**Conquest Web Application Hacker**

Daniel Zhang, Abhi Rathod, Tanner Harper, Asad Mahdi

May 5, 2017

**Summary**

Conquest is a web application attack tool that takes a url input and automates reconnaissance, vulnerability testing, and exploitation on the target address. Our app seeks out live web pages belonging to the target and attempts to exploit discovered cross-site scripting (XSS) and SQL injection vulnerabilities found on the page. There is an option for an authenticated execution, where the user provides valid login credentials for further website access.

Example usage:

python3.6 WebHack.py <website url>

python3.6 WebHack.py -u <username> -p <password> <website url>

**Step 1: Wide Reconnaissance**

Conquest uses two different forms of reconnaissance to gather the live pages on a target. The first Conquest module runs a crawler that recursively follows all hypertext links beginning at the url that was input. In addition to the crawler, the module also utilizes a common files and directories list to forced browse the target website, observing successful responses and adding them to the list of known live pages.

If username and password flags are passed, Conquest will attempt to look for login pages based on observed password fields and post the input credentials to the page. Once authenticated, conquest crawls again to discover the pages only available to logged in users.

**Step 2: Identifying Vulnerabilities**

Once all the live pages are gathered, a second module begins to probe each web page to see if they have vulnerabilities. At this stage, Conquest parses the victim pages’ html to identify all the input fields. The module then inserts multiple test scripts and injection strings into the fields and posts the page to observe the response from the server. The pages that exhibit vulnerabilities are marked for exploitation.

**Step 3: Exploitation**

Conquest offers a range of XSS and SQL exploits to users.

* Iframe exploit - This stored XSS exploit is set up for comment fields typically seen on public forums and blogs. Conquest will scrape the login page of the targeted website and display it in an iframe over the exploited page. Any credentials entered into the iframe are output to a log file.
* Redirect exploit - A more general exploit, this is another stored form of XSS that allows the user to cause an exploited page to execute a redirect as soon as it it requested. The redirect can be customized to any url the user needs.
* Table enumeration - After finding a SQL vulnerable field, Conquest can attempt to incrementally enumerate the table’s properties to obtain information such as database, table, and column names. The user is then then given the option of printing out the contents of the database.
* Authorization exploit - This SQL injection exploit attempts to input attack strings into username and password fields to bypass authorization. If successful, the user can then take these strings and use them in place of username and password.
* Cookie exploit - A victim that visits a page compromised by this XSS exploit will have their cookie sent to the Conquest user. The cookie can be used for stealing the victim’s browser sessions.

**Difficulties Encountered**

While the reconnaissance and vulnerability checking modules performed effectively, Team 4 experienced much frustration with SQL exploitation. We knew automating SQL injection would be difficult, but after attempting to integrate it into our tool, we realized exactly how nuanced enumerating just a lightly protected database and accessing information is. Even powerful tools like sqlmap must have output monitored closely by a human and tweaked tamper scripts carefully tweaked to outwit protections like filters, whitelists, and blacklists. In the end, we were able to implement only basic injection testing and exploitation.

Coming up with payloads like SQL injection strings and XSS scripts was also more difficult than expected. It is simple to create specific payloads tailored to our demo website, but making a payload that worked for most vulnerable pages took a lot more effect, especially given our team’s lack of scripting experience.

**Future Improvements**

* Currently, our SQL exploits only work for SQLite database implementations. The first thing we would attempt is to expand our tool’s compatibility with other database systems like MSSQL, MySQL, or PostgreSQL.
* The nature of Conquest allows for indefinite expansion. As long as new exploits continue to be discovered, we can add them into the tool. We can increase its depth by including more SQL and XSS exploits, or we can expand Conquest’s breadth by exploiting more kinds of vulnerabilities such as command injection of remote and local file inclusion.
* Currently, the different kinds of exploits use a combination of messy terminal print statements and output log files to record information.   
  We feel that our tool would be suited to a graphical user interface down the road.