Chad Ballay

CYBR350-342N

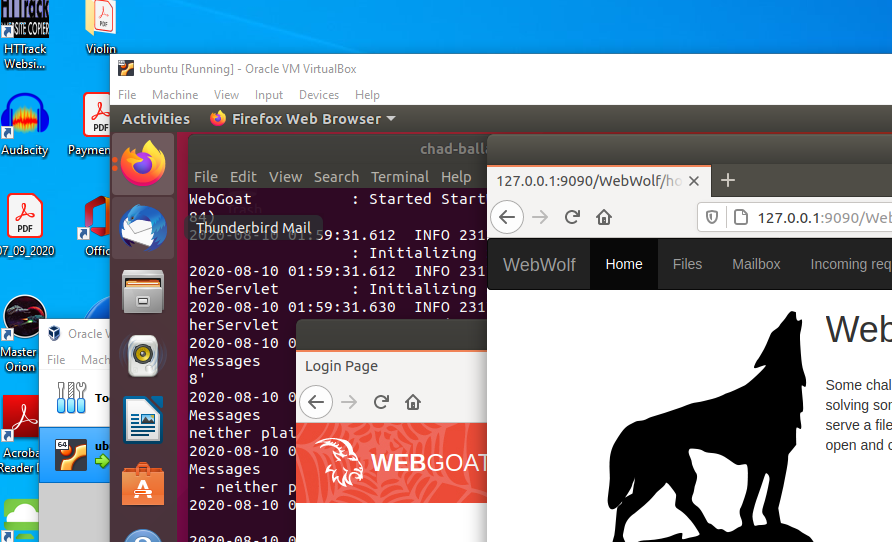
Week 6

08/08/2020

WebGoat – Journal

## Preface

I did end up giving up on my Toxic environment. There seems to be some vital step that I’m failing at. Spun up an Ubuntu VM with VirtualBox and am now running release 8.1.0 locally.



Since this evening has been oodles of fun with learning some networking stuff and troubleshooting VM’s I figured that I should also work through the WebGoat milestones from scratch.

# Introduction

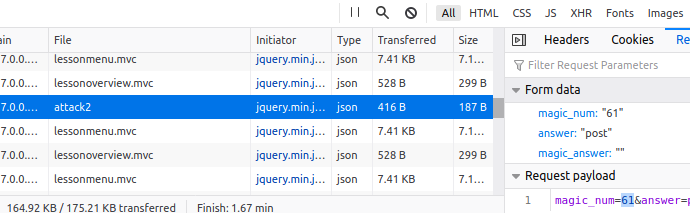


Not much to do here for WebGoat. The WebGoat intro was a good recap.

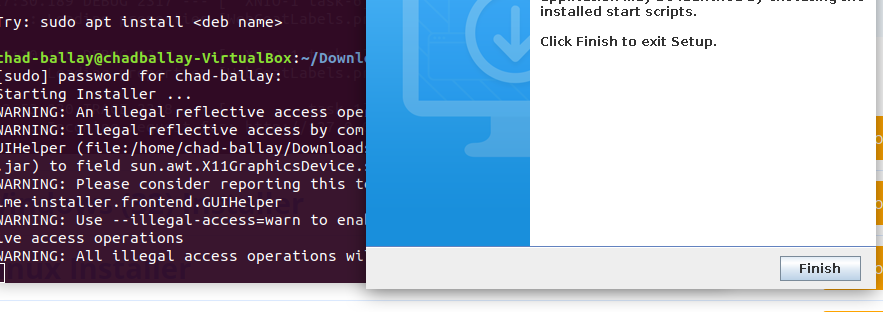
The WebWolf introduction included hands on which made for a much more compelling introduction. Was able to complete both demo’s but they were little more than cut/paste exercises. They highlighted the potential and that is still huge on the wow factor.

# General

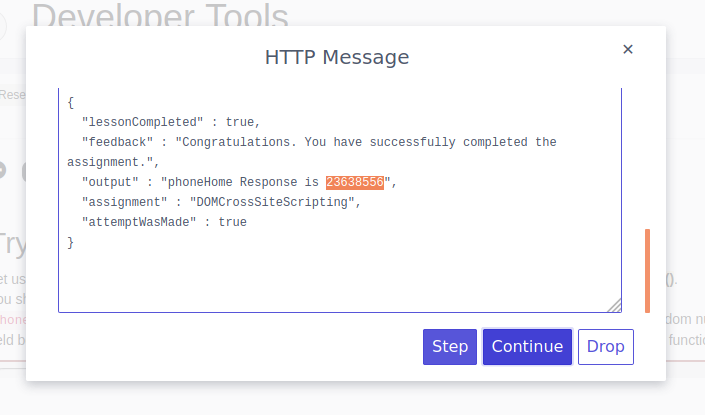
This one narrows the focus down more. The HTTP Basics seems to have an open github issue that was a head scratcher for me. I wanted to see how else to solve this besides just hitting Developer’s Tools and spying on the request. The hint mentioned turning on Show Parameters. That button doesn’t exist anymore I guess. (<https://github.com/WebGoat/WebGoat/issues/417>) But was able to spy on the POST and see the magic number.



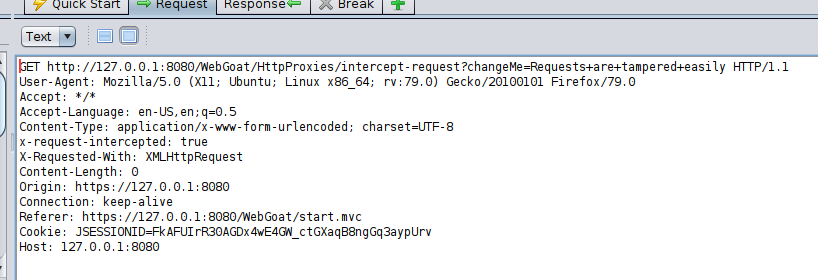
Getting Zed installed was also another achievement by the OWASP team. Documentation was clear and ended up being a smooth install process.



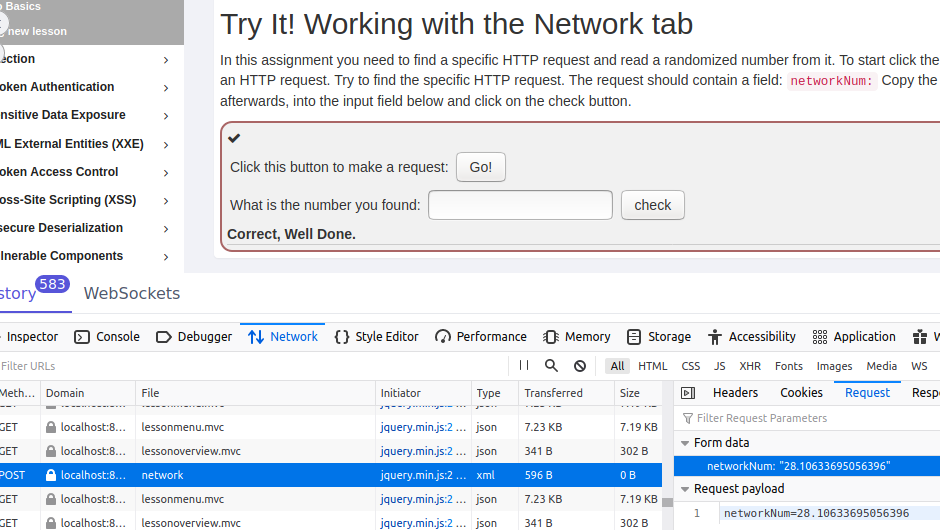
The breakpoints walkthru was really helpful. Often times being able to step through a function with the values it has being agreeable to troubleshooting has been a real boon.



Modifying the POST to a GET was straight forward as was adjusting the URL. It took more times than I care to count to tweak the header. The takeaway for this one is that you can’t control what the users will try so you must default to ban/block/ignore everything but a very explicitly enumerated subset of inputs.



The developer tools section highlights that if you don’t control the environment that the code is running on then you will have a rough time controlling what the user does while the code is running. The console allows for web browsers to interrogate the system for all sorts of functionality. Most of the times my experience has been for good but I wonder what bugs I’ve not noticed while playing around with the console or looking at all the http request/response calls.

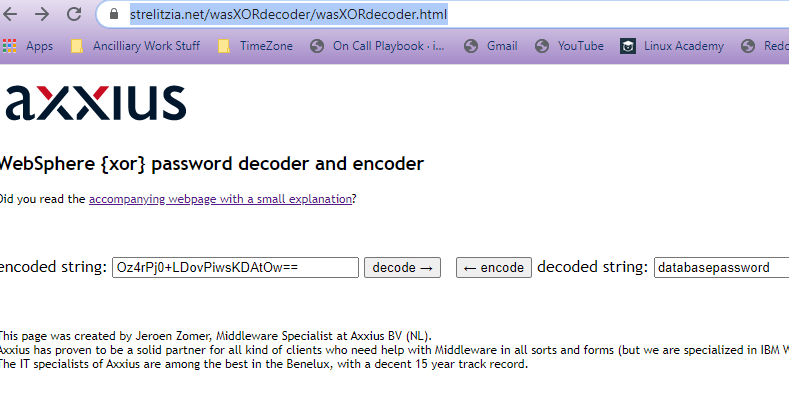


CIA Triad section thankfully was already fresh in my mind. Got SEC+ a month or so ago and Professor Messer and Mike Meyers videos helped hammer that one in. That and hearing the various incarnations of examples around Alice and Bob scenarios that started with Ron Rivest.

* Confidentiality – Only those that should have access, do.
* Integrity – The information contained is the information received.
* Availability - This one.... This is the one that my previous career has been the focus around. How do we make systems reliable while still being as much a single source of truth. CAP theorem, chaos testing, load balancing, etc, etc... This is a tough nut to crack since people speak in absolutes but what they really mean is akin to Xeno’s paradox. He’ll never get to the wall but for all intents and purposes he gets close enough. Same with availability. The more 9’s you strive for the sooner your budget of manpower, money, and laws of physics run out of wiggle room.

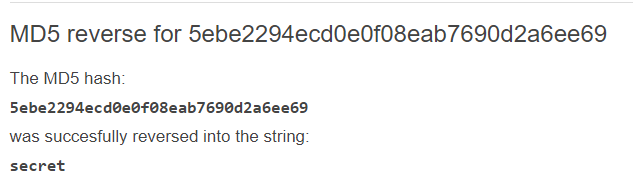


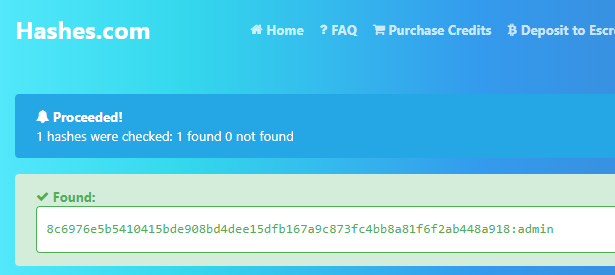
Oh lord... Security through obscurity... So at my workplace we have some automated scanning that occurs. Looks for PII and other easily identified formatted data. Teams will have Splunk tokens, API keys, userid/passwords, etc, etc... in their source code commits. The scanner will catch these items and all too often the team’s response is to base64 it since the scanner doesn’t do that step. The gulf between the letter of the policy and the spirit of the policy is a wide one for some damn fools. It was heartening to see that this is common enough to end up being part of WebGoat. Misery loves company



The Base64 one struck a nerve. Arguing with some teammates about the use of security through obscurity over some data they want to save. It’s amazing how often people think a little bit of extra math and all of a sudden their data is secure. Encrypted text vs hashed text vs transformed text all have their places. Using a transformative function like base64 to just obfuscate the text from being human readable does nothing to actually make things more secure.

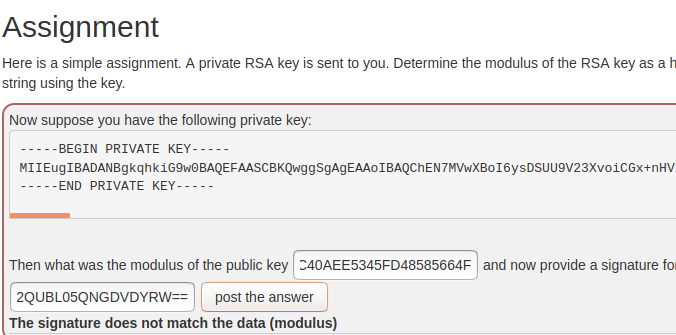
The next two were right easy but required some source code reading and other external websites. They were very helpful to label it md5sum and sha256sum. Based on that I was able to find several sites that brute forced that for me. I need to learn what it was that they were doing and implement that myself in python or some other language as a hands on proof of concept. I just didn’t have enough time to do this.



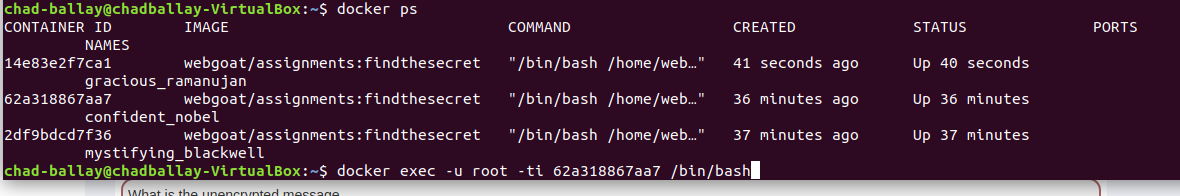


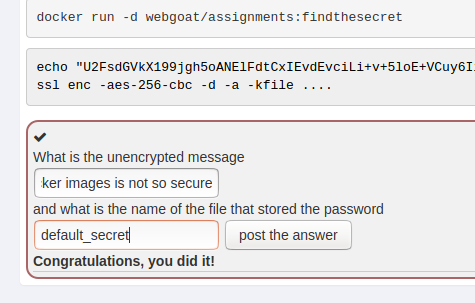
The signature/modulus one was a tough one. I’m showing my age but I always loved this scene in Sneakers how it bootstraps the audience into seeing how cryptography is just math. Really hard math that involves operations that are easy in one direction but exceedingly complex in the opposite direction. <https://youtu.be/F5bAa6gFvLs?t=96>

As for this one. I was way off on my approach. I knew it would involve calculating from the private key but I had no idea around how to do so. According to a writeup that I finally gave in and looked at, OpenSSL was the tool I should have went for. I get the concept but I spent a couple of hours trying to figure out how to convert ascii into binary and then calculating P and N. That was fun but unsuccessful.



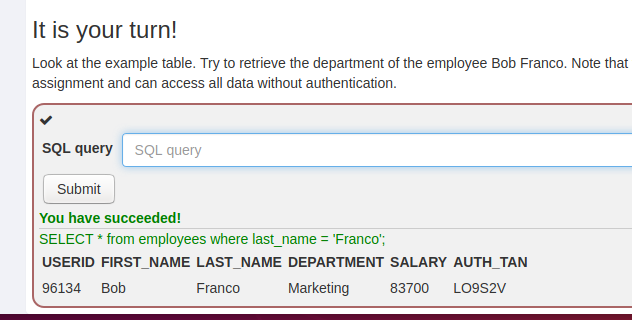
The docker one was interesting. Using the hints it was straight forward to see we would abuse the container to get root. <https://stackoverflow.com/questions/25845538/how-to-use-sudo-inside-a-docker-container> So getting root was straight forward and a concrete reminder that containers don’t magically make things more secure.



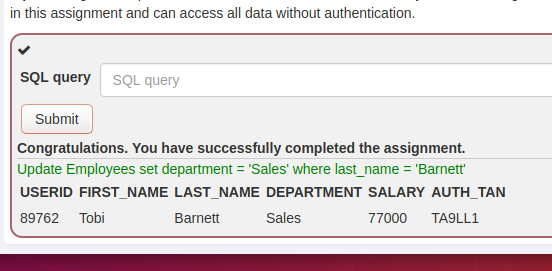


# SQL Injection (Intro)

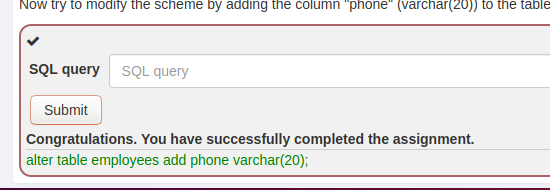
Starts off with a bang and needs us to do a straightforward SELECT statement.



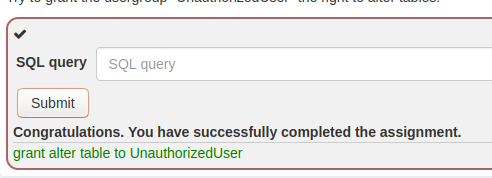
Followed up by a basic UPDATE statement. The trickiest part here was remembering the UPDATE statement syntax.



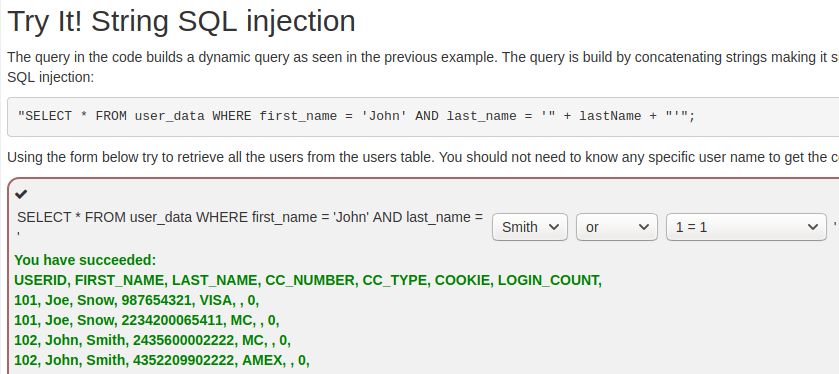
Next came an alter table exercise.



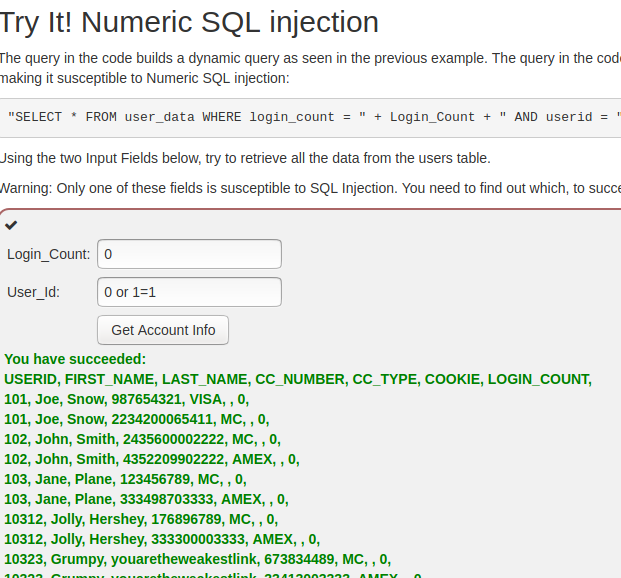
And granting privileges to a user.



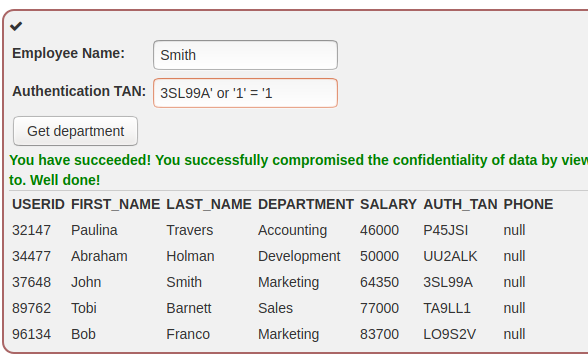
The first exercise involving SQL Injection was a great example. Using the drop downs provided it helped to intuit that we would have quotes in the where clause with the quote around the whole query.



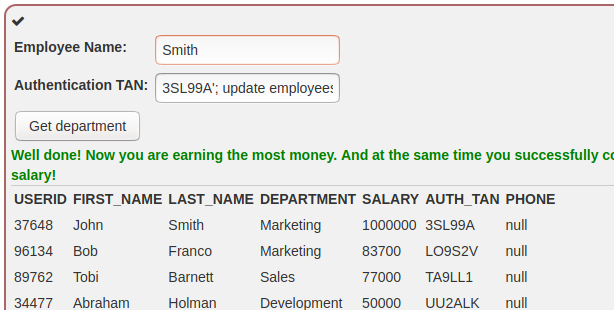
The next exercise was a bit more freeform and took a bit more pondering. The warning was useful since it narrowed the attack to ½ the work. Just had to pick the right field. On a hunch I started with the idea that most SQL Injections occur at the edge of the where clause so I started with the last field. Lucked out. From there it was just piecing together how to add a 1=1 type additional clause.



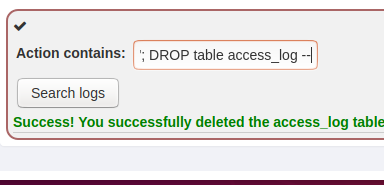
Same with the next one. The intent is to end the input for the value of a variable but instead we short circuit that then add more items to the where clause that makes it more inclusive to returning critical data sets.



This next one highlighted how ingenious you must be. I wouldn’t think of much of these approaches until they have been pointed out to me. The idea of prematurely ending one query and then adding a second query is obvious but it wasn’t until I saw someone do it that I understood what it was that could be done.

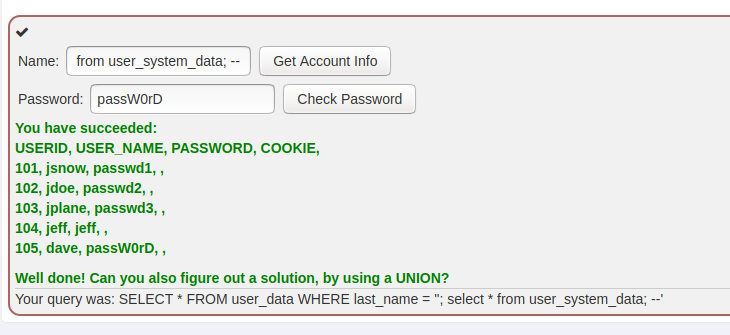


The final Introduction exercise I kind of disagreed with the restrictiveness of the allowed solutions. They wanted us to DROP the table, but I believe a DELETE on all records could also be workable.



# SQL Injection (Advanced)

So at this point they have us attempting to dump a table unreleated to the one being used by the query. For this I used the same pattern of short circuit the existing query, then append the next query that you actually want ran. The results pages will have a few extra pages but for the most part this seems ubiquitous.



At this point I hit a blocker. I need a better understanding to figure out how to get a list of tables in the database. Maybe an ER diagram of their database. But without knowing how to get a list of tables it will take more information that I have on hand. The end goal though is to chain a query on to the text being submitted so that I can run the arbitrary SQL I want.

