AIC Training Module

**Title**: Finding Vulnerable Telerik Instances with Scytode.py and Telerik\_RCE\_Scan.py  
**Architecture**: NA  
**Environment**: NIX Environment Preferred  
**Dependencies**: python3  
**Difficulty**: Easy  
**Steps**:   
**Application**: Identifying Reachable MS/IIS Targets and assessing them for Telerik RCE  
**Competencies**:

---------------------------------------------------------------------------------

Contents

[Background 2](#_Toc46293792)

[Targeted Enumeration 4](#_Toc46293793)

[Execution: 5](#_Toc46293794)

[Vulnerability Identification Algorithm: 8](#_Toc46293795)

[Unit Testing 8](#_Toc46293796)

[Unit Test Summary 8](#_Toc46293797)

## Background

Telerik UI for ASP.NET AJAX contains a security vulnerability that if exploited exposes users to remote code execution. This vulnerability exists in Telerik's UI file handler RadAsyncUpload, which is the IOE (indicator of Exploitability) that the scanning tool is based upon.

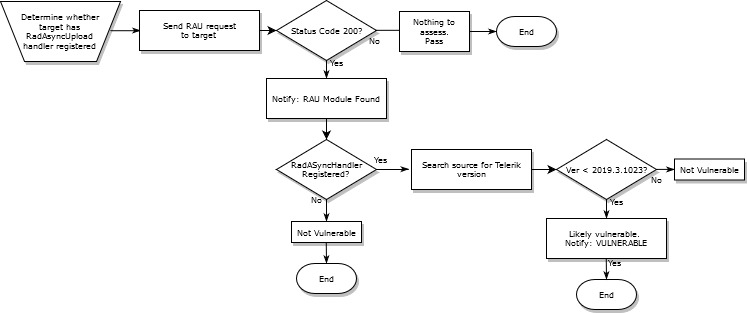
Telerik UI instances susceptible to exploitation must have the file handler registered, which can be confirmed by accessing the path to the RadAsyncUoload UI web resource:



The next step is to identify the Telerik ASP.NET AJAX version running on the target, as versions through 2019.3.1023 require a non-default setting to remediate the vulnerability. Simply put, finding an ASP.NET AJAX release prior to 2019.3.1023, indicates a high probability of exploitation.

It’s important to note that this search and evaluate technique is completely unauthenticated. If the target in question does require authentication to identify its version of Telerik, there’s a brute force method of guessing the correct version of Telerik in order to upload a file to a vulnerable instance. That technique is not covered in this demonstration.

## Vulnerability Identification Algorithm

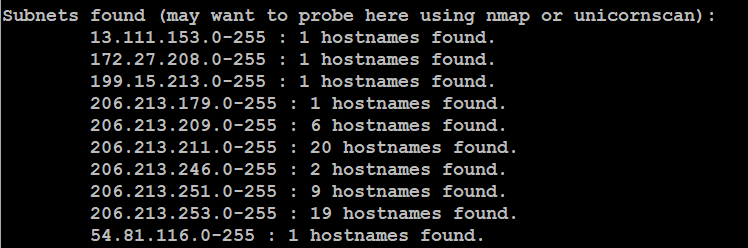


## Scoping for Ranges

To target relevant systems, some DNS recon is in order. A no-nonsense approach to generating a solid list of targets, IP’s, and interesting network ranges is to the DNS enumeration tool Fierce. Let’s find some interesting Aetna and CVSCaremark ranges to assess:

**Aetna**

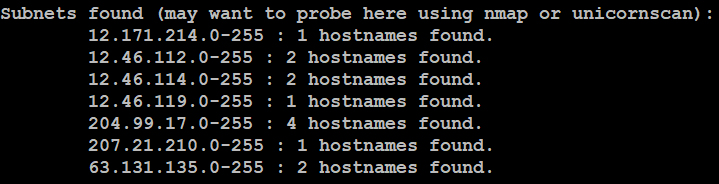
|  |
| --- |
| **$ fierce -dns aetna.com** |



**CVS**

|  |
| --- |
| **$ firce -dns cvscaremark.com** |

**CVS Results:**



This gives us something to work with. On the aetna side, we have the following network ranges:

13.111.153.0/24  
172.27.208.0/24  
199.15.213.0/24  
206.213.179.0/24  
206.213.209.0/24  
206.213.211.0/24  
206.213.246.0/24  
206.213.251.0/24  
206.213.253.0/24  
54.81.116.0/24

On the CVS side, we have the ranges below:

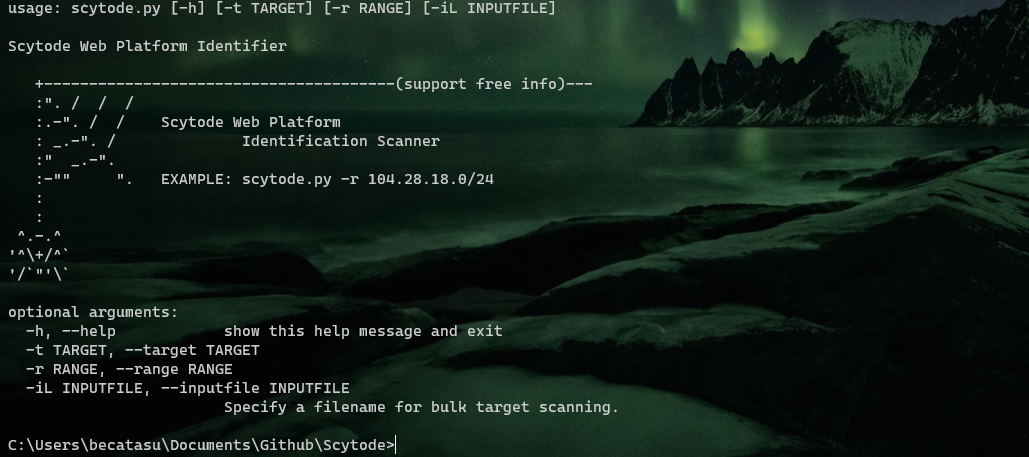
12.171.214.0/24  
12.46.112.0/24  
12.46.114.0/24  
12.46.119.0/24  
204.99.17.0/24  
207.21.210.0/24  
63.131.135.0/24

## 

## Targeted Enumeration

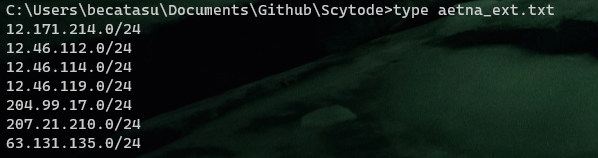
Because Telerik UI instances run on ASP.NET AJAX, web servers running Microsoft IIS are the targets of interest. There are a lot of ways to identify Microsoft IIS web servers, but, I of course like to roll my own.

A Scytodes Lugubris is a species of spitting spider found in Tropical Asia, Hawaii and Mexico. Recently I became aware that they’re now relatively common in Arizona as well, and I found the Genus Scytodes an appropriate name for my new scanner.



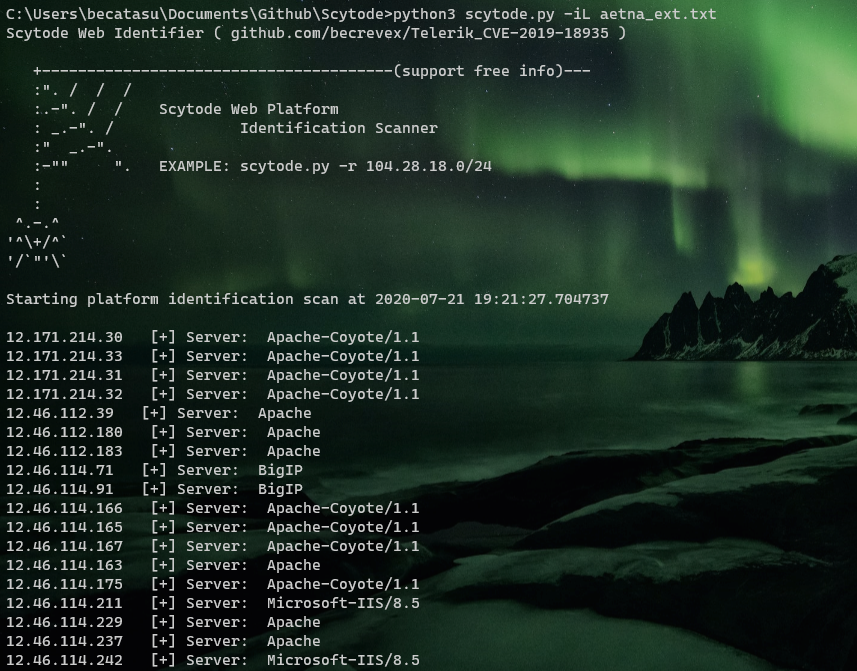
Scytode evaluates a single target, a CIDR notation range, or a list of hosts via an input file for their respective server types. To find a list of Microsoft IIS targets, we can supply it with a network range and let it do its thing. It’s a multi-threaded script, which speeds up the process from its original iterative approach, reducing the average time of completion to scan a C class subnet from 2 minutes to about 15 seconds.

After saving the identified Aetna network ranges to a file (aetna\_ext.txt), the file is used as an argument to scytode to scan, identify, and sort the findings by server type:

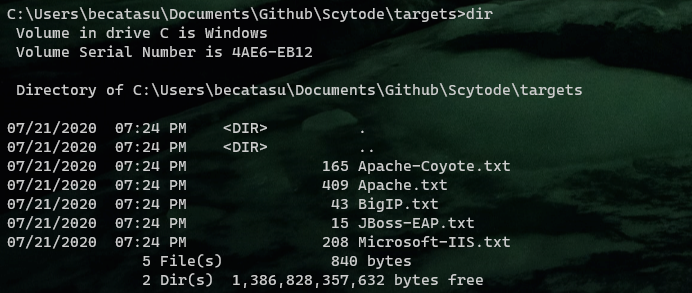


## Execution:

|  |
| --- |
| $ python3 scytode.py -iL aetna\_ext.txt |

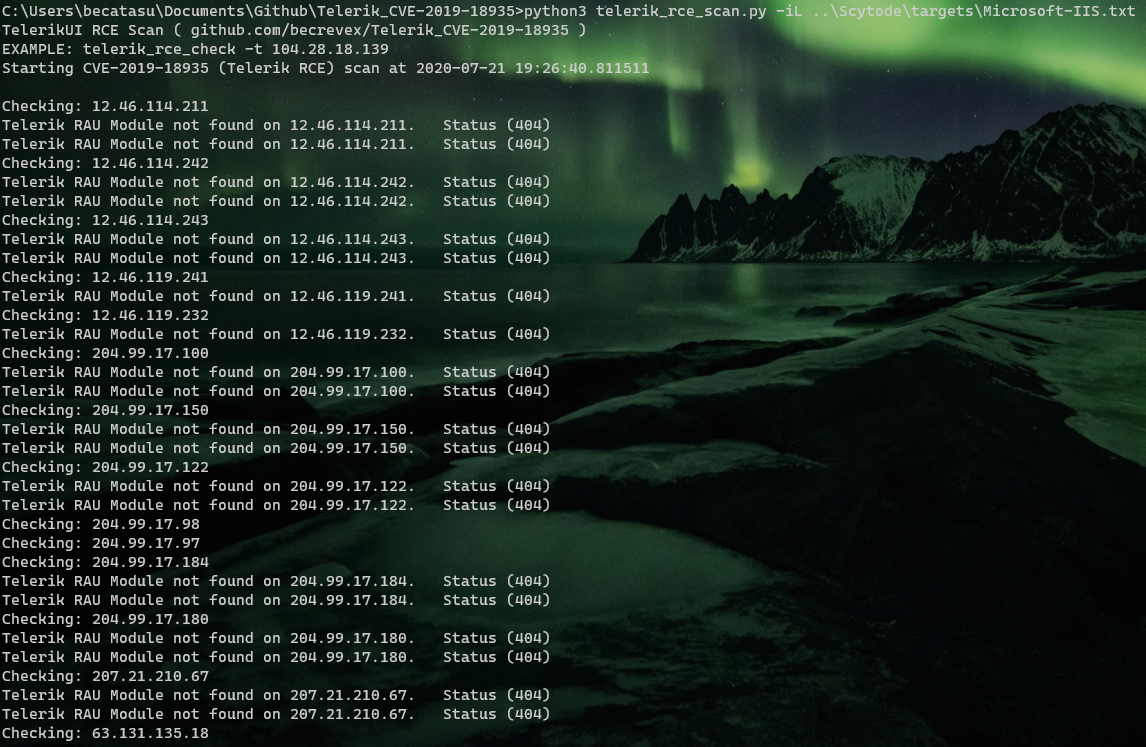


All discovered server types are grouped and saved to their respective files in the ./targets/ directory.



This allows us to target just the Microsoft-IIS servers for the Telerik RCE vulnerability.

|  |
| --- |
| $ python3 telerik\_rce\_scan.py -iL ..\Scytode\targets\Microsoft-IIS.txt |



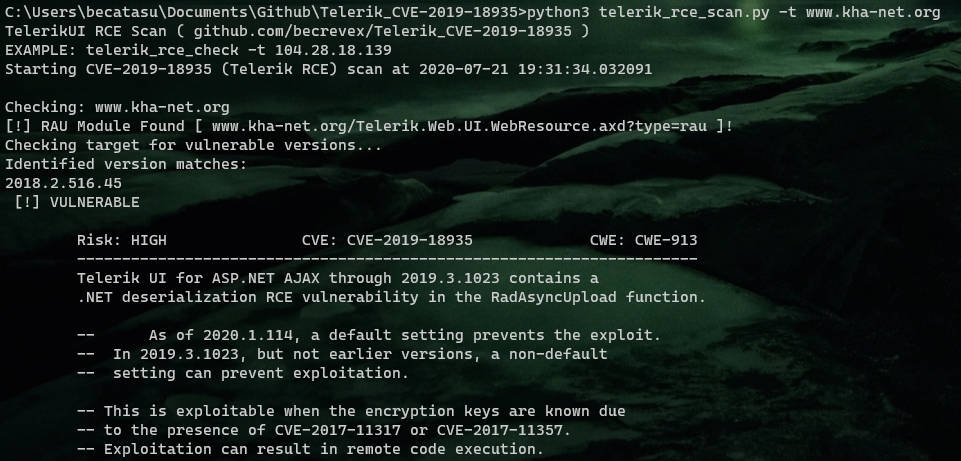
And it looks like the Aetna Network Ranges assessed are clean.

To show the identification of a vulnerable instance, some GoogleFu was used to ID URL’s in the wild that contain the term Telerik.

* www.kha-net.org
* www.cityofbeaufort.org/common/admin/Jobs2/
* www.cityofjackson.org
* www.townoffarragut.org [208.90.188.135] (208.90.188.0/24)

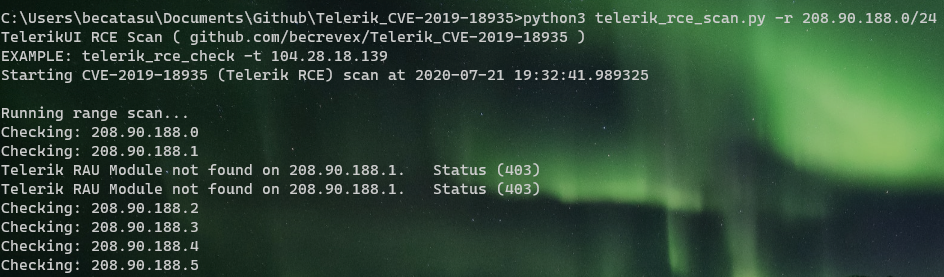
As mentioned above, the scanner can be used to target one instance:

|  |
| --- |
| $ python3 telerik\_rce\_scan.py -t www.kha-net.org |



CIDR notation ranges:

|  |
| --- |
| $ python3 telerik\_rce\_scan.py –r 208.90.188.0/24 |



## Unit Testing

Unit tests will address the core functionality of each function, ensuring the features of the tool perform as expected, generate accurate data points, and handle exceptions gracefully.

Use cases to be tested:

* Single Target Scan of likely vulnerable IP
* Single Target Scan of likely vulnerable Hostname
* Single Target Scan of unresolvable hostname
* Single Target Scan of invalid IP address
* Target Range Scan of IPs
* Target Range Scan of Hostnames via Input File
* Target Range Scan of IP’s via Input File

## Unit Test Summary

The testing performed assessed 11 individual controller modules, each containing various features and functions that stood to execute one primary function of the application. Of the 11 cases assessed, 4 did not pass the unit tests performed.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case / Module** | **Test Case ID** | **Pass/Fail** | **Summary of Defect** | **Comments** |
| Single Target Scan of LV IP  check\_vuln(target) | Case001 | Pass | NA | None |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case / Module** | **Test Case ID** | **Pass/Fail** | **Summary of Defect** | **Comments** |
| Single Target Scan of LV Hostname  check\_vuln(target) | Case002 | Pass | NA | None |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case / Module** | **Test Case ID** | **Pass/Fail** | **Summary of Defect** | **Comments** |
| Single Target Scan of Unresolvable Hostname  check\_vuln(target) | Case003 | Pass | NA | More specific exception handling can be applied to provide information in the event of NewConnectionError, Timeout, or SSL issue. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case / Module** | **Test Case ID** | **Pass/Fail** | **Summary of Defect** | **Comments** |
| Single Target Scan of invalid IP  check\_vuln(target) | Case004 | Pass | NA | Fails gracefully, but more context can be provided to illustrate why. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case / Module** | **Test Case ID** | **Pass/Fail** | **Summary of Defect** | **Comments** |
| Target Range Scan of IPs  check\_vuln(target) | Case005 | Pass | NA | None |

Implement error handling when providing wrong argument types –especially to ranges.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case / Module** | **Test Case ID** | **Pass/Fail** | **Summary of Defect** | **Comments** |
| Single Target Scan of Hostnames via File | Case006 | Pass  Fail |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case / Module** | **Test Case ID** | **Pass/Fail** | **Summary of Defect** | **Comments** |
| Single Target Scan of IP via File | Case007 | Pass |  |  |

----- Appendices -----