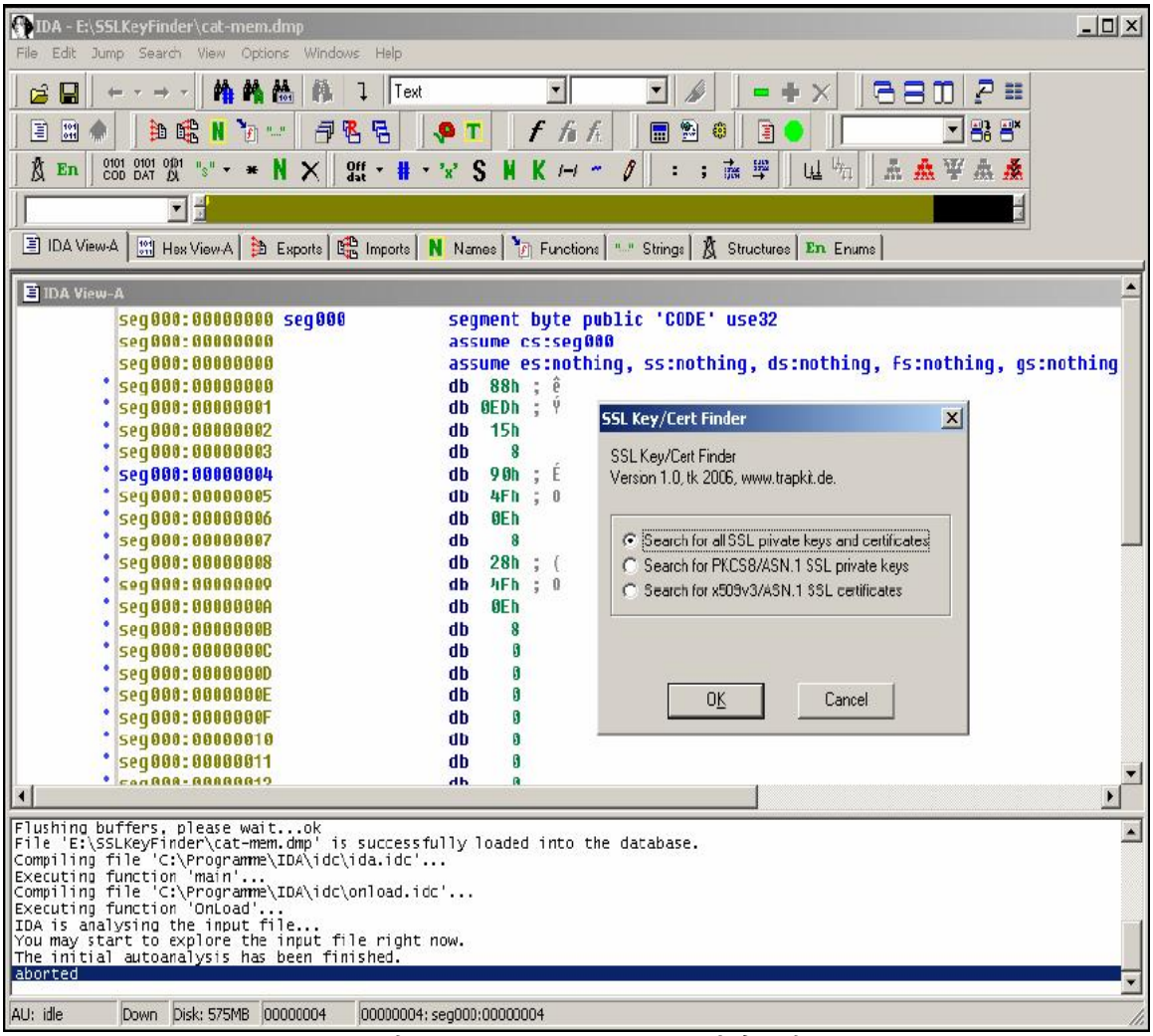
IDA Pro 플러그인 SSL Key/Cert Finder

첫 번째는 Datarescue [4]의 디스어셈블러인 IDA Pro의 플러그인이 실행된 것이다.
예로든 pd [5] 유틸리티는 아파치 [6] 프로세스 (version 2.2.0, with SSL support, SSL certificate with passphrase)를 덤프하기 위해 사용된다. 그리고 프로세스 덤프의 데이터 맵핑들(stack, heap, etc.)은 MMP [7]와 연결되어 진다. 프로세스의 데이터 맵핑된 파일을 통해 RSA 개인키 와 인증서를 찾을 수 있다.

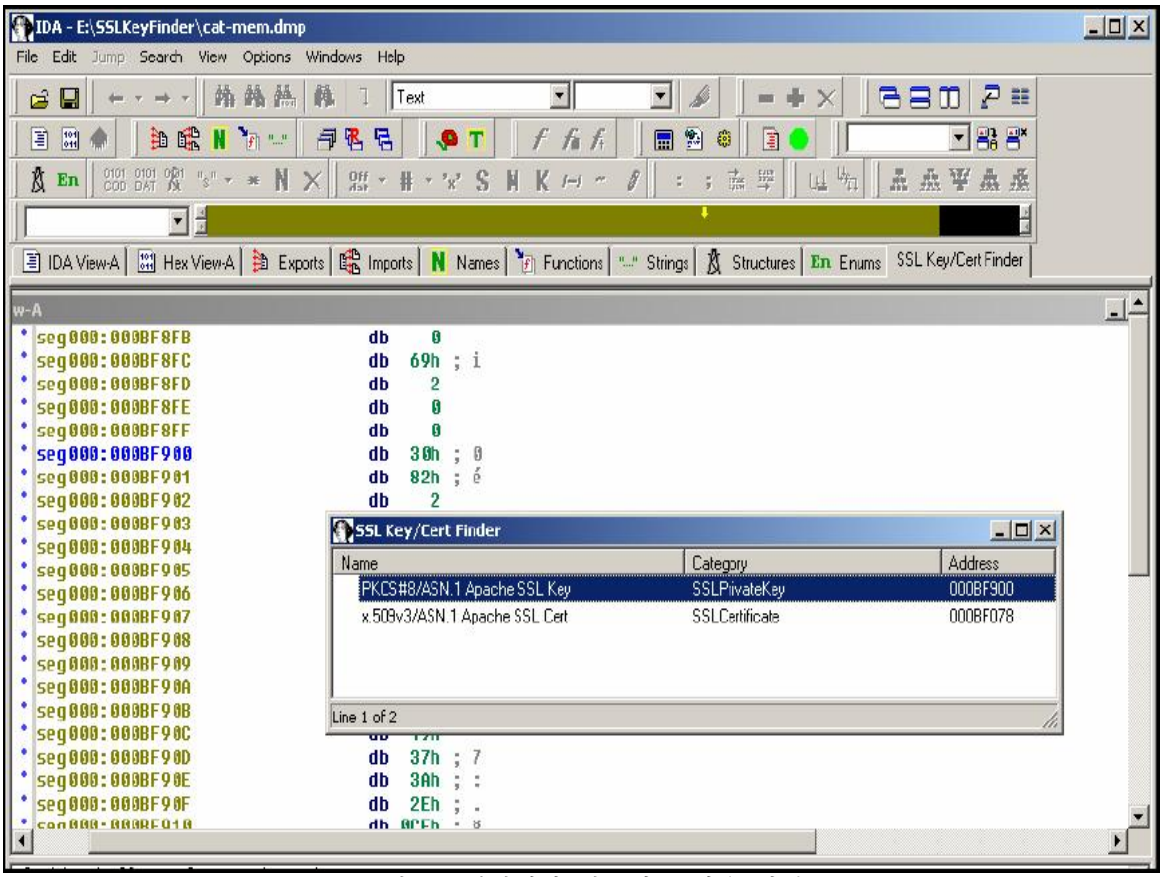


[그림 1] Concatenate data mapping with MMP

[그림1]은 아파치 프로세스의 데이터 맵핑(data mapping) 파일이 IDA Pro에 의해 로드된 화면이다. SSL Key/Cert Finder 플러그인은 단축키 SHIFT + S를 누르면 된다.

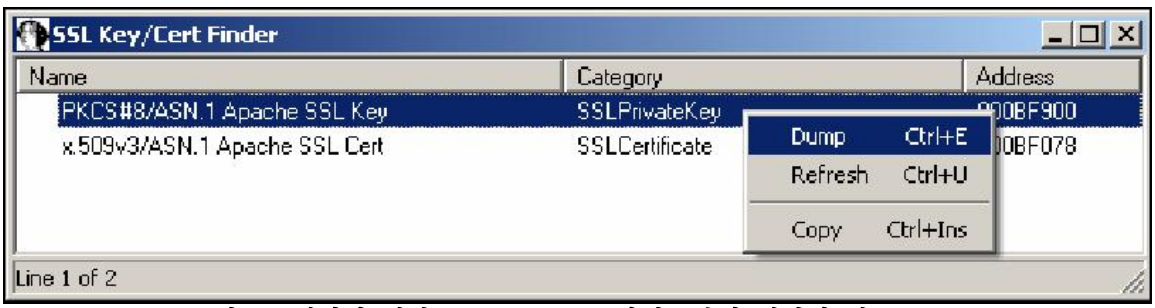


[그림 2] SSL Key/Cert Finder 플러그인



[그림 3] 개인키와 인증서를 찾은 화면

오른쪽 클릭을 하게 되면 디스크에서 개인키나 인증서의 추출이 가능한 Dump 메뉴가 있다.



[그림 4] 데이터 맵핑(data mapping) 파일로부터 개인키 덤프(dump)

현재 우리는 획득한 개인키와 인증서가 유효한지 확인하기위해서 OpenSSL을 사용할 수 있다.

```
$ openssl rsa -inform DER -check -text -in extracted_key.txt
Private-Key: (1024 bit)
modulus:
00:be:19:37:3a:2e:cf:2a:6c:fd:8a:44:da:43:d6:
dd:19:5a:10:5a:f4:bf:93:bb:60:be:1c:24:96:f1:
40:f5:f1:97:ca:2c:40:ed:dd:85:58:7b:26:68:4c:
c7:d2:7a:11:82:6f:45:9e:ff:a5:ff:11:ac:da:26:
7f:6d:9d:90:7f:12:64:ee:03:1b:f9:44:96:c3:3a:
76:4a:3c:58:9d:f1:32:8b:dc:d2:29:2b:12:89:96:
a8:b7:fd:5d:b9:7f:76:4c:db:12:e8:b1:33:56:85:
d3:b2:ed:08:0e:29:7a:05:a3:3e:3c:17:24:69:8d:
1c:bd:27:8d:b5:38:35:86:c9
publicExponent: 65537 (0x10001)
privateExponent:
46:f3:c8:6e:39:fc:6e:dc:61:41:93:73:57:f0:c1:
73:6d:ef:3e:d3:ad:11:a9:d5:70:ff:b6:14:74:95:
87:76:95:ee:0a:d8:6d:2f:ca:4e:7d:20:97:bb:58:
b5:d1:83:e9:88:38:97:20:da:47:3a:c4:a6:63:ca:
1a:12:be:54:59:f2:5d:53:5d:4c:58:70:d1:60:2f:
ff:1d:7a:c0:37:f7:8d:0d:80:ff:7c:47:8d:8e:92:
1b:d0:ee:54:cf:5a:b3:b8:d2:0c:6e:bb:31:0c:9b:
a5:1b:67:92:17:cf:e4:35:9b:0e:d6:e9:30:a0:f1:
f4:f6:99:64:4e:a6:b9:91
prime1:
00:f4:59:01:9c:c6:4a:a2:45:f5:af:0b:d9:1d:9a:
d6:42:6f:d3:ce:56:a3:cb:51:be:39:8f:35:3f:85:
d3:86:cd:d1:ef:09:29:d7:57:3c:b5:74:3f:91:9b:
e6:d7:42:a9:13:00:dc:e3:90:73:37:ef:2e:2b:4e:
a3:64:1b:ed:75
prime2:
00:c7:29:ef:a0:41:67:1d:56:67:69:0d:9e:73:c6:
ab:22:a6:28:74:fb:81:62:3b:a9:a7:0d:a8:d8:b7:
b2:c1:7a:58:c9:c1:4a:0b:db:a9:e0:25:d4:6c:e7:
49:7f:78:47:9b:24:62:bf:e9:53:26:ac:49:b5:1b:
92:38:74:65:85
exponent1:
55:fa:0b:8b:32:6a:88:76:bd:5f:fe:77:42:e7:7c:
84:9b:fc:97:19:fd:40:49:5e:f9:b9:de:2e:9f:d4:
32:16:b1:cb:be:19:ae:df:cf:48:b9:c2:b4:65:7a:
f0:3b:50:6a:93:5f:25:e3:69:e7:40:8d:aa:47:5d:
4e:98:55:11
exponent2:
2f:9c:7b:d7:70:ab:28:dd:45:fd:5c:2f:1b:f8:4b:
63:0e:1b:af:d3:8c:1b:a2:ad:ac:ec:dc:07:6a:ea:
c5:cb:ec:bb:d6:84:50:0f:64:2d:dc:7d:4a:c7:83:
cf:80:3e:85:fd:0d:ca:59:09:f2:bd:cf:25:07:81:
4e:13:ad:4d
coefficient:
00:c8:12:55:89:1f:5b:bf:52:62:17:bd:b2:a4:dc:
```

```
02:80:85:2c:be:d3:99:48:03:12:8a:72:27:ac:f1:
e0:21:29:17:9e:aa:a9:75:b6:5a:5d:91:7d:b4:b1:
c3:09:47:55:45:fd:2f:d6:17:28:f9:10:dc:de:4a:
fb:57:a2:81:89
RSA key ok
writing RSA key
-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQC+GTc6Ls8qbP2KRNpD1t0ZWhBa9L+Tu2C+HCSW8UD18ZfKLEDt
3YVYeyZoTMfSehGCb0We/6X/EazaJn9tnZB/EmTuAxv5RJbDOnZKPFid8TKL3NIp
KxKJlqi3/V25f3ZM2xLosTNWhdOy7QgOKXoFoz48FyRpjRy9J421ODWGyQIDAQAB
AoGARvPIbjn8btxhQZNzV/DBc23vPtOtEanVcP+2FHSVh3aV7grYbS/KTn0gl7tY
tdGD6Yg4lyDaRzrEpmPKGhK+VFnyXVNdTFhw0WAv/x16wDf3jQ2A/3xHjY6SG9Du
VM9as7jSDG67MQybpRtnkhfP5DWbDtbpMKDx9PaZZE6muZECQQD0WQGcxkqiRfWv
C9kdmtZCb9POVqPLUb45jzU/hdOGzdHvCSnXVzy1dD+Rm+bXQqkTANzjkHM37y4r
TqNkG+11AkEAxynvoEFnHVZnaQ2ec8arIqYodPuBYjuppw2o2LeywXpYycFKC9up
4CXUbOdJf3hHmyRiv+lTJqxJtRuSOHRlhQJAVfoLizJqiHa9X/53Qud8hJv8lxn9
QEle+bneLp/UMhaxy74Zrt/PSLnCtGV68DtQapNfJeNp50CNqkddTphVEQJAL5x7
13CrKN1F/VwvG/hLYw4br9OMG6KtrOzcB2rqxcvsu9aEUA9kLdx9SseDz4A+hf0N
ylkJ8r3PJQeBThOtTQJBAMgSVYkfW79SYhe9sqTcAoCFLL7TmUgDEopyJ6zx4CEp
F56qqXW2Wl2RfbSxwwlHVUX9L9YXKPkQ3N5K+1eigYk=
-----END RSA PRIVATE KEY-----
```

```
$ openssl x509 -inform DER -text -in extracted_cert.txt
Certificate:
Data:
Version: 3 (0x2)
Serial Number:
f6:26:f7:15:27:7a:ed:ec
Signature Algorithm: sha1WithRSAEncryption
Issuer: C=DE, ST=Some-State, L=HN, O=COMPANY, CN=www.company.net
Validity
Not Before: Jan 4 10:21:51 2006 GMT
Not After : Feb 3 10:21:51 2006 GMT
Subject: C=DE, ST=Some-State, L=HN, O=COMPANY, CN=www.company.net
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
RSA Public Key: (1024 bit)
Modulus (1024 bit):
00:be:19:37:3a:2e:cf:2a:6c:fd:8a:44:da:43:d6:
dd:19:5a:10:5a:f4:bf:93:bb:60:be:1c:24:96:f1:
40:f5:f1:97:ca:2c:40:ed:dd:85:58:7b:26:68:4c:
c7:d2:7a:11:82:6f:45:9e:ff:a5:ff:11:ac:da:26:
7f:6d:9d:90:7f:12:64:ee:03:1b:f9:44:96:c3:3a:
76:4a:3c:58:9d:f1:32:8b:dc:d2:29:2b:12:89:96:
a8:b7:fd:5d:b9:7f:76:4c:db:12:e8:b1:33:56:85:
d3:b2:ed:08:0e:29:7a:05:a3:3e:3c:17:24:69:8d:
1c:bd:27:8d:b5:38:35:86:c9
Exponent: 65537 (0x10001)
X509v3 extensions:
X509v3 Subject Key Identifier:
BA:A7:CD:BC:B0:B9:C6:CC:10:9C:80:6B:F5:38:DE:20:4F:21:F6:2F
```

```
X509v3 Authority Key Identifier:
keyid:BA:A7:CD:BC:B0:B9:C6:CC:10:9C:80:6B:F5:38:DE:20:4F:21:F6:2F
DirName:/C=DE/ST=Some-State/L=HN/O=COMPANY/CN=www.company.net
serial:F6:26:F7:15:27:7A:ED:EC
X509v3 Basic Constraints:
CA:TRUE
Signature Algorithm: sha1WithRSAEncryption
3f:f4:32:1d:3e:1c:6b:ad:80:03:f4:a1:50:4f:70:a5:01:dd:
56:ac:54:29:93:ac:d5:ff:6e:97:27:09:a4:7f:57:a2:c5:3f:
75:2b:1f:69:1e:3d:ae:85:18:c7:43:7f:ad:80:71:f0:a3:8d:
90:8b:34:b2:a4:dc:a1:da:f0:ce:53:b3:f7:7c:bd:f7:4c:c4:
36:aa:ec:e3:41:5f:1f:26:ec:ee:e6:78:1e:a8:e9:eb:69:68:
d5:dd:60:02:44:3d:0a:60:2c:39:2c:69:cf:f8:f5:57:3e:2e:
85:b9:54:7d:86:f5:1f:ec:69:ab:ff:3e:d5:dc:c0:3e:ea:f2:
f5:fb
-----BEGIN CERTIFICATE-----
MIIC9TCCAl6gAwIBAgIJAPYm9xUneu3sMA0GCSqGSIb3DQEBBQUAMFsxCzAJBgNV
BAYTAkRFMRMwEQYDVQQIEwpTb21lLVN0YXRlMQswCQYDVQQHEwJITjEQMA4GA1UE
ChMHQ09NUEFOWTEYMBYGA1UEAxMPd3d3LmNvbXBhbnkubmV0MB4XDTA2MDEwNDEw
MjE1MVoXDTA2MDIwMzEwMjE1MVowWzELMAkGA1UEBhMCREUxEzARBgNVBAgTClNv
bWUtU3RhdGUxCzAJBgNVBAcTAkhOMRAwDgYDVQQKEwdDT01QQU5ZMRgwFgYDVQQD
Ew93d3cuY29tcGFueS5uZXQwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAL4Z
Nzouzyps/YpE2kPW3RlaEFr0v5O7YL4cJJbxQPXxl8osQO3dhVh7JmhMx9J6EYJv
RZ7/pf8RrNomf22dkH8SZO4DG/lElsM6dko8WJ3xMovc0ikrEomWqLf9Xbl/dkzb
EuixM1aF07LtCA4pegWjPjwXJGmNHL0njbU4NYbJAgMBAAGjgcAwgb0wHQYDVR0O
BBYEFLqnzbywucbMEJyAa/U43iBPIfYvMIGNBgNVHSMEgYUwgYKAFLqnzbywucbM
EJyAa/U43iBPIfYvoV+kXTBbMQswCQYDVQQGEwJERTETMBEGA1UECBMKU29tZS1T
dGF0ZTELMAkGA1UEBxMCSE4xEDAOBgNVBAoTB0NPTVBBTlkxGDAWBgNVBAMTD3d3
dy5jb21wYW55Lm5ldIIJAPYm9xUneu3sMAwGA1UdEwQFMAMBAf8wDQYJKoZIhvcN
AQEFBQADgYEAP/QyHT4ca62AA/ShUE9wpQHdVqxUKZOs1f9ulycJpH9XosU/dSsf
aR49roUYx0N/rYBx8KONkIs0sqTcodrwzlOz93y990zENqrs40FfHybs7uZ4Hqjp
62lo1d1gAkQ9CmAsOSxpz/j1Vz4uhblUfYb1H+xpq/8+1dzAPury9fs=
-----END CERTIFICATE-----
```

익스플로잇 페이로드(exploit payload)로서의 SSL Key/Cert Finder

두 번째 실행은 Linux IA-32 플랫폼을 위한 익스플로잇 페이로드(exploit payload)이다. 이 페이로드는 서비스에서 메모리 변조 취약점을 공격하여 아파치 웹 서버로부터 인증서와 동일한 개인키를 추출하는데 성공적으로 테스트되었다 [9]. 더 자세한 것은 소스 코드를 보기 바란다.

두 개의 개념 증명 실행은 공개적으로 이용할 수 있다. [10].

4 대응책

민감한 암호화된 자료를 보호하기 위한 방법은 메모리에 데이터가 저장되는 것을 피하는 것이다. 이것은 추가적인 하드웨어의 사용으로 이루어질 수 있는데 하드웨어 시큐리티 모듈(HSM:Hardware Security Modules)을 사용함으로서 이러한 기능을 제공받을 수 있다.

5 참고문헌

[1] Sahmir, A; van Someren, N.: *Playing hide and seek with stored keys*, 1998.

[2] RSA: *PKCS #8: Private-Key Information Syntax Standard*, An RSA Laboratories Technical Note, Version 1.2, Revised November 1, 1993.

[3] Housley, R. et al: *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, Request for Comments: 3280, April 2002.

[4] Datarescue: *IDA Pro Disassembler and Debugger*, http://www.datarescue.com.

[5] Klein, T.: *pd – Process Dumper*, http://www.trapkit.de/research/forensic/pd/.

[6] Apache Software Foundation: *Apache Webserver*, http://www.apache.org.

[7] Klein, T.: *Memory Parser (MMP)*, http://www.trapkit.de/research/forensic/mmp/.

[8] OpenSSL, http://www.openssl.org.

[9] CAN-2002-0656

[10] SSL Key/Cert Finder implementations: http://www.trapkit.de/research/sslkeyfinder/.