CPRE 492 Bi-Weekly Report #5

Report Period: March 15th - March 29th

Group Number: 16

Project Title: Cloudflare WAF AI

Client/Advisor: Cylosoft; Dr. Yong Guan, Dr. Berk Gulmezoglu

Team Members:

Ryan Burgett: Meeting Leader
Giovanni Mejia: Report Manager
Jordan Heim: Documentation Manager
Eric Reuss: Documentation Manager
Presiian Iskrenov: Meeting Scribe
Benjamin Fedderson: Stakeholder Correspondent

Summary:

During this period, we have had a change in how the project was originally heading. After a meeting with our client, we have decided to focus solely on the Console Application and the Database of the project. The reasons were for lack of training data for the AI and most of Cloudflare's APIs cover a good basis of what we would be training our AI model in the first place. We are in the same groups but working on different things. Now the previous 4 are working on the Database and hard-coding detections for malicious attacks. We are wanting to account for the same things that the AI would have been trained on. The project shell required reformatting and was modified for the team to make contributions based on their selected attacks.

Past weekly Accomplishments:

The Database was reformatted to allow for static-scanning with the methods that we want to implement. Each member from the former AI team took over a discussed method to code within our database. Eric and Ryan were able to integrate a functional reading function so the Database can connect and react with our implemented methods. Ben has been working on filter features with the Console Application to ensure full functionality and final touches for that portion of the project.

Pending Issues:

Currently no pending issues.

Individual Contributions:

Team Member:	***Individual Contributions***	***Hours this week***	***Hours Cumulative***
Ryan Burgett	Implemented flooding attack detection with the database scanner. Integrated Connection within Database read the numerous log files.	8 hours	34 hours
Giovanni Mejia	Implemented SQL injection attack detection within the Database Scanner.	6 hours	33 hours
Jordan Heim	Implemented Admin-Attack detection within the Database Scanner.	6 hours	30 hours
Eric Reuss	Reformatted project shell for database and integrated connection with database to read the numerous log files.	7 hours	33 hours
Presiian Iskrenov	Implemented OS Console attack detection within the Database Scanner.	6 hours	32 hours
Benjamin Fedderson	Finalized filter functionally within the developer Web Application.	7 hours	37 hours

Plans for the upcoming week:

Test our new database scanner methods against the massive amounts of log-file data. We want to make sure the pushed code works with Cylosoft servers and its services. Of course as the semester is coming close to an end, we need to start finalizing touches on both the Database and Console Application. This includes filter features with the Application and Database testing with our new methods.

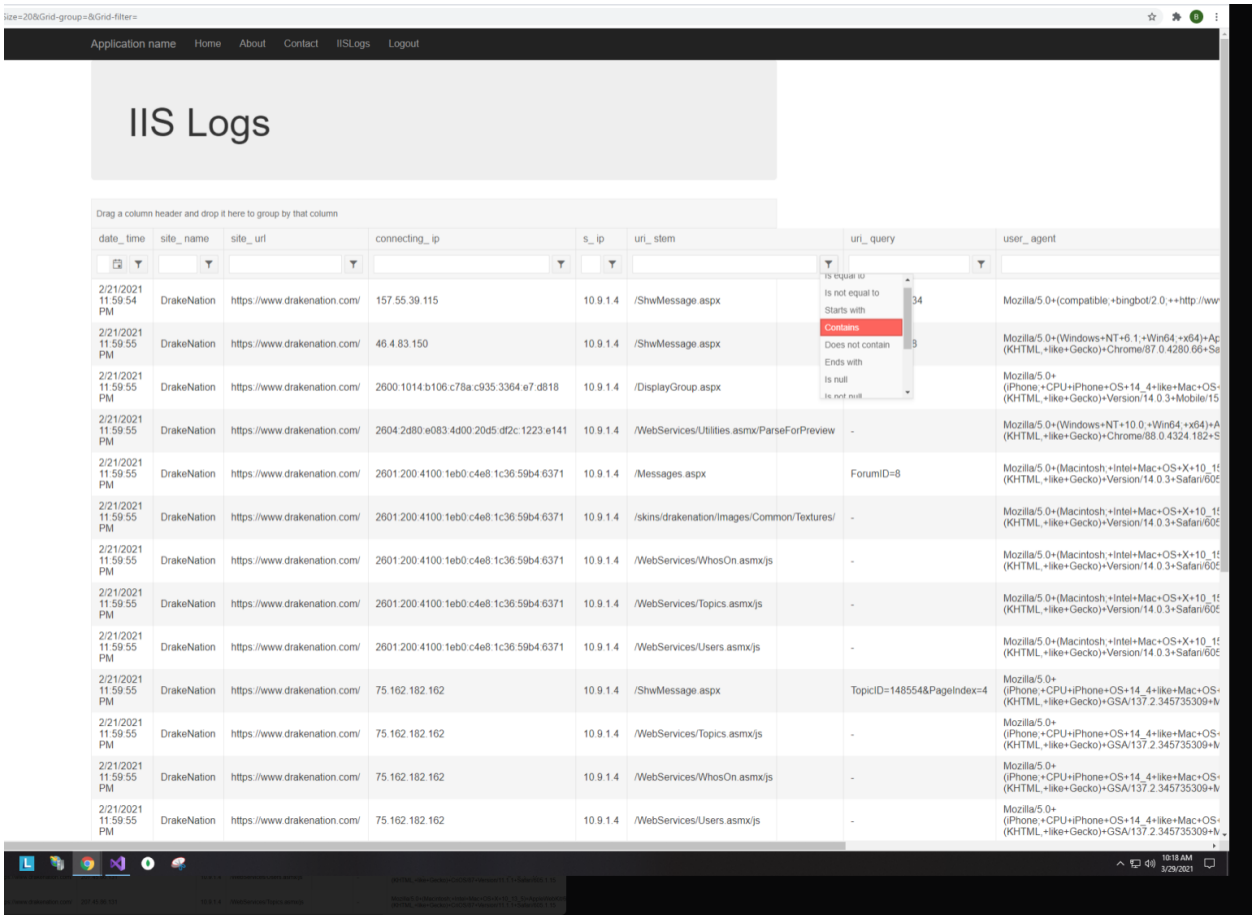
Screenshots from the work this report period:

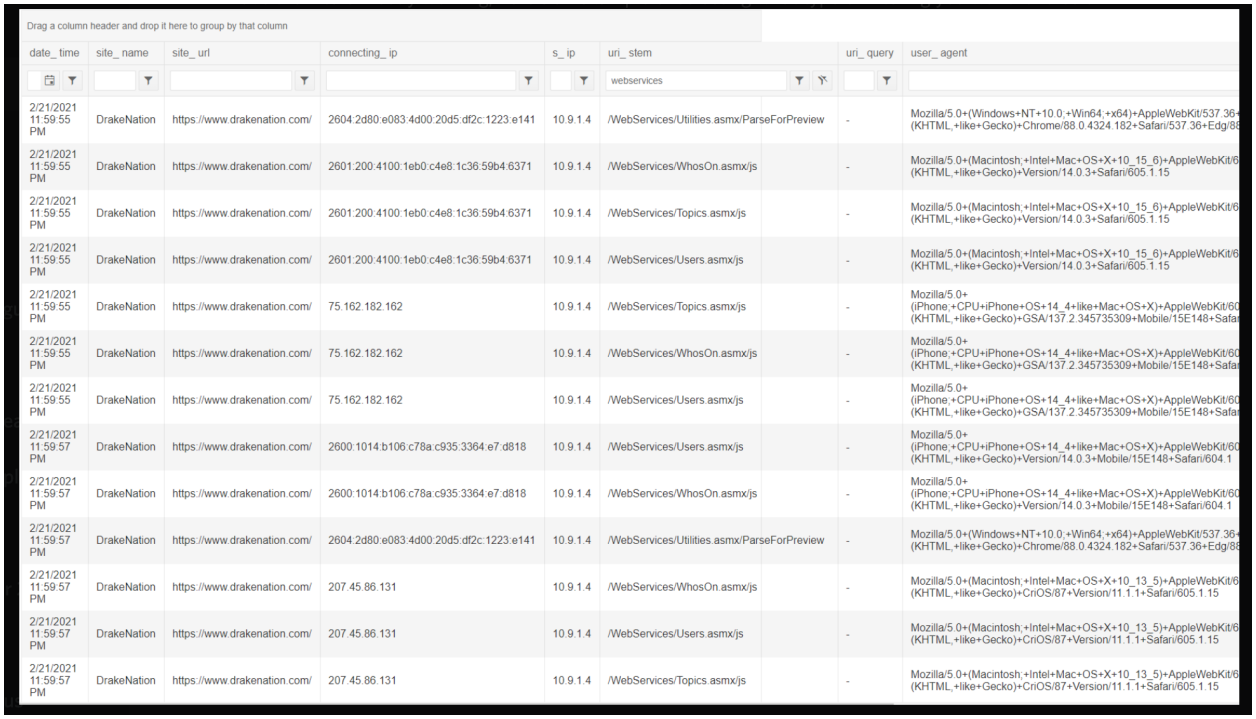
Database Scanning Implementation with various methods.

```
25
26          public List<string> SQLInjectionDetection()
27          {
28
29              List<String> ip = new List<String>();
30              foreach (Data d in data)
31              {
32                  // Basing off what Ryan had in the AI folder. I also researched some common methods of SQL attacks. Attacks vary from Error, Union, and
33               if(d.CsUserName.Contains("1=1") || d.CsUserName.Contains("or=") || d.CsUserName.Contains("DROP TABLE") || d.CsUserName.Contains("INSERT")
34                  {
35                      ip.Add(d.CIP);
36                      Console.WriteLine("Date: \t" + d.Date.ToString() + "Username: \t" + d.CsUserName + "Exact IP: \t " + d.CIP);
37                  }
38              }
39              return ip;
40          }
41
42          public List<string> AdminAttackDetection()
43          {
44
45             List<String> ip = new List<String>();
46              foreach (Data d in data)
47              {
48                  //most webpages have some variation of "admin" or "login" in their login pages Contain() should find them with just "admin"
49                  //examples found in table:
50                  if(d.CSUriStem.Contains("admin") )
51                   {
52                     ip.Add(d.CIP);
53                    Console.WriteLine("Date: \t" + d.Date.ToString() + "Username: \t" + d.CsUserName + "Exact IP: \t " + d.CIP);
54                  }
55                  //optional filter for login pages.
56                  // most webpages have some variation of "admin" or "login" in their login pages Contain() should find them with just "admin"
57                  //examples found in table: login.aspx
58                  if (d.CSUriStem.Contains("login"))
59                  {
60                      ip.Add(d.CIP);
61                      Console.WriteLine("Date: \t" + d.Date.ToString() + "Username: \t" + d.CsUserName + "Exact IP: \t " + d.CIP);
62                  }
63              }
64              ip.Add("1.1.1.1");
65              return ip;
66
67          }
68
69          public List<String> OSConsoleAttackDetection()
70          {
71              List<String> ip = new List<String>();
72              foreach (Data d in data)
73              {
74
75                 if(d.CSUserAgent.Contains("Linux"))
76                  {
77                      ip.Add(d.CIP);
78                      Console.WriteLine("Date: \t" + d.Date.ToString() + " USER_AGENT: \t" + d.CSUserAgent + "IP:\t " + d.CIP);
79                  }
80
81
82              }
83
84              // SqlCommand query = dt.SetupTextQuery("SELECT DISTINCT [cs(User-Agent)],[c-ip] FROM LogTable WHERE ([date-time] > DATEADD(minute, -2,GETU
85
86
87              return ip;
88          }
89      }
90  }
```

This was gathered from our Git-Lab Repo. This screenshot includes that various methods that were implemented for our static database scanner. This includes SQl injection, Admin attacks, and OS Console based attacks. SQL injection will be based on the user-input and will output corresponding IP addresses that meet the criteria. This follows for the other two attacks as well. Although with admin detection, we want to know which IP addresses tried logging into the database with Admin privileges. Finally with a OS Console attack, each user will have a different Operating System when logging in. This helps to detect the type of attack and which type of OS(Mac, Windows, Linux... etc) the attack came from.

Filter Features with the Web Application in development with Cylosoft.





These two presented screenshots are things that Ben has been working on within the Application. The first screenshot shows his implementation of the filter feature within all our massive amount of log data. Each column has a corresponding filter feature so any data will be easier to access. Then finally, the last screenshot shows the feature actually being used in the “uri-stem” column. It is filtering the data so that it will only show the keyword “web services” in all the IIS log file data.