

Stratum 0

Hackerspace Braunschweig

[CC BY-SA 3.0 larsan](#)

- [RSS](#)

<input type="text" value="Search"/>
<input type="button" value="Navigate..."/> ▾

- [Blog](#)
- [Archives](#)
- [Writeups](#)
- [Calendar](#)
- [Wiki](#)

Hack.lu 2013: ECKA

Oct 26th, 2013 by comawill & tsuro & spq

Hey you!

Come over and help me, please. We discovered a strange key agreement protocol used on this server: ctf.fluxfingers.net:1330. They send a curve which they use later. But we think the robots are a bit UBER-cautious and do not use the curve's P. So they first exchange their public point with a technique we could not figure out. It looks like they do not need a public key for this step. Afterwards they use another technique to agree on a key which they ultimately use to send some encrypted password.

We need this last password to shut down the robo-factory on our way to the Oktoberfest.

Oh btw, the robots use AES-ECB for symmetric encryption.

Hint: He, we have the latest news for you. The first part of their strange key agreement was designed by the famous SHA-Robot Мир!

First we started to find out what happens when you connect to the service. So we found out ECKA stands for Elliptic Curve Key Agreement and learned all translations for Мир.

After googeling "sha mir" ("mir" is the transcription for "Мир") we obviously realized, that the first part of the key agreement could be Shamir's three-pass-protocol. For the second

part we guessed a Diffie–Hellman key exchange.

Shamir's three-pass-protocol with elliptic curves

1. Alice and Bob generate a secret and its inverse
2. Alice selects a point P
3. Alice encrypts P with her secret: $\alpha \cdot P = eP_1$
4. Bob encrypts eP_1 with his secret: $\beta \cdot eP_1 = \alpha \cdot \beta \cdot P = eP_2$
5. Alice decrypts eP_2 with the known inverse $\alpha^{-1} \cdot eP_2 = \alpha^{-1} \cdot \alpha \cdot \beta \cdot P = \beta \cdot P = eP_3$
6. Bob decrypts eP_3 with the known inverse $\beta^{-1} \cdot eP_3 = \beta^{-1} \cdot \beta \cdot P = P$
7. knows P now

Diffie–Hellmann with elliptic curves

1. Alice and Bob generate both a secret
2. Alice sends $s_{alice} \cdot P$ to Bob
3. Bob sends $s_{bob} \cdot P$ to Alice
4. Both know the secret $s_{bob} \cdot s_{alice} \cdot P$

The easiest way to implement this key exchange is, by using 1 as a secret each round. So the following script decrypts the flag `b3nDer_<3_3PDHKE` with about a 50/50 chance.

```

1  #!/usr/bin/env python
2
3  from sock import Sock
4  from Crypto.Cipher import AES
5
6  s = Sock("ctf.fluxfingers.net", 1330)
7  # ignore first line
8  s.read_until("\n")
9  # get a * P
10 aP = s.read_until("\n")
11 # send 1 * a * P == a * P
12 s.send(aP)
13 # get 1 * P = P
14 P = s.read_until("\n")
15 # get b * P
16 bP = s.read_until("\n")
17 # send P
18 s.send(P)
19 # get cypher
20 cypher = s.read_until("\n")
21
22 # decrypt the cypher using b * P as the key
23 key = ("%064x" % int(bP[1:-2].split(" : ")[0])).decode("hex")
24 crypt = AES.new(key, AES.MODE_ECB)
25 flag = crypt.decrypt(cypher.strip().decode("base64"))[:-1]
26 print flag

```

Posted by comawill & tsuro & spq Oct 26th, 2013 Categories: [ctf](#), [english](#), [hack.lu13](#), [writeup](#)

[« Hack.lu 2013: Wannabe Zeitabgleich: CCCAC »](#)

[Like this post](#)

Be the first to comment...



Free forums and commenting by Muut

Recent Posts

- [Stratumnews Vol. 19](#)
- [Stratumnews Vol. 18](#)
- [Stratumnews Vol. 17](#)
- [Stratumnews Vol. 16](#)
- [Replacing a Husqvarna Viking Designer Topaz 30 Needle Holder](#)

Categories

- [english \(30\)](#)
- [ctf \(25\)](#)
- [writeup \(24\)](#)
- [german \(17\)](#)
- [hack.lu13 \(10\)](#)
- [CSAW2013 \(6\)](#)
- [asisCTF13 \(4\)](#)
- [stratumnews \(4\)](#)
- [crypto \(4\)](#)
- [howto \(3\)](#)

[Impressum](#)

Copyright © 2017 – Stratum 0 – Powered by [Octopress](#)