Ασφάλεια Λογισμικού & Δικτύων

Εργασία: Τίμια Ψηφιακά Ζάρια

Διδάσκοντας: Ιωάννης Μαριάς

Ελένη Τράμπαρη Λάρδα: f3312217
Δημήτρης Κλαδούχος: f3312211

Δύο διαδικτυακοί παίκτες (ένας παίκτης και ο server) παίζουν ζάρια (από ένα ο καθένας). Ο μεγαλύτερος κερδίζει. Στην ισοπαλία δεν κερδίζει κανείς

Ακολουθήθηκε ως πρότυπο το πρωτόκολλο που απεικονίζεται στην εκφώνηση της εργασίας. Έχουμε φτιάξει το πρωτόκολλο με την ίδια λογική υποθέτοντας ότι μας ενδιαφέρει μόνο αν ο server έκλεψε και όχι ο client.

Ερώτημα Α:

Στο πρώτο ερώτημα σχεδιάσαμε ένα κρυπτογραφικό πρωτόκολλο επικοινωνίας μεταξύ client και server. Για την υλοποίησή του χρησιμοποιήσαμε Django – python.

Συγκεκριμένα το πρωτόκολλο αυτό είναι το εξής:
Αρχικά, δημιουργείται ένα μοναδικό και τυχαίο rA για τον server και rB για τον client (όπου έχουν 10 τυχαία στοιχεία: κεφαλαία, πεζά, αριθμοί).

Client → rB Server

Client $y = SHA_{256}(server\_dice||rA||rB)$ ← Server

Client → client_dice Server

Client server_dice||rA||rB||result ← Server

Στο **1ο βήμα** ο client στέλνει το rB του στον server.

```
<script >
    function myFunction(){
        //rb
        //kefalaia mikra arithoi 10 stoixeia
        //------------------RB GENERATION------------------------------------
        const rB_func = (length) => {
            const characters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789';
            const charactersLength = characters.length;
            let rB_ = '';

            // Create an array of 32-bit unsigned integers
            const randomValues = new Uint32Array(length);

            // Generate random values
            window.crypto.getRandomValues(randomValues);
            randomValues.forEach((value) => {
                rB_ += characters.charAt(value % charactersLength);
            });
        return rB_;
        }

        //--------------------RB GENERATION------------------------------------
        //local storage --> client dice & rB
        localStorage.setItem("rB",rB_func(10));
        console.log("rb from client interface = ",rB_func(10));

        rB=localStorage.getItem("rB");

        //pernas sto hidden to rb
        document.getElementById("client_rB").value = rB;
        //kaleis submit button
        document.getElementById('submit-button').click();
    };
```

Δημιουργία του rB και εισαγωγή του ως value στον κουμπί submit

```
<form id="form1" method="POST" action = "{% url 'test' %}">

    {% csrf_token %}

    <!-- input for rB to store -->
    <input type="hidden" id="client_rB" name="client_rB_hash">
    <button type="button" id="click_button" onclick="myFunction()" >Play</button>

    <!-- hidden button clicked by code send data to server (send rB) -->
    <button type="submit" id="submit-button" style="display: none;" >Submit</button>
</form>
```

Το rB στέλνεται μέσω φόρμας με submit button και καλείται η μέθοδος test από views.py

Στο **2ο βήμα** ο server όταν το λάβει ρίχνει το ζάρι και κρυπτογραφεί το αποτέλεσμα του ζαριού (server_dice) μαζί με το rA, rB και το στέλνει στον client (y= SHA_{256} (server_dice||rA||rB)). Με αυτό τον τρόπο ο server δεν μπορεί να αλλάξει το αποτέλεσμα του ζαριού αφού το έχει στείλει στον client και μπορεί να επιβεβαιώσει αργότερα τι έστειλε στην αρχή και επίσης ο client δε μπορεί να δει τι έχει μέσα η κρυπτογραφημένη συνάρτηση γιατί είναι υπολογιστικά αδύνατο. Ακόμα, δεν μπορεί να υλοποιηθεί man-in-the-middle και να αλλάξει το αποτέλεσμα του ζαριού.

```
def test(request):
    if request.method == 'POST':
        #Client.client_rB = request.POST.get('client_rB_hash', '')
        temp_rB = request.POST.get('client_rB_hash', '')
        print(temp_rB + " client's rB")

        #save rB in client's model
        Client.client_rB = temp_rB
        print("client rb in test = "+Client.client_rB.__str__())

        #rA upologismos
        chars = string.ascii_uppercase + string.ascii_lowercase + string.digits
        r_server = ''.join(random.choice(chars) for _ in range(10))

        serverDice = random.randrange(1,7)
        print("server dice is =  "+str(serverDice))

        #save rA in server's model
        Server.server_rA = r_server
        print("server rA in test function = "+ Server.server_rA.__str__())

        #save server dice in server's model
        Server.server_dice = serverDice
        print("server dice in test = "+Server.server_dice.__str__())

        #sha256 (dice number,rA,rB) ypologismos
        string_for_hash = str(serverDice)+r_server + temp_rB
        Server.server_string = string_for_hash
        print("server string= "+ Server.server_string.__str__())
        h_commit =hashlib.sha256(string_for_hash.encode('utf-8')).hexdigest()
        print(h_commit)

        #stelno h_commit ston client
        return render(request,'home.html',{'h_commit':h_commit})

    return render(request,'home.html')
```

Λαμβάνει το rB, ρίχνει το ζάρι, υπολογίζει την sha256 και τη στέλνει

Στο **3ο βήμα** ο client αποθηκεύει το y, ρίχνει το ζάρι κ στέλνει το δικό του αποτέλεσμα (client_dice) στον server (μη κρυπτογραφημένο καθώς δεν χρειάζεται).

```
{% if h_commit %}
<script >
    window.onload = (event)=> {
        console.log('Page loaded ');
        localStorage.setItem("h_commit","{{h_commit}}");

        //check if stored correctly
        const t= localStorage.getItem("h_commit")
        console.log("h_commit = ",t)

        //kalei tin clientnumber()
        client_number()
    };
</script>
{% endif %}
```

Αποθηκεύει το y (μεταβλητή στον κώδικα: h_commit)

```
function client_number(){
    //client_dice = random.randrange(1, 7)
    client_dice = getRandomIntInclusive(1,6)
    //---------------CLIENT DICE GENERATOR-----------------------------

    function getRandomIntInclusive(min, max) {
        min = Math.ceil(min);
        max = Math.floor(max);
        return Math.floor(Math.random() * (max - min + 1) + min);
    }

    //---------------CLIENT DICE GENERATOR END-------------------------
    localStorage.setItem("dice",client_dice);
    console.log("client dice from client interface = ",client_dice);

    document.getElementById("dice").value = client_dice;
    document.getElementById('submit-button2').click();
};
```

Ρίχνει το ζάρι και αποθηκεύει το αποτέλεσμα του στο value του κουμπιού submit

```
<form id="form2" method="POST" action = "{% url 'receive' %}">

    {% csrf_token %}

    <!-- input for dice to store -->
    <input type="hidden" id="dice" name="dice2">
    <button type="button" id="click_button2" style="display: none;" >Play</button>

    <!-- hidden button clicked by code send data to server (send rB) -->
    <button type="submit" id="submit-button2" style="display: none;" >Submit</button>
</form>
```

Με το κουμπί submit στέλνει την τιμή του id=”dice” και καλείται η receive() του views.py

Στο **4ο βήμα** ο server υπολογίζει ποιος κέρδισε συγκρίνοντας τα αποτελέσματα client_dice & server_dice (string: result). Έπειτα, στέλνει το server_dice,rA,rB,result στον client ώστε να επιβεβαιώσει ο client ότι ο server δεν έκλεψε και ότι το server_dice είναι το ίδιο που έστειλε στην αρχή.

```
def receive(request):
    if request.method == 'POST':
        #get the client dice from client
        client_dice = request.POST.get('dice2','')
        print("the client's dice is = "+str(client_dice))

        #xrisi toy model class gia get server_rA,server_dice kai client_rB
        rA = Server.server_rA.__str__()
        rB = Client.client_rB.__str__()
        s_dice = Server.server_dice.__str__()
        print ("rA = "+ rA + " rB= "+rB + " server_dice= "+ s_dice)

        #winner is
        if (client_dice > s_dice):
            result = "You are the winner"
        elif (client_dice < s_dice):
            result = "Server is the winner"
        else:
            result = "Tie, same dice number"
        server_string = Server.server_string
        print(server_string+" = server string in receive")

        #send to client server_dice,rA,rB to compute sha256 and check if cheated
        return render(request,'home.html',{'result':result,'rA':rA,
                                           'rB':rB,'s_dice':s_dice})

    return render(request,'home.html')
```

Λαμβάνει το client_dice, υπολογίζει τον winner , στέλνει τα απαραίτητα

Στο **τέλος** ο client λαμβάνει αυτά που του στέλνει ο server και υπολογίζει αν y= SHA_{256} (των στοιχείων που έστειλε στο τέλος), όπου στοιχεία που έστειλε: server_dice,rA,rB (ο client γνωρίζει με ποια στοιχεία το κρυπτογράφησε εξ αρχής).

```
<script >
    window.onload = (event)=> {
        console.log('Page loaded for result');
        localStorage.setItem("result","{{result}}");

        //bring data from temporary storage
        const h_sent= localStorage.getItem("h_commit")
        console.log("h_commit = ",h_sent)
        const rB_saved = localStorage.getItem("rB")
        //for checking if rB_saved is same with rB_send

        rB_sent = "{{rB}}"
        console.log("rB_sent is = "+rB_sent+ " rb stored = "+rB_saved)

        result = "{{result}}"
        localStorage.setItem("result", "{{result}}")
        console.log("result is "+result)

        rA = "{{rA}}"
        console.log("rA is "+ rA)

        server_dice = "{{s_dice}}"
        //kalei function gia emfanisi apotelesmatos print (form)
        check(rA,rB_saved,rB_sent,h_sent,result,server_dice)

    };
</script>
{% endif %}
```

Λαμβάνει από server τα result, rA, server_dice και καλεί την check() για έλεγχο αν έκλεψε και εμφάνιση αποτελεσμάτων στο browser

```
function check(ra,rb_saved,rb_sent,h,r,dice){
    //check if h_sent is same with h_stored
    string_for_hash = dice+ra+rb_saved
    console.log("string hash client "+string_for_hash)

    h_calc = CryptoJS.SHA256(string_for_hash).toString();
    if(h_calc==h){
        cheat="NO cheating of the server"
        console.log("correct");
    }
    else{
        cheat="YES cheating of the server"
        console.log("error");
    }
    //print the winner
    document.getElementById('dice_server').value = dice;
    document.getElementById('dice_client').value = localStorage.getItem("dice");
    document.getElementById('result').value = r;
    document.getElementById('cheat').value = cheat;
};
</script>
```

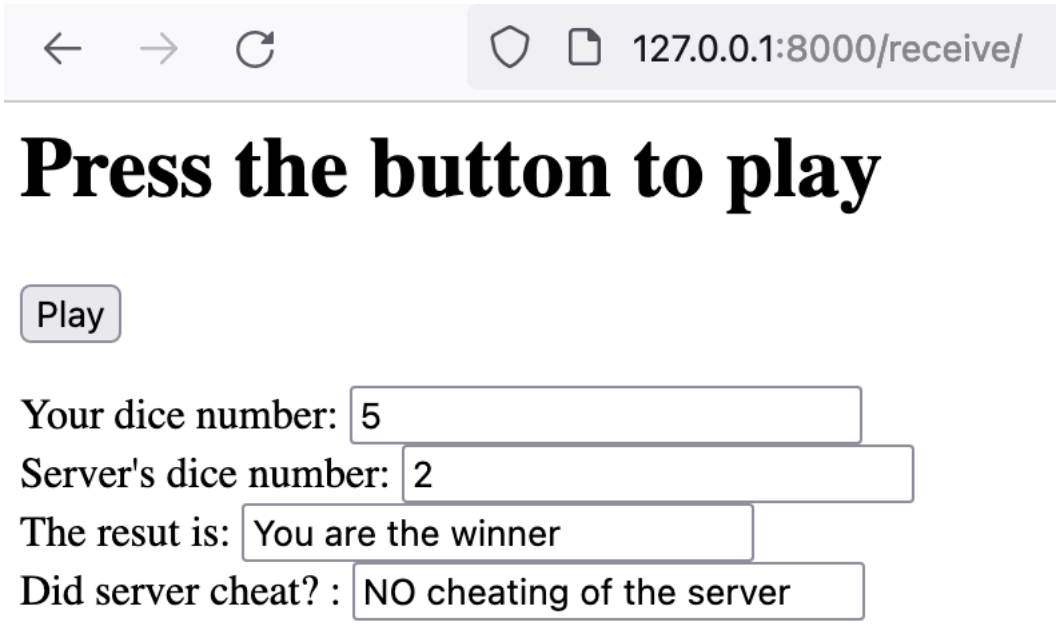
Έλεγχος αν y= SHA_{256} (των στοιχείων που έστειλε) και βάζει values στη φόρμα εμφάνισης

```
<form id="form3"  action = "">
    <label for="cdice">Your dice number:</label>
    <input type="text" id="dice_client" name="rdice_client_print"><br>
    <label for="sdice">Server's dice number:</label>
    <input type="text" id="dice_server" name="dice_server_print"><br>
    <label for="winner">The resut is:</label>
    <input type="text" id="result" name="result_print"><br>
    <label for="servercheat">Did server cheat? :</label>
    <input type="text" id="cheat" name="cheat_print">

    <button type="button" id="click_button3" style="display: none;" >Play</button>

    <!-- hidden button clicked by code send data to server (send rB) -->
    <button type="submit" id="submit-button3" style="display: none;" >Submit</button>
</form>
```

Εμφανίζει στον browser του client τα κατάλληλα δεδομένα



Ερώτημα Β:

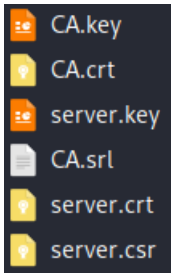
Για την προσθήκη SSL/TLS για κρυπτογράφηση από άκρο σε άκρο, τρέξαμε το αρχείο *ssl.sh* με τα παρακάτω περιεχόμενα και δημιουργήσαμε ένα self-sign certificate με ταυτόχρονη δημιουργία *.key* αρχείων που χρειάζονται στη κρυπτογράφηση:

```
$ ssl.sh
[redacted] > $ ssl.sh
1  openssl genrsa -out CA.key 2048 &&
2  openssl req -new -x509 -days 365 -key CA.key -out CA.crt &&
3  openssl genrsa -out server.key 1024 &&
4  openssl req -new -key server.key -out server.csr &&
5  openssl x509 -req -in server.csr -CA CA.crt -CAkey CA.key -CAcreateserial -out server.crt -days 365
6
```

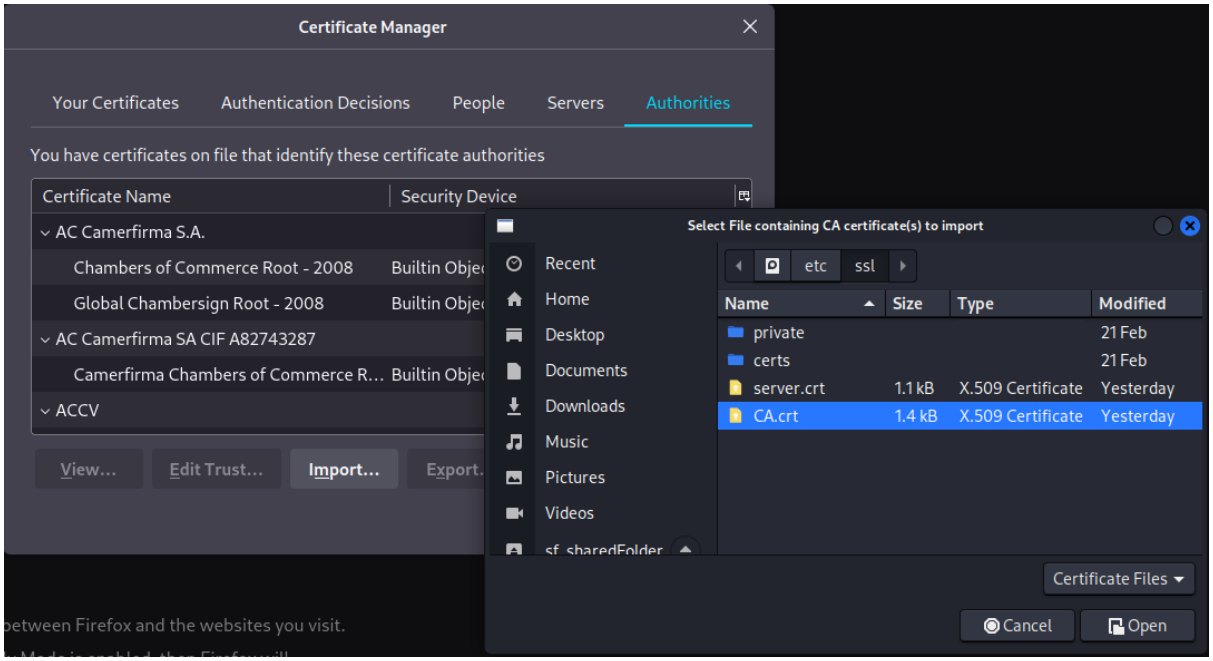
```
┌──(dimitris㉿kali)-[/media/sf_sharedFolder]
└─$ sh ssl.sh
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:GR
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:Athens
Organization Name (eg, company) [Internet Widgits Pty Ltd]:aueb
Organizational Unit Name (eg, section) []:aueb
Common Name (e.g. server FQDN or YOUR name) []:diceProject
Email Address []:admin@example.com
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:GR
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:Athens
Organization Name (eg, company) [Internet Widgits Pty Ltd]:aueb
Organizational Unit Name (eg, section) []:aueb
Common Name (e.g. server FQDN or YOUR name) []:player
Email Address []:admin2@example.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:ABCabc
An optional company name []:diceCompany
Certificate request self-signature ok
subject=C = GR, ST = Some-State, L = Athens, O = aueb, OU = aueb, CN = player, emailAddress = admin2@example.com
```

Έτσι, δημιουργήθηκαν τα παρακάτω αρχεία:



Για να αναγνωρίζει ο browser (Firefox) την Certificate Authority (CA) ως έγκυρη, κάναμε import το CA.crt αρχείο στα certificates, στο Privacy Settings tab του Firefox.



Στη συνέχεια, ανεβάσαμε τα αρχεία στον Apache και τον παραμετροποιήσαμε έτσι ώστε να ανακατευθύνει τα http σε **https** .

```
#   SSL Engine Switch:
#   Enable/Disable SSL for this virtual host.
SSLEngine on

#   A self-signed (snakeoil) certificate can be created by installing
#   the ssl-cert package. See
#   /usr/share/doc/apache2/README.Debian.gz for more info.
#   If both key and certificate are stored in the same file, only the
#   SSLCertificateFile directive is needed.
SSLCertificateFile      /etc/ssl/server.crt
SSLCertificateKeyFile   /etc/ssl/server.key
```

Παραμετροποίηση του */etc/apache2/sites-available/**default-ssl.conf*** αρχείου στον virtual host στην port 443.

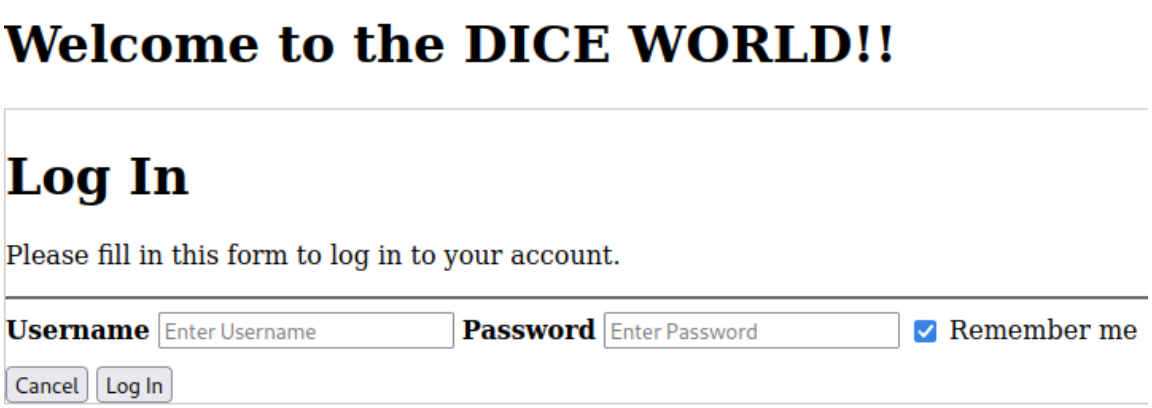
Δημιουργήσαμε ένα ***.htaccess*** αρχείο στο *var/www/<domain name>*

```
1 RewriteEngine On
2 RewriteCond %{HTTPS}  ≠on
3 RewriteRule ^/?(.*) https://%{SERVER_NAME}/$1 [R,L]
```

Τέλος, επιβεβαιώσαμε από τον browser ότι ανακατευθύνεται σε https.

Ερώτημα Γ:

Δημιουργήσαμε μια Log In σελίδα με εισαγωγή Username/Password



Επίσης δημιουργήσαμε μία σχεσιακή βάση δεδομένων gdpr και πίνακα users με στήλες *name, username,password*.

```
create database gdpr;
use gdpr;

create table users (
        name varchar(20),
        username varchar(20) primary key,
        password char(35)
);

delimiter $$
create trigger hash_password_insert
before insert on users
for each row begin
        set @hash_password = sha2(new.password, 256);
        set @salt = substring(sha2(rand(), 256), 1, 16);
        set @hash_password_salt = sha2(concat(@hash_password, @salt), 256);
        set new.password = concat(@hash_password_salt, concat(':', @salt);
end $$
delimiter ;

delimiter $$
create trigger hash_password_update
before update on users
for each row begin
        if old.password <> new.password then
                set @hash_password = sha2(new.password, 256);
                set @salt = substring(sha2(rand(), 256), 1, 16);
                set @hash_password_salt = sha2(concat(@hash_password, @salt), 256);
                set new.password = concat(@hash_password_salt, concat(':', @salt);
        end if;
end $$
delimiter ;

insert into users (name, username, password) values ('foithtes', 'f3312211f3312217', '12345!@');
insert into users (name, username, password) values ('admin', 'admin', 'admin');
```

Εισάγαμε στον πίνακα users δύο χρήστες. Ο πρώτος με όνομα χρήστη τον αριθμό μητρώου μας και ο δεύτερος με όνομα χρήστη admin. Τα password αποθηκεύονται με χρήση

συνάρτησης hash και προσθήκη τιμής **salt** για αποφυγή rainbow table attacks σε περίπτωση ανάκτησης του αρχείου από κακόβουλο χρήστη.