HPY 413
Assignment 3
Code Injection Attacks on the Web

msiotos 2016030030
gangelopoulos 2016030083

Σκοπός της άσκησης ήταν ο εντοπισμός 2 SQL injection vulnerabilities, 1 reflected XSS vulnerability και 1 DOM-XSS vulnerability και μας ζητήθηκε να τα κάνουμε exploit στον real server.

2 SQL injection vulnerabilities

Αρχικά δουλέψαμε για τον mock server 'http://127.0.0.1:8080'. Παρατηρώντας τον κώδικα στο αρχείο app.py, εντοπίσαμε ότι στη γραμμή 41, το user input (password) εισέρχεται απευθείας μέσα στο query string χωρίς να έχει γίνει sanitized. Αντικαταστήσαμε λοιπόν το {password} με το payload *'OR 1=1 --'* και δούλεψε στο mock application. Δοκιμάσαμε τον ίδιο κωδικό και στον user του real server και πράγματι μας δόθηκε πρόσβαση.

Για το δεύτερο vulnerability, εντοπίσαμε ότι στη γραμμή 82 η μεταβλητή name εισέρχεται απευθείας και αυτή στο query string. Οπότε αντικαταστήσαμε στη γραμμή 75 το name = request.form["item_name"] με το payload: *name = " ' UNION SELECT username, password, null FROM users -- "* έτσι ώστε να αποκτήσουμε πρόσβαση σε sensitive information. Πράγματι, μορφοποιώντας τον κώδικα πήραμε πρόσβαση στον κωδικό του admin-superadmin (το οποίο προφανώς εμείς ήδη γνωρίζαμε από την εκφώνηση). Χρησιμοποιώντας το παραπάνω string στο "Search for an item" field του live server, μας δόθηκε ο κωδικός του admin που ήταν *$thisIsUncrackable$* .
Αφού κάναμε login ως admin με τον παραπάνω κωδικό πήραμε το secret flag:
TUC{55987a25fd535d2f114ff3a35da038b47047d9fe94bedd199bcc7ec15964fd21}
Αξίζει να σημειωθεί ότι κάθε φορά παίρναμε διαφορετικό flag, αλλά αυτό ήταν το πρώτο.

1 XSS vulnerability

Για το reflected XSS, αφού πήραμε access στον user όπως προαναφέρθηκε, όταν μας ζητήθηκε να κάνουμε search an item, κάναμε inject το payload *<script>alert("XSS")</script>* και επιβεβαιώθηκε κανονικά το vulnerability γιατί μας πέταξε το παραθυράκι με το alert. Το vulnerability ξεκινάει από το line 44 του dashboard.html. Αξίζει να σημειωθεί ότι δοκιμάσαμε αρκετά διαφορετικά payloads, αλλά αυτά με ερωτηματικά(;) δεν πέρασαν, γιατί απαγορεύονται στο συγκεκριμένο search field.

1 DOM-XSS vulnerability

Για το DOM-XSS, κοιτάξαμε μεσα στο greet.js αρχείο. Χρησιμοποιήσαμε το payload: *<script>alert('DOM-XSS')</script>* για να δούμε αν θα μας πετάξει παραθυράκι όταν τρέξει ο κώδικας γιατί βάζουμε άμεσα δικά μας data στην document.write, η οποία δεν είναι αρκετά ασφαλής. Για τον real server, καναμε inject κατευθείαν αυτό το URL:

139.91.71.5:11337/dashboard#<script>alert(‘DOM-XSS’);</script> ώστε να δούμε αν θα τρέξει το script και επιβεβαιώθηκε ότι υπάρχει το συγκεκριμένο vulnerability, αφού εμφανίστηκε το alert στην οθόνη μας.