

Communication LTD STRIDE analysis

We analyzed each one of our vulnerabilities using the STRIDE and DREAD model.

Dread model:

- Damage – how bad would an attack be?
- Reproducibility – how easy is it to reproduce the attack?
- Exploitability – how much work is it to launch the attack?
- Affected users – how many people will be impacted?
- Discoverability – how easy is it to discover the threat?

Each threat in our DREAD model is rated between 1 to 3 in all 5 categories.

The DREAD model says that cyber threats with a rating of five to seven are considered a low risk, while cyber threats with a rating of eight to 11 are medium risk. If a cyber threat has a rating of 12 to 15, on the other hand, it's considered a high risk.

S – Spoofing

1. A user can use XSS in order to insert a JS script to the 'Info' input field on the system page, that will get the other user's session token. This token is used to authenticate the user while talking to the server. This will lead to a position where the attacker can use the stolen token to spoof his identity and delete and add customers as if he were someone else.

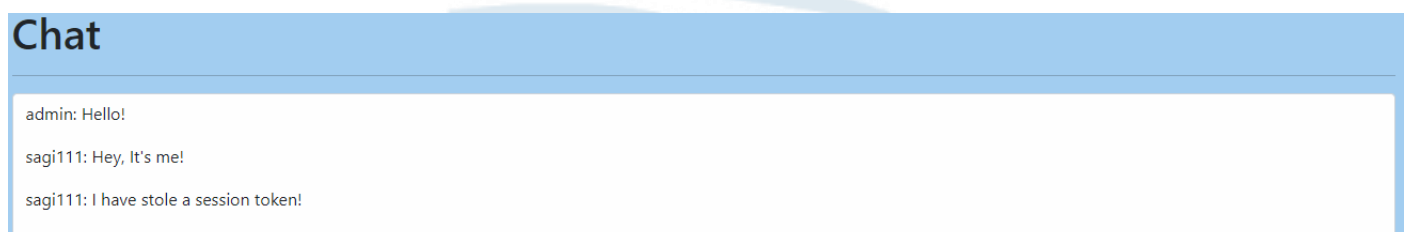


Info

Information about the customer purchases...

Submit Cancel

2. An attacker can use the same method as in #1 and send chat messages to the server with the session token of the victim. This will display the chat message to all the other users like the victim has sent it.



Chat

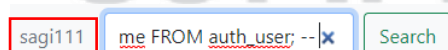
admin: Hello!

sagi111: Hey, It's me!

sagi111: I have stole a session token!

3. Use SQL Injection on the 'Search' field to get all the auth_user table. Then select one user from that table and get his hashed password and username. Using a strong GPU, crack his password (brute force) and use the plain text password and the username to login as this user. Now you can do any operation as this user.

Example:



sagi111 me FROM auth_user; --|x Search

Search Results

0	13	yossi@gmail.com	Yossi Cohen		3G with unlimited calls	8
1	14	Avi@gmail.com	Avi Levi		4G with SMS	8
2	1	sagi313	pbkdf2_sha256\$260000\$IGH2mVvWnGIAomkc0i4hSU\$z8oqE3uBVZoiKJGPN1lkqL4V0EpA/Ba52zfBYjBY2dY=		Sagi	Buria
3	2	lior	pbkdf2_sha256\$260000\$9JATn7uNPBikC9pZMs7JNr\$LgU/1de5BT155Uq+m09ki8pf45Oxp2adCgw8K8U5D/M=		lior	lior
4	3	viko	pbkdf2_sha256\$260000\$GyxheCpFQ0uM6VWPchngF\$6H11xvOoa1ZibabTux6tbT7KbUkOoplMkLuyLwnJ44=		viko	viko
5	4	sagi	pbkdf2_sha256\$260000\$JAlxRe1wTHUOIhRjPvQLDG\$XKgc6PJzinP3I7FSbDA5+Xk3N4xYdVrgbmfkHwWvR8=		sagi	sagi
6	5	vico	pbkdf2_sha256\$260000\$sHhEMnHrcI7ycHWms1NbVm\$IACWck84pLsugFPWzMKq4ITGo4wh0dYRq6LQKvt/wLg=		vico	vico
7	6	admin	pbkdf2_sha256\$260000\$CXTkfz2SF3ur2YBJmociE1\$175xiHit3+ZVgj/f4j5i1S0qW5i1/MmWGo9nObAl3RUU=			
8	7	sagiii	pbkdf2_sha256\$260000\$95ZP4BNHkRJC0S0CJAkxc9\$+sqxdTMxVtlr5AbMePR3TIngMRO680h/bGQtiTaNFy=		sagi	sagi
9	8	sagi111	pbkdf2_sha256\$260000\$WgcVxvgViZqaD57Da5Dbxc\$7Hh6P4LgK50svfGfjqAb7UVIX9P2qjCHT8QW99N3yki=		sagi	buria

- A user can register as another user. Because we do not ask to validate the email address when signing up, a user can insert details of another person, and post chat messages as if he were that person. The only issue is that the victim can take over the account if he uses the 'Forgot Password' function.
- Because the Django admin page is exposed to everyone an attacker can brute-force it in order to login as an admin. The management page in the url '/admin' shouldn't be exposed publicly. Because it is exposed and because we didn't restrict login attempt on this view, an attacker can brute force the passwords and usernames of an admin. Then the attacker will be viewed as the admin.

Threat #	D- Damage Potential	R- Reproducibility	E- Exploitability	A- Affected users	D- Discoverability	Total	Severity
1	2	2	2	1	1	8	Medium
2	1	2	2	1	1	7	Low
3	2	2	1	1	1	7	Low
4	1	2	3	1	2	9	Medium
5	3	1	1	3	2	10	Medium

T – Tampering

- A user can use SQL injection attack vector in the 'Search' bar field in order to tamper with the data in the database. For example, one use case can be, a salesperson at the company that wants to harm his co-worker, can delete all his customers from the DB.
- A non-logged user can use SQL injection in the 'Search' bar field to delete all the database. This can be done from the 'Search' bar input field. With the help of the DROP query, he can delete whatever table he wants.
- A user can use the 'Info' field on the system main page to perform XSS. This user can inject a JS script that sends continuous POST request with spoofed data. Because this script will run from any user that renders that page, this will fill up the table and won't allow other users to view any of the relevant data.
- A user can hijack a session token from another user in order to delete their data. A malicious user can get a hold of another user session token using XSS in the 'Info' field on the main page. The token will be sent to the attacker and then he can use it to send POST request with that token. Those POST requests can include the execution of a deletion of customers in the name of the victim.

Info

Information about the customer purchases...

Submit Cancel

- An attacker can perform SQL injection on the 'Search' bar field to delete audit logs from the database. If an attacker gets the Django_admin_log table from the database, he can delete logs and change them to make the admin think clients have logged in.

Threat #	D- Damage Potential	R- Reproducibility	E- Exploitability	A- Affected users	D- Discoverability	Total	Severity
1	2	2	3	2	2	11	Medium
2	3	2	3	3	2	13	High
3	1	2	1	2	1	7	Low
4	2	2	1	2	1	8	Medium
5	1	2	1	1	2	7	Low

R – Repudiation

1. Using SQL injection on the 'Search' bar field an attacker can insert costumers into the database on the behalf of another salesperson. The system will lack the ability to trace the attacker.
2. A logged user can use XSS attack on the 'Info' field inorder to send POST request of the behalf of the user that is viewing that page. For example, the attacker will insert a simple JS script that sends a POST request to the site's serve and asks to add a new costumer. Now, any user that will render that page will execute this script. And the request will be sent without a way of telling it was the attacker who sent it.
3. A non-logged user can perform an SQL injection through the 'Search' bar inorder to change the admin's password. Because SQL injection will let you modify the data in the database, the attacker can take a simple password, use sha256 to hash it and then push it into the auth_user DB. Then, he can just log-in with the admin username and the simple password he just used.
4. A logged user can use XSS attack vector to insert a JS script in the system page. The script can be insert into the 'Information' or 'Costumer Name' fields. The script will redirect the other users of the site to a malicious domain that contains malware such as a key-logger. This key-logger will be downloaded to the user's machines and will record their password and forward it to the C&C server which the attacker owns. The attacker can then use those details to log-in as admin
5. An attacker can perform XSS attack vector on the 'Info' field in the system page, to insert a JS script that redirect the other clients to a phishing page. The phishing page will look the same as the login page. The clients won't know they are at the wrong site. So, they will enter their credentials. The attacker will take the details and then login as the victim. He can now act as this user.

Threat #	D- Damage Potential	R- Reproducibility	E- Exploitability	A- Affected users	D- Discoverability	Total	Severity
1	1	2	2	1	2	8	Medium
2	1	2	1	1	2	7	Low
3	3	2	3	2	1	11	Medium
4	3	1	1	2	2	9	Medium
5	2	3	2	3	2	12	High

I - Information disclosure

1. Data leakage using SQL injection. A malicious acter can perform an SQL injection attack and get all the user's credentials data base. This database includes email addresses, full names and hashed (and salted) passwords. This information is private and should stay that way.

Example:

The following query has been used:

```
"%' UNION SELECT id, username, password, first_name, last_name FROM auth_user; -- "
```

sagi111

me FROM auth_user: --

Search

Search Results

0	13	yossi@gmail.com	Yossi Cohen	3G with unlimited calls	8
1	14	Avi@gmail.com	Avi Levi	4G with SMS	8
2	1	sagi313	pbkdf2_sha256\$260000\$IGH2mVvWnGIAomkc0i4hSU\$z8oqE3uBVZoiKJGPN1lkqL4V0EpA/Ba52zfBYjBY2dY=	Sagi	Buria
3	2	lior	pbkdf2_sha256\$260000\$9JATn7uNPBlkC9pZMs7JNr\$LgU/1de5BT155Uq+m09ki8pf45Oxp2adCgw8K8U5D/M=	lior	lior
4	3	viko	pbkdf2_sha256\$260000\$GyxheCpFQ0uM6VWpChngF\$6H11xvkOoa1ZibabTUx6tbT7KbUkOoplMkLuyLwnJ44=	viko	viko
5	4	sagi	pbkdf2_sha256\$260000\$JAlxRe1wTHUOIhRjPvQLDG\$XKgc6PJzinPI3I7FSbDA5+Xk3N4xYdVrgbmfkHwWoR8=	sagi	sagi
6	5	vico	pbkdf2_sha256\$260000\$sHhEMnHrcI7ycHWms1NbVm\$IACWck84pLsugFPWzMKQ4ITGo4wh0dYRq6LQKvt/wLg=	vico	vico
7	6	admin	pbkdf2_sha256\$260000\$CXTkfz2SF3ur2YBJmociE1\$175xiHit3+ZVgj/f4j5i1S0qW5i1/MmWGo9nObAl3RUU=		
8	7	sagiii	pbkdf2_sha256\$260000\$95ZP4BNHkRJC05CJAlkxc9\$+sqxdTMxvTlrI5AbMePR3TIngMRO680h/bGQtiTaNfy=	sagi	sagi
9	8	sagi111	pbkdf2_sha256\$260000\$WgcVxvgViZqaD57Da5Dbxc\$7Hh6P4LgK50svfGfjqAb7UVIX9P2qjCHT8QW99N3ykl=	sagi	buria

- Data leakage using SQL injection. A malicious acter can perform an SQL injection attack and get all the costumer's data. This database includes their email addresses, and information about their purchases.

Example:

The following query has been used: "%' -- "

AnonymousUser

%' --

Search

Search Results

0	13	yossi@gmail.com	Yossi Cohen	3G with unlimited calls
1	14	Avi@gmail.com	Avi Levi	4G with SMS

- The client's IP can be exposed using XSS attack vector. An attacker can insert a JS script into the 'Message' field that will send the client's IP address to a C&C server.

Example:

We can craft a script that send a GET request to <https://api.ipify.org/?format=json%22>. This API will return the IP of the client. Then, the script will send the POST request to the C&C server with that IP

Message

Submit Cancel

- An attacker can perform XSS attack vector on the 'Info' field in the system page, to insert a JS script that redirect the other clients to a phishing page. The phishing page will look the same as the system page, that was, the clients won't know they are at the wrong site. So, they will enter new costumers to the system. This data will be passed to the attacker instead of the original server.
- A malicious user can use the chat app on the site ('/chat') to use social engineering to make other clients give him their information. The attacker can create a user with a username like 'Important And Scary Admin'. This username will give him credibility in the chat, and users can think he is the admin and give him personal data.

Threat #	D- Damage Potential	R- Reproducibility	E- Exploitability	A- Affected users	D- Discoverability	Total	Severity
1	2	3	3	2	1	11	Medium
2	3	3	3	3	1	13	High
3	2	2	2	3	2	11	Medium
4	2	3	2	3	2	12	High
5	1	2	1	1	2	7	Low

D - Denial of Service

1. An attacker can lock users out of their accounts by typing wrong passwords. Because of the password attempts limitation, that blocks a user for X amount of time when he is unable to login, there is a possibility that an attacker will guess usernames and type wrong passwords for them, which will lead to their account getting blocked

Example:

An attacker can use the Hydra tool to perform this. Hydra lets you brute force web pages easily.

2. Lack of CDN usage can cause a vulnerability of DDOS attacks in Layer 3 and 4. Layer 7 is also vulnerable, but this can also occur when using CDN.
3. An attacker can use SQL injection to delete all the database. This can be done from the 'Search' bar input field. In case there is no backup, this will result a denial of service, because users won't be able to log-in to the system. In order to do this, he will first need to get all the table's names and drop the correct one.

Example:

```
"%'; DROP TABLE auth_user; -- "
```

4. A non-logged user can use the Chat feature of the app to perform a XSS attack. Using this attack vector, the attacker can insert a simple JS that redirects the user to another page. This means that every user that will go into the chat app, will be automatically redirected to another page, making the chat app useless.

Example:

With the following script: `<script> window.location.replace("http://google.com");</script>`


Every client that will go to the chat app will be redirected to google.

Message

`<script> window.location.replace("http://google.com");</script>`

Submit Cancel

We can now see that every user will be redirected to Google.com.

 google.com

5. A user can spam the DB. In both the chat app ('/chat') and the system page ('/') there is no limit to how many inputs one user can do. A user can spam the database with many new records which won't allow the legitimate data to be displayed.

Threat #	D- Damage Potential	R- Reproducibility	E- Exploitability	A- Affected users	D- Discoverability	Total	Severity
1	1	2	2	2	3	10	Medium
2	1	1	2	2	2	8	Medium
3	2	2	2	2	3	11	Medium
4	2	3	3	2	2	12	High
5	1	2	1	1	2	7	Low

E - Elevation of Privilege

1. A logged user can use SQL injection in order to get the user credentials DB. This will give him the admin's hashed (and salted) password. Using a strong GPU, he can then crack that password and get into an admin account.

Example-

The following query has been used:

```
"%' UNION SELECT id,username, password, first_name, last_name FROM auth_user; -- "
```

sagi111

me FROM auth_user; --

Search

Search Results

0	13	yossi@gmail.com	Yossi Cohen	3G with unlimited calls	8
1	14	Avi@gmail.com	Avi Levi	4G with SMS	8
2	1	sagi313	pbkdf2_sha256\$260000\$IGH2mVvWnGIAomkc0i4hSU\$z8oqE3uBVZoiKJGPN1lkqL4V0EpA/Ba52zf8YjBY2dY=	Sagi	Buria
3	2	lior	pbkdf2_sha256\$260000\$9JATn7uNPBlkC9pZMs7jNr\$LgU/1dE5BT155Uq+m09ki8pf45Oxp2adCgw8K8U5D/M=	lior	lior
4	3	viko	pbkdf2_sha256\$260000\$GyxeheCpFQ0uM6VWPchngF\$6H11xvKOa1ZibabTUx6tbT7KbUkOoplMkLuyLwnJ44=	viko	viko
5	4	sagi	pbkdf2_sha256\$260000\$JAlxRe1wTHUOIhRjPvQLDGSXKgc6PjzinPI3l7FSbDA5+Xk3N4xYdVrgbmfkHwWoR8=	sagi	sagi
6	5	vico	pbkdf2_sha256\$260000\$sHhEMnHrcI7ycHWms1NbVm\$IACWck84pLsugFPWzMKQ4ITGo4wh0dYRq6LQKvt/wLg=	vico	vico
7	6	admin	pbkdf2_sha256\$260000\$CXTkfz2SF3ur2YBjmociE1\$175xiHit3+ZVgJf4j5I1S0qW5i1/MmWGo9nObAl3RUU=		
8	7	sagiii	pbkdf2_sha256\$260000\$95ZP4BNHkRJC0S0CjAlkxc9\$+sqxdTMxTlri5AbMePR3TIngMRO680h/bGQtiTaNFY=	sagi	sagi
9	8	sagi111	pbkdf2_sha256\$260000\$WgcVxvgViZqaDS7Da5Dbxc\$7Hh6P4LgK50SvfGFjqAb7UVIX9P2qjCHT8QW99N3yki=	sagi	buria

- A non-logged user can do the same (as #1), and perform an SQL injection to get users and admins credentials, crack their hashed passwords and log-in as a user or as admin
- A logged user can use XSS attack vector to insert a JS script in the system page. The script can be inserted into the 'Information' or 'Customer Name' fields. The script will redirect the other users of the site to a malicious domain that contains malware such as a key-logger. This key-logger will be downloaded to the user's machines and will record their password and forward it to the C&C server which the attacker owns. The attacker can then use those details to log-in as admin
- A non-logged user can perform an SQL injection through the 'Search' bar in order to change the admin's password. Because SQL injection will let you modify the data in the database, the attacker can take a simple password, use sha256 to hash it and then push it into the auth_user DB. Then, he can just log-in with the admin username and the simple password he just used.
- A logged user can use SQL injection in the 'Search' bar to alter his privileges. If a user gets the auth_user table, he will see the is_staff column, which will allow him to change his own record to '1', and make him an admin instead of a regular user.

Threat #	D- Damage Potential	R- Reproducibility	E- Exploitability	A- Affected users	D- Discoverability	Total	Severity
1	1	2	1	2	3	9	Medium
2	2	2	1	2	3	10	Medium
3	3	1	1	2	2	9	Medium
4	3	2	3	2	1	11	Medium
5	2	2	2	1	1	8	Medium