

# Τμήμα Πληροφορικής και Τηλεπικοινωνιών

## Προστασία και Ασφάλεια Υπολογιστικών Συστημάτων

# Εαρινό Εξάμηνο 2018-2019

Υπεύθυνος Καθηγητής: Χατζηκοκολάκης Κ.



## Project #2: Capture The Flag

Ομάδα: **ComputerInsecurity**

Μπιζίμης Μιχαήλ 1115201500102  
Παντελιάδη Ευφροσύνη 1115201400301

# Εισαγωγή

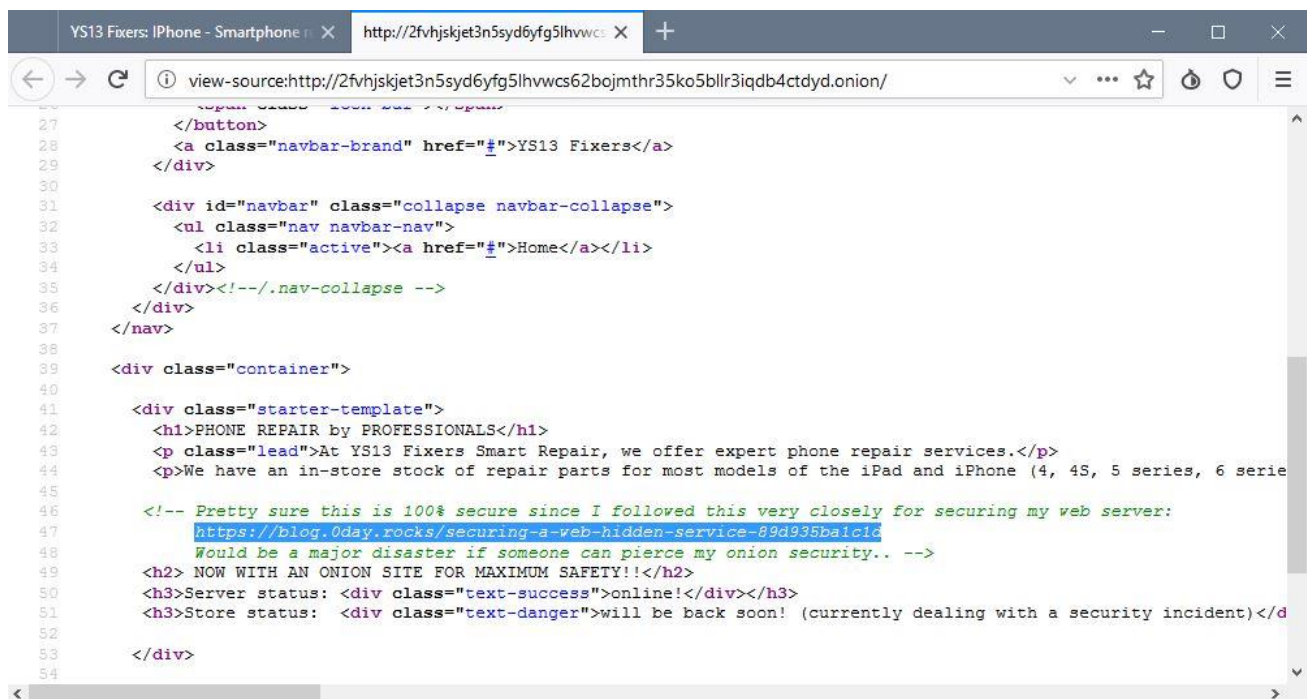
Ο στόχος αυτής της εργασίας ήταν να ακολουθήσουμε hints και τυχόν ευπάθειες ασφαλείας σε ένα δοδμένο onion site (μέσω TOR) με σκοπό να ανακαλύψουμε την απάντηση στις ακόλουθες ερωτήσεις:

- 1) Πού είναι ο Γιώργος αυτή τη στιγμή;
- 2) Ποιό είναι το hobby της Υβόνης, και ποιό το τελευταίο συστατικό που της λείπει;

Onion Site: <http://2fvhjjskjet3n5syd6yfg5lhvwc62bojmt3r35ko5bllr3iqdb4ctdyd.onion>

## Αρχικά Βήματα

Το πρώτο hint που βρήκαμε βρισκόταν με την μορφή σχολίων στον source html κώδικα του *index.html* του site:



```
27 </button>
28 <a class="navbar-brand" href="#">YS13 Fixers</a>
29 </div>
30
31 <div id="navbar" class="collapse navbar-collapse">
32 <ul class="nav navbar-nav">
33 <li class="active"><a href="#">Home</a></li>
34 </ul>
35 </div><!--/.nav-collapse -->
36 </div>
37 </nav>
38
39 <div class="container">
40
41 <div class="starter-template">
42 <h1>PHONE REPAIR by PROFESSIONALS</h1>
43 <p class="lead">At YS13 Fixers Smart Repair, we offer expert phone repair services.</p>
44 <p>We have an in-store stock of repair parts for most models of the iPad and iPhone (4, 4S, 5 series, 6 serie
45
46 <!-- Pretty sure this is 100% secure since I followed this very closely for securing my web server:
47 https://blog.0day.rocks/securing-a-web-hidden-service-89d935ba1c1d
48 Would be a major disaster if someone can pierce my onion security.. -->
49 <h2> NOW WITH AN ONION SITE FOR MAXIMUM SAFETY!!</h2>
50 <h3>Server status: <div class="text-success">online!</div></h3>
51 <h3>Store status: <div class="text-danger">will be back soon! (currently dealing with a security incident)</div>
52
53 </div>
54
```

Πηγαίνοντας σε αυτόν τον ιστότοπο διαβάσαμε για διάφορα security issues που μπορεί να προκύψουν από κακή ρύθμιση ενός server και είπαμε να τα δοκιμάσουμε στο εν λόγω site.

Έπειτα, δοκιμάσαμε αν είναι ενεργοποιημένες οι σελίδες `/server-info` και `/server-status`, από τις οποίες ήταν ενεργοποιημένη η πρώτη. Εκεί, πολύ καλά κρυμμένη, υπήρχε το url ενός δεύτερου onion site που έκανε host ο ίδιος server:

Δεύτερο onion site: <http://jt4grrjwzyz3pjkylwfaux5xnjai23vxmhsqaevfhrfylelw4hvxcuycd.onion>

<http://jt4grrjwzyz3pjky/wfau5xnaj23vxmhskqaeyfhrfylelw4hvxcuyd.onion/blogposts/diary2.html>

οπότε το πρώτο που σκεφτήκαμε να κάνουμε είναι να δούμε αν έχουμε πρόσβαση στο directory listing του `/blogposts` και πράγματι είχαμε:

Εκεί είδαμε ότι το *post3.html* και τα αρχεία στο */tmp/* αφορούσαν τον Γιώργο και άρα το πρώτο ερώτημα της εργασίας, ενώ το *dairy2.html* και ο server (pico) στην θύρα 47474 του αρχικού YS13 Fixers site αφορά τα hobby της Υβόνης και άρα το δεύτερο ερώτημα της εργασίας.

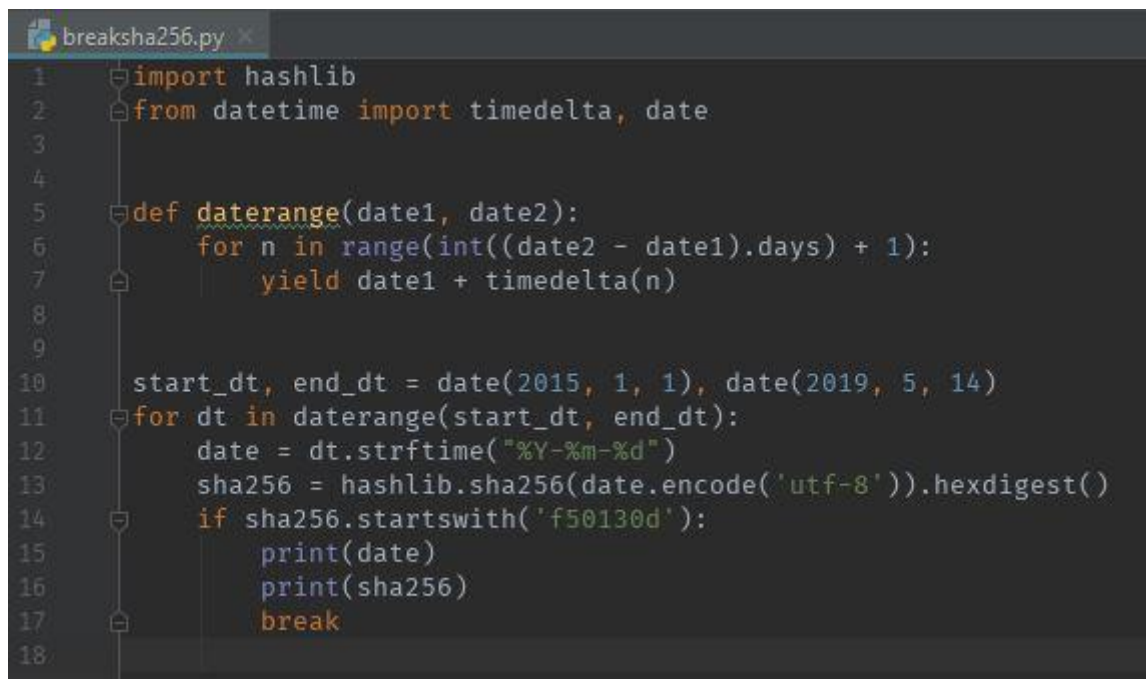
## Πρώτο Ερώτημα

---

Μέσα στα αρχεία του Γιώργου περιλαμβάνονται δύο *.gpg* κρυπτογραφημένα αρχεία, ένα αρχείο *notes.txt.truncated* και ένα αρχείο *passphrase.key.truncated*. Σύμφωνα με το δεύτερο (*.txt*), ο Γιώργος κρυπτογράφησε με GPG τα *.gpg* αρχεία χρησιμοποιώντας ως passphrase το SHA256 hash μίας RFC3339 ημερομηνίας (την ημερομηνία που τα κρυπτογράφησε), το οποίο είχε αποθηκεύσει σε αρχείο *passphrase.key*. Στο αρχείο *passphrase.key.truncated* επομένως βρίσκεται ένα μέρος (prefix) του hash που χρησιμοποίησε.

Έχοντας ένα prefix του hash και υποθέτοντας ότι η ημερομηνία που κρυπτογράφησε τα αρχεία του θα είναι κάποια λογική ημερομηνία πριν την *last-modified* ημερομηνία των αρχείων εκεί ("2019-05-14"), για να βρούμε ολόκληρο το passphrase αρκεί να δοκιμάσουμε ημερομηνίες πριν την 2019-05-14 μέχρι αυτήν που το hash της να έχει ως prefix το *passphrase.key.truncated*.

Το παραπάνω το κάναμε με ένα δικό μας python 3 script που ονομάσαμε *breaksha256.py*. Αυτό το script παίρνει τα SHA256 hashes ημερομηνιών από το *start\_dt* μέχρι *end\_dt* και αν έχουν ως prefix το 'f50130d' (*passphrase.key.truncated*) τότε σταματά, εκτυπώνει την ημερομηνία και το hash της (το οποίο είναι το passphrase για το gpg).



```
1 import hashlib
2 from datetime import timedelta, date
3
4
5 def daterange(date1, date2):
6     for n in range(int((date2 - date1).days) + 1):
7         yield date1 + timedelta(n)
8
9
10 start_dt, end_dt = date(2015, 1, 1), date(2019, 5, 14)
11 for dt in daterange(start_dt, end_dt):
12     date = dt.strftime("%Y-%m-%d")
13     sha256 = hashlib.sha256(date.encode('utf-8')).hexdigest()
14     if sha256.startswith('f50130d'):
15         print(date)
16         print(sha256)
17         break
18
```

Το αποτέλεσμα αυτού του script ήταν:



```
2019-03-20
f50130d62f7776f5202d01aa028a366af78db589d7f1bb3bfeb3b0ba312ebd1e
```

δηλαδή η κρυπτογράφηση είχε γίνει στις 2019-03-20 και το passphrase για το GPG είναι το f50130d62f7776f5202d01aa028a366af78db589d7f1bb3bfef3b0ba312ebd1e .

Έχοντας το passphrase αποκρυπτογραφήσαμε επιτυχώς τα .gpg αρχεία του κινητού του Γιώργου, και βρήκαμε την εξής συνομιλία στο *signal.log*:

```
D:\Documents (Data_Drive)\Μαθήματα - ΣΧΟΛΗ\8ο Εξάμηνο\Προστασία και Ασφάλεια\Άσκηση 2>gpg --decrypt signal.log.gpg
gpg: AES256 κρυπτογραφημένα δεδομένα
gpg: κρυπτογραφημένο με 1 φράση κλειδί
22 Feb 15:18 - You:      Hey Maria :)
23 Mar 20:44 - You:      Maria? I need a favor.
24 Mar 13:32 - You:      Maria, I passed from your place the other day but you were not there. Please call me it's urgent.
24 Mar 13:34 - Maria:    ???
25 Mar 13:35 - You:      Hey Maria! I'm trying to make sense of the inbetweens. I think I'm part of some weird game other people are playing on me...
25 Mar 13:35 - You:      my flat got poisoned by those people and im now sleeping in the balcony.
25 Mar 13:35 - You:      the neighbors are looking at me when i sleep.
25 Mar 13:36 - You:      i saw a girl in the eleveator yesterday holding a plant with big branches. she started talking to me. i didnt naswer. i think.
25 Mar 13:36 - You:      my mobile phone is broken. its tracking me and sending details to those men. i need to send it for repair.
25 Mar 13:37 - You:      i think im not well. need to escape this city. they are looking at me. i dont git why...
25 Mar 13:37 - You:      please come and find me. you are the only person i can commit to: 574940e15c0a68d89f3fd4dd2ef068dce9a77367
25 Mar 13:38 - You:      hope to see you soon! thanks!>
25 Mar 17:12 - Maria:    ??? What are you talking about? I'm not Maria. I think you got the wrong number mate.
```

Ψάχνοντας στο Google το hash (574940e15c0a68d89f3fd4dd2ef068dce9a77367) της συνομιλίας αυτής ανακαλύψαμε ότι πρόκειται για το hash ενός commit στο github:

asn-d6 / tor  
forked from torproject/tor

Watch 1 Star 0 Fork 387

Code Pull requests 0 Projects 0 Security Insights

Comments

bug23512\_v2

asn-d6 committed on May 8 1 parent b3492d5 commit 574940e15c0a68d89f3fd4dd2ef068dce9a77367

Showing 1 changed file with 9 additions and 0 deletions. Unified Split

src/feature/hs/hs\_service.c

```
@@ -226,6 +226,15 @@ remove_service(hs_service_ht *map, hs_service_t *service)
226 226     }
227 227     }
228 228
229 + /** Hey maria... Come and find me here:
230 + *
231 + * x + y^2 = 614.6863999999999
232 + * x^2 + y^2 = 1793.6720000000003
233 + * x^2 + 2*y = 1261.9856000000004
234 + *
235 + * See you soon!
236 + */
237 +
229 238     /* Set the default values for a service configuration object <b>c</b>. */
230 239     static void
231 240     set_service_default_config(hs_service_config_t *c,
```

Λύνοντας αυτές τις εξισώσεις, καταλείξαμε στις συντεταγμένες:

$$x = 34.84 \text{ και } y = 24.08$$

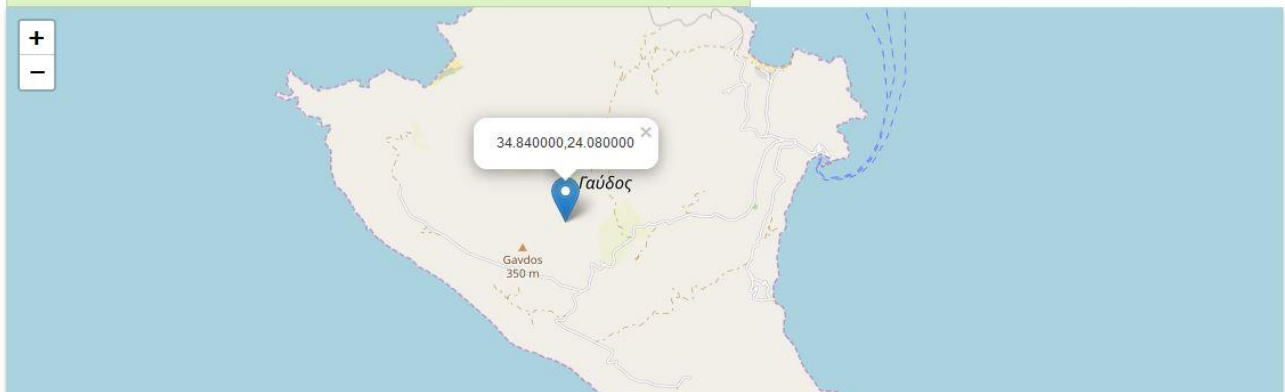


οι οποίες πρέπει να είναι η τοποθεσία του Γιώργου:

## Reverse Geocoding Convert Lat Long to Address

**Reverse geocoding** is the process to convert the latitude and longitude coordinates to a readable address. Type the lat long coordinates and press Convert button. Reverse geocoded address will shown below. Also the municipality, subdivision and country name can be found.

Latitude	Longitude	
<input type="text" value="34.84"/>	<input type="text" value="24.08"/>	<input type="button" value="Convert"/>
Example: 40.785091	Example: -73.968285	
Reverse geocoded address:		
<div>73001 Γαύδος Γαύδος Ελλάδα</div>		



## Δεύτερο Ερώτημα

---

Όσον αφορά το δεύτερο ερώτημα (το hobby της Υβόνης), διαβάζοντας το <http://jt4grrjwzyz3pjkylwfau5xnjaj23vxmhskqaeyfhrfylelw4hvxcuyd.onion/blogposts/diary2.html> ήταν εμφανές ότι η Υβόνη είχε σετάρει τον pico server (<https://github.com/chatziko/pico>) στην θύρα 47474 του αρχικού onion site. Προσπαθώντας να επισκευτούμε την συγκεκριμένη διεύθυνση:

`http://2fvhjskjet3n5syd6yfg5lhvwcs62bojmrthr35ko5bllr3iqdb4ctdyd.onion:47474/`

ο server ζητάγε authentication, άρα έπρεπε να βρούμε έναν τρόπο να περάσουμε το authentication αυτό για να μάθουμε το hobby της Υβόνης.

Προκειμένου να το κάνουμε αυτό, σετάραμε τον pico server τοπικά και εξετάσαμε τον C κώδικα του για ευπάθειες. Αρχικά, ψάξαμε μήπως υπάρχει θέμα buffer overflow, κάτι συνηθισμένο σε C κώδικα, αλλά όλα φαινόταν να γίνονται προσεκτικά. Έπειτα, σκεφτήκαμε μήπως κάποια ειδική είσοδος μπορεί να κάνει bypass τον έλεγχο της `check_auth()` στο `main.c`, αλλά δεν βρήκαμε κάτι τέτοιο. Τέλος, προσέξαμε το warning του compiler κατά την μεταγλώττιση του server:

```
"main.c:126:12: warning: format not a string literal and no format arguments [-Wformat-security] printf(auth_username);"
```

και υποπτευθήκαμε ότι εκεί πρέπει να είναι το πρόβλημα.

Ψάχνοντας στο Google περί αυτού του warning πέσαμε στο ακόλουθο πολύ χρήσιμο κείμενο περί «Format String Vulnerability»:

[http://www.cis.syr.edu/~wedu/Teaching/cis643/LectureNotes\\_New/Format\\_String.pdf](http://www.cis.syr.edu/~wedu/Teaching/cis643/LectureNotes_New/Format_String.pdf)

το οποίο εξηγεί πολύ καλά πως μπορεί κανείς να εκμεταλευτεί το `printf(auth_username);` του pico. Η ιδέα είναι η εξής:

Το `auth_username` θα ερμηνευτεί από τον κώδικα που εκτελεί την `printf` ως format string, το οποίο μπορεί να περιέχει format parameters όπως `"%d"`, `"%x"`, `"%s"`, etc. Η `printf(auth_username);` όμως δεν δέχεται παραμέτρους στην κλήση της για να ματσάρουν τα πιθανά format parameters του `auth_username`, επομένως ο (low-level) κώδικας της – ο οποίος δεν θα το πιάσει ως runtime error στην C – θα συμπεριφερθεί σαν να είχε της απαιτούμενες παραμέτρους στην στοίβα του process (ενώ στην πραγματικότητα δεν της έχει). Με αυτόν τον τρόπο μπορεί να διαβαστεί (ακόμα και να γραφτεί με `"%n"`) μνήμη της στοίβας που δεν έχει να κάνει με την `printf()`!

Εμείς αυτό που θα θέλαμε να διαβάσουμε από την μνήμη είναι κάποια credentials (username, password) που να μας επιτρέψουν να περάσουμε το authorization check του pico. Αυτά τα credentials αποθηκεύονται σε ένα αρχείο (.htpasswd) και διαβάζονται από τον server κατά την εκκίνησή του σε μία global δομή "char users[100][100]". Το γεγονός ότι αυτή η δομή είναι global είναι χρήσιμο επειδή η διεύθυνσή πχ του `users[0]` (του πρώτου χρήστη που διαβάστηκε) μένει σταθερή (μεταξύ εκτελέσεων) επομένως λογικά θα είναι η ίδια στον server της Υβόνης με αυτήν που παίρνουμε για την τοπική εκτέλεση του pico: `e004e2c0`.

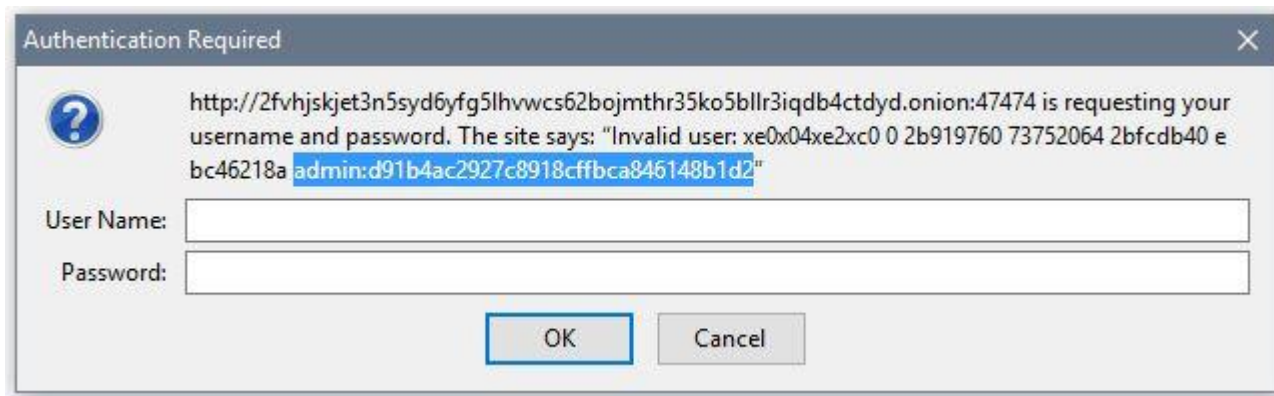
Το `auth_username` είναι τοπικό array από chars και άρα αποθηκεύεται στην στοίβα. Δίνοντας στο `auth_username` την τιμή `"\xe0\x04\xe2\xc0 %x ... %x %s"` ουσιαστικά βάζουμε στην στοίβα της διεύθυνση του `user`, ενώ ταυτόχρονα ανάλογα με το format parameter κάνουμε τον κώδικα της `printf()`:

- Για `"%x"`: να εκτυπώσει ως δεκαεξαδικό την επόμενη τιμή στην στοίβα, ενώ ταυτόχρονα μετακινεί τον stack pointer πιο πάνω. Ουσιαστικά θέλουμε απλά να προχωρήσουμε το stack pointer για να αποφύγουμε άλλα πράγματα της στοίβας (όπως άλλες τοπικές μεταβλητές) πριν το `auth_username`.
- Για `"%s"`: να διαβάσει ως C-συμβολοσειρά τα bytes στα οποία δείχνει η διεύθυνση που υπάρχει (εκείνη την στιγμή) στην στοίβα (και φυσικά να μετακινήσει τον stack pointer πιο πάνω). Ουσιαστικά θέλουμε να πείσουμε πάνω στην διεύθυνση που βάλαμε εμείς: `e004e2c0`, έτσι ώστε να διαβαστεί το `users[0]`, που θα έχει το username και το password για κάποιον authorized χρήστη.

Το tricky κομμάτι είναι να βρούμε πόσα `"%x"` πρέπει να βάλουμε πριν το `"%s"`, ώστε να πείσουμε ακριβώς στην διεύθυνση του `users[0]` που βάλαμε οι ίδιοι στην στοίβα. Δοκιμάζοντας το τοπικά βρήκαμε χρειάστηκε 6 `"%x"`, οπότε δοκιμάσαμε να μπούμε στο site της Υβόνης με username:

```
"\xe0\x04\xe2\xc0 %x %x %x %x %x %x %s"
```

και πήραμε την εξής απάντηση:



Που αποκαλύπτει ότι υπάρχει authorized user με username "admin" και το MD5 hash του κωδικού του είναι `"d91b4ac2927c8918cffbca846148b1d2"`.



Αυτό που μένει να κάνουμε τώρα είναι να βρούμε ένα input (όχι απαραίτητα το πραγματικό κωδικό) που να δίνει αυτό το MD5 hash.

Θεωρητικά ένα hash πρέπει να είναι μονοσήμαντο, αλλά ψάχνοντας σε site για "MD5 decrypt" (πχ: <https://md5decrypt.net/en/>) καταφέραμε να βρούμε ότι ένα τέτοιο input είναι το "chubbyruppiesarecutelol", το οποίο μαζί με το username "admin" μας επέτρεψε να περάσουμε το authorization και να μπούμε στην σελίδα με το hobby της Υβόνης, η οποία προσπαθεί να κατασκευάσει έναν Large Hadron Collider (LHC) στο υπόγειο του καταστήματος YS13 και το τελευταίο συστατικό που της λείπει είναι μαγνήτες:

