What to do

Tuesday, March 10, 2020

3:00 PM

**New**



**Jake Karnes**  [10:21 AM](https://netspi.slack.com/archives/C3YCY3AEL/p1670430067967159)

**For newer WaPen consultants:**

We've recently improved our onboarding documentation for the WaPen service line. While it's intended for new hires, this content covers topics that would be valuable for anyone who's recently started doing NetSPI WaPens. It dives into our processes and what's expected of you. Here are some of the highlights:

* [Testing environment setup](https://outline.netspi.com/doc/wapen-testing-environment-setup-and-common-resources-sM4mPP05go)
* [NetSPI's approach to WaPens](https://outline.netspi.com/doc/netspis-approach-to-wapens-Azv3OWJFPx)
* [WaPen project life cycle](https://outline.netspi.com/doc/wapen-project-management-and-project-life-cycle-nUTz4ga6U4)
* [How do to a kickoff call](https://outline.netspi.com/doc/kickoff-calls-Sj8Wcrbd8r) with [an example of the consultant's portion](https://outline.netspi.com/doc/kickoff-call-example-6cfwqk4V4B)
* [What to expect from a demo call](https://outline.netspi.com/doc/demo-call-osz1snqwYd) and [example questions to ask](https://outline.netspi.com/doc/demo-call-typical-questions-5hqCnkJq3m)
* [Resolve Overview for Consultants](https://outline.netspi.com/doc/resolve-overview-2Ni7PeYMzE) (this is existing content)
* [How do to a report readout](https://outline.netspi.com/doc/report-readout-VuUuGgnH76)
* [How to handle common scenarios and typical project problems](https://outline.netspi.com/doc/common-scenarios-Sk8T6MksEl)

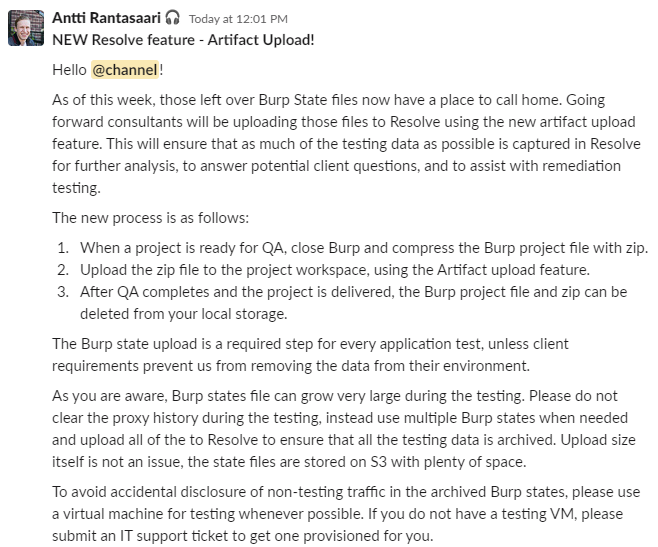
This is newly created content and all feedback is welcome. Please send notes and suggestions to myself and [@paulry](https://netspi.slack.com/team/U80EGKTK9). Let me know if you have any questions

:slightly_smiling_face:

 (edited)

From <[*https://app.slack.com/client/T0B7H2E59/C3YCY3AEL*](https://app.slack.com/client/T0B7H2E59/C3YCY3AEL)>

1. [Resolve Daily Communication (DO DAILY)](onenote:#Resolve%20Daily%20Communication%20(DO%20DAILY)&section-id={CC6FB3C2-CD8F-4D64-BCAE-3DC744447654}&page-id={965A2186-BE89-440D-98B0-0DC5A423166D}&end&base-path=https://netspi-my.sharepoint.com/personal/mallen_netspi_com/Documents/Marissa%20@%20Netspi/Resolve7%20WAPEN%20Checklist.one)

1. 
2. For findings Aaron said:

If it's static content and you can't take an action in the app, it's not an issue

If you can take action in the app then it is an issue

So yeah, if you can't exploit it or theorize an attack then it's not a finding.   
Example:

In regards to if a form does not contain an anti-CSRF token - You would not call out a lack of tokens if you cannot do anything with it nor can you theorize an attack because CORS is implemented correctly. If CORS didn't block you, maybe you might call it out. But like you'd probably just call out CSRF instead.

1. Verify your user accounts. If a user is read-only, it means that you should use it only to view data. Do not test parameters using this user session. Do not change settings in this user session.  In an actual app where we are given an account that is for read only, changes to the account could cause very real issues for the client.

Austin Altmann 
8:41 AM 
Hello, everyone! In a recent test, a client's dev environment was impacted for about an hour due to data modification. If you're 
not aware Of this, you have nothing to do with it. Here are a few reminders Of what we can do to prevent any issues, and some 
points where the tester did a very good job at making sure this was a much smaller issue than it could've been. This was a shared 
environment and not isolated for testing, so these suggestions would perhaps be more cautious than needed for most tests. 
Least privilege - is administrator access required? Discuss with the team if an account with lower authorization can be created 
for testing functionality and which functionality would specifically require administrator access. 
Always confirm with the team if data can be modified - reach out in writing to confirm if data within the application can be 
modified or if specific data cannot be. If it's discussed during a call that's not recorded, follow up in an email. Something like 
"Just wanted to confirm the points from our discussion today." A paper trail is very good to have when things go wrong. 
Make data identifiable - In this case, some disaster was avoided because the tester not only kept careful logs of what they 
were doing, but because they tested with an identifiable name and with identifiable data. Try to say "netspi" when possible so 
that data is attributed to us, and try and have "netspi" and/or your name somewhere in the account name. 
The tester deserves a kudos for their overall methodology and communication, although I wouldn't want to publicly name anyone. 
They did end up getting praised by the client during a damage control call! 

If you're doing a web app test on a prod patient data portal that has actual patient data in it, DO NOT UPLOAD THE BURP FILE TO RESOLVE! Reach out to someone like @Saurabh Jaolikar and they will help you with this. Make sure you redact the details of PHI from the report and after engagement and triaging you need to delete the burp file completely. The burp project file will be filled with tons of PHI throughout the test period.

There are scripts that can help you remove PHI from Resolve that Jake Reynolds has you can use before you upload any data. You do accidently notice there is some sensitive data. It can be run after uploading them to Resolve. We are not supposed to touch access or see any patient data, but if we do encounter it, then we can notify the client and redact it accordingly. Again, do not upload those burp scans into Resolve if they do have the PHI data. Then redact it in the report and report it to the client that they had this sensitive data open. We do not process any PII, NPI, ePHI.

From <[*https://app.slack.com/client/T0B7H2E59/C3YCY3AEL*](https://app.slack.com/client/T0B7H2E59/C3YCY3AEL)>



Saurabh Jaolikar Q:) 9:59 AM 
Also @here this is what we convey to all out client "NetSPl's 
penetration testing on occasion uncovers Pll, NPI em-ll. etc. stored inside Of client systems. part Of the 
testing engagement is attempting to access and extract this information. NetSPl 
then stores this information in internal systems for triaging and reporting 
purposes. NetSPl informs client about the discovered Pll if any and promptly 
deletes it. And only process the vulnerability related data." 
We should stick to this when delivering as well 
Please let me know if you need any further help 
also kindly keep me informed throughout the engagement and lets sync up before the report is delivered 
Let me know if there any questions 

12:58 PM 
Ethan Hobart 
tl;dr- We shouldn't be testing any web apps that have real patient data in them. Its one thing if we inadvertently uncover PHI that 
wasn't known to exist, but an entirely different animal if we're asked to test a prod web app that is known to have real PHI in it. if 
you're asked to do that ping @arantasa or @Saurabh 
CD g 

Antti Rantasaari 
1:20 PM 
Just to add to that 
If you will encounter protected health information (PHI) during a production test 
I. Stop any testing related to the PHI disclosure (e.g. on an Expen, stop testing that specific website) 
2. Contact the NetSPl privacy officer to determine the next steps 
3. Never try to validate if the data is real (in production, always assume it's real), and never search for PHI for specific people 
(like family members, coworkers, or celebrities) 

Parker Daudt 
Jan 29th at 1:01 PM 
Another question. Application deals with financial reporting. Unmasked SSNs 
are present. What's the best way to report it? Unmasked SSNs are rated Low, 
whereas General SSN disclosure is rated Medium. What's the best route? I 
don't want to ignore the context Of the application, but it is Pll 
2 replies 
N Added to your saved items 
Anti Rantasaari 4 months ago 
Many financial, healthcare, etc. apps require access to full SSNs but they 
should be masked by default; thus the low severity 

1. By Day 1 you should have:
   * Created a workspace and checklist,
   * Browsed the application and created a site map,
   * Reviewed user roles and access restrictions,
   * Mapped out critical functionality workflows,
   * Started testing the application.

1. In Resolve always make sure to FP stuff before deleting it, otherwise it could still be published by accident and then you have to go and delete it.

1. Machine generated alternative text:
   3.3.3 Common Tools Utilized 
   Note the below list is not a comprehensive list of tools that may be utilized during your test; however, it is 
   representative of the most common tools NetSPl utilizes when delivering Penetration Tests. 
   Burp Suite Professional 
   nmap 
   Nessus 
   Specialized tools, such as SQLMap, as needed 
   API testing tools such as SoapUl and Postman 

1. Make sure to use the Burp project options from the S drive for every single customer application.

Load the NetSPI standard Burp project configurations file. Burp tab -> Project options -> Load project options from S:\Assessment Team\Assessment Tools\Burp\burp\_netspistandard\_projectoptions.json  
**Turn off Use advanced scope control to avoid this advanced url scope:**

Machine generated alternative text:
Site map Scope 
O Target Scope 
Issue definitions 
O Define the in-scope targets foryourcurrentwork. This configuration•ffectsthe behavior oftools throughout the suite. The easiest way to configure scope isto browse to 
use 3dvsnced scope control 
Include in scope 
Enabled 
Paste URL 
Exclude from scope 
Enabled 
Add URL to include in scope 
Specify a regular expression to mstch each URL component. 
leave blank to match any item. An IP range can be specified 
instead Of a hostname. 
x 
Host IF range 
Host I IF range 
Host IP range: 
File: 
Enter regex, range or leave blank 
Enter regex or leave blank 
Enter regex or leave blank 

Prevent items from being logged in issues outside of scope:

Machine generated alternative text:
2 Proxy (811 Y8ffiO 
A L Idit - 
Edit live audit task 
3 of 382421 
Def. u t t c nfÉW8tz 
Def. u t t c nfÉW8tz 
O 
e tæk. 
o 
det8iB 
Task Type 
Tools Scope 
Proxy 
URL scope 
whi& 
o 
Y8ffic will to seled by 
tæk 
Audit 
Event log 
MAY 2020 
29 20 
28 2020 
by 
tæk. 

There is also a copy of this file in C:\Users\mallen\Desktop\Tools

Those are the settings for Burp we're supposed to use.  Biggest two things, it uses NetSPI's burp collaborator and when it tries to submit forms it submits like [security-audit@netspi.com](mailto:security-audit@netspi.com) or NetSPI for names and stuff. Burp collaborator links will also show up as: <https://nxocwggxxxp7h0f6x21fzu0y6pcf04.net-spi.com/receive_code/>

1. Crawl the site using Burp Suite and **make sure** that you have these extensions turned on before you crawl:

All projects:

* Logger++
* Error Message Checks
* J2EEScan (does not have to be used for every active scans for a project, repeats same checks multiple times)
* Retire.js
* Active Scan++ (Requires standalone jython JAR)
* Additional Scanner Checks
* HTML5 Auditor
* Software Vulnerability Scanner
* Backslash Powered Scanner (maybe, produces tons of false positives)

To be used when needed:

* Content-Type Converter
* SAML Raider
* JSWS Parser
* AWS Signer (NetSPI Github)
* WCF and AMF Deserialization Extensions (NetSPI Github)
* Wsdler

Personal Preference:

* .NET Beautifier
* JSON Beautifier

1. Don't forget to **crawl** the site **twice** using **two different files**.  Authenticated and unauthenticated so you can compare site maps for the check list. In the picture there is a spot to put in the credentials.

Machine generated alternative text:
Scan details 
Scan 
configuration 
Application 
login I 
Resource 
pool 
Application Login 
Specify the account credentials that should be submitted to any login functions. The crawler will use these to discover authenticated content behind login 
functions. 
Label 
Username 
New . 
Edit 
Delete 
2 

Make a project.  Crawl.  Close project.  Make a new project. Log in, give crawler creds.  Crawl. If you're already logged in and then click scan, it should use your session cookie(use if you need things other than just the credentials to login).

Compare two files. IF allowed you may AUDIT. If not allowed you may just passively crawl or possibly crawl without audit.

Open up a new burp project once the original crawl stops and then give the new project the credentials.

**While it's crawling, go around and shove your fuzzing string into everything you find.  
Crawling rules: Always allowed to passively crawl, active scanning or right clicking to crawl with or without audit are usually a no for Production environments. Can usually crawl with audit, sometimes can crawl without audit and do active scanning for non-production environments. Depends, if you see settings or email anywhere, don't crawl and audit.**



1. How to make Burp always get a non-cached response.

Machine generated alternative text:
Dashboard Target Proxy 
Intercept HITP history 
• Convert HITP 
Remove secure 
Intruder Repeater Sequencer 
ebSockets history Options 
s to HI-rp 
from cookies 
Decoder 
Comparer 
Extender 
Project options 
User options 
Logger*+ 
Comment 
JSON Beautifier 
para 
0 
Matc 
d Replace 
These set 
s are used to automatically replace parts of requests and responses passing through the Proxy. 
-nabled 
Edit 
Remove 
Up 
Item 
Request header 
Request header 
Request header 
uest header 
header 
Request 
Response header n 
Match 
Replace 
User-Agent: 
MozillaJ6_0 (iPhon__ 
User-Agent: MozillaJ6_0 (Linux__ 
Type 
Regex 
Regex 
Regex 
Regex 
Regex 
Regex 
Regex 
Emulate iOS 
Emulate Android 
Require non-cached response 
Require non-cached response 
Hide Referer header 
Require non-compressed respon_ 
Ignore cookies 

1. **While it's crawling, go manually map the site. Setup Autorize and use the other user's cookie to check for IDOR and MFLAC. And go around and shove your fuzzing string into everything you find.**

Map the whole site manually with the separate user accounts user1 and user2, search the results for anywhere that user1, user2, or 939 accountid show up. Check if you can you change those values and get into another account or company.

' // '" "" `` \*\* ${2+2} \\//\\// <script> << >> "> '> << )) (()) %

<InPUT type=image src=1 onerror=alert(1)><<SCr<script>script>alert(1)<><SCR<script>/script>

1. MFLAC/IDOR tester:

Use two browsers, for two different sessions, do autorize on both of them at once to save time by checking it against each other as you go through the site map browsing. Take a cookie from PenBuyer and one from PenSeller put these cookies in different burp suite sessions.  
I think the trick to the autorize extension is to use **two different browsers.**  When I was using an incognito window it seems like I was getting false positives.

Make sure that you have the burp cookie jar OFF for extensions or it will do false positives.

Sometimes you will have to unload the plugin and then reload it. Because it seems like after it gets a cookie, giving it a new one doesn't work.

Autorize, for each request you do it will send an equal request but with something changed and this something is usually the cookie of the session or any additional headers that you need to in order to connect to the webapp. And by doing this we can automate the process of looking for idors or access control issues. USER -A will browse the webapp and ask autorize to add the cookies of USER -B automatically. Copy ALL of the cookies. Give it lower privileged user's cookie and browse around as the higher privileged user.

Machine generated alternative text:
Response 
Params Headers He. 
def 
e.b-.ond.y. . lcBe r -f - 

Add any additional headers the request has that you might need.

Machine generated alternative text:
Aut 
Prevent 304 Not 
Ch«k 
here 

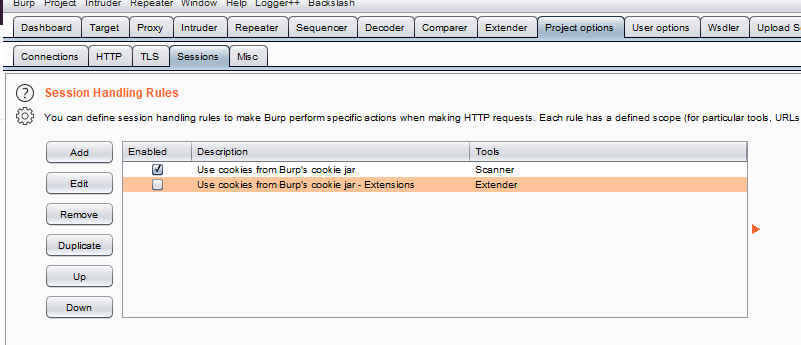
Paste the cookies here:

Machine generated alternative text:
Uodified Request _l_ Modified Rescx)nse _1_ Original Request Original Res 
Unauthenticated Request 
Unauthenticated Response 
'qnore 304/204 status code ressx»ns 
Authorization checks 
Auto Scroll 
Temporary headers 
C 00kie: 
hevent 304 Not Modified Status cod 
Check unauthenticated 
market ing_vistor_ id ff -4b67- 7b-680dff82da4d. 
_fbp-fb,1.1S60026941057.3211S6316: 
_g,d-CAlQ.9702644SMS600.'696S: 
web- mondavs sess - m zQ- loeNSC xaXOVvg.c ZICüvjRgRrA62308 
last request _ j 
Detector UnauttRnticated 
InterceMion Filters Taue Filte 

Click to add blank filter. Adds scope items only.

Machine generated alternative text:
Fitte LEt: 
E —u red) 
URL Not 
Add fitte 
fitte 

Turn off extensions in session rules. Turn it back on afterwards.



Then turn autorize on.

Machine generated alternative text:
Req t/ 
304/204 s t8tL_E æde 
o 
Præt 304 Not rubdfied s t8tLE æde 
from 
._lt'mtiæted 

1. **[SKIP, Autorize unless the app is removed]** To easily test for MFLAC unauthenticated (sometimes it doesn't work if the website redirects even when you're authenticated):   
   In Chrome, manually map the site. Go to any urls discovered with GoBuster and Dirbuster.

To get all of the urls from the manual Chrome site-map browsing, close Chrome and run these commands in PowerShell:

cd "C:\Users\mallen\AppData\Local\Google\Chrome\User Data\Default"

sqlite3 History "select url from urls order by last\_visit\_time desc" > history\_export.txt

The text file will be found in C:\Users\mallen\AppData\Local\Google\Chrome\User Data\Default

Then copy and paste all the urls from the text file into the Copy All Urls(Umbrella) extension in Chrome. It will automatically load all of the urls for you.

Open your unauthenticated burp session and copy and paste all the urls from the text file into the Copy All Urls(Umbrella) extension in Chrome. It will automatically load all of the urls for you. Check all urls found in Chrome and check for MFLAC as an anonymous user. Compare the authenticated and unauthenticated site maps to check for any differences.

**Another way to test for unauthenticated MFLAC** is by using this method to get all of the unique urls and then using the umbrella extension:

[Test for POST and PUT](onenote:#Test%20for%20POST%20and%20PUT&section-id={CC6FB3C2-CD8F-4D64-BCAE-3DC744447654}&page-id={A2A0E30B-4DB1-4C5C-BD49-FD4324BEF650}&end&base-path=https://netspi-my.sharepoint.com/personal/mallen_netspi_com/Documents/Marissa%20@%20Netspi/Resolve7%20WAPEN%20Checklist.one)

1. If a website uses HTTPS then mark **TCP** and ports as 443, if it runs over HTTP mark as TCP and ports as 80 in Resolve. When doing webapps everything is going to be TCP. Probably for web apps it's going to be TCP and the port 99.9% of the time unless there is a really weird api.  The HTTP thing is probably used on thick apps or mobile apps more.

1. For Nessus we don't include lows on other ports. So if the app is an https on port 443, only verify the lows that are on port 443. If you see a low that's on port 1121 then mark it as a false positive. For medium findings we do the same thing and verify the medium findings if they're on the same port. If they're on a different port, that's harder. It needs to be meaningful. SSL ciphers on a random port no. a MGMT interface yes. Unless they gave us access to a non-prod env, then no. Med Injections or Excessive Privileges - Yes, Externally available MGMT interface - yes (it's reportable because it's not a best practice, in an ideal world you don't expose SSH to the world.  That's what a VPN is for). SSL issues - no.

Good way to verify if it's important or not: there's like 17 thousand instances of it in resolve, so obviously we report it a lot. If there were 3,i probably wouldn't.

1. Fuzz everything with Burp fuzzing lists or payload all the things lists. Double check to make sure there are no exploitable injections by using Intruder with the linked basic payloads on all of the parameters found by paramalyzer. For open-redirect uncheck url encode, mark follow redirect as always and check process cookies in redirections.

For non-prod:  
[C:\Users\mallen\Desktop\Tools\CustomAllInjectionPayload.txt](file:///C:/Users/mallen/Desktop/Tools/CustomAllInjectionPayload.txt)  
For prod:  
[C:\Users\mallen\Desktop\Tools\productionCustomAllInjectionPayload.txt](file:///C:/Users/mallen/Desktop/Tools/productionCustomAllInjectionPayload.txt)

<https://github.com/1N3/IntruderPayloads/tree/master/FuzzLists>

Open-redirect check:

<https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Open%20Redirect#common-injection-parameters>  
  
Also make sure to then go back and check for XSS and DOM XSS in any of the url parameters that have = and test them to see if they launch any scripts.

If you see any JSON or XML. Send it to repeater, and then try to use content type converter to change it and test for sqli or xml injections.



1. Base64 decode cookies and all things/headers.

1. When uploading things into Resolve. Upload the full findings file first, then check what you don't need from it/verify findings via the Dashboard. Then go and select those specific finding issues in the dashboard and upload those. Delete the original burp findings file that contained all of the issues so the resolve workspace doesn't look like a mess.

1. Run SSL scans using [SSLSCAN + Nmap](onenote:#SSLSCAN%20+%20Nmap&section-id={CC6FB3C2-CD8F-4D64-BCAE-3DC744447654}&page-id={7648E34A-235B-4EFC-A661-FEA2D7345C10}&object-id={2311EF36-8509-43A0-BB6A-55499ED57EC7}&10&base-path=https://netspi-my.sharepoint.com/personal/mallen_netspi_com/Documents/Marissa%20@%20Netspi/Resolve7%20WAPEN%20Checklist.one) if there's an open port go inspect it using the port number, example port 1120. See what port it is, could be apache tomcat admin page or something. This is how you find the public admin page if there's no directory enumeration. <https://www.test.com:1120>

1. Search for [amazonaws.com](http://amazonaws.com/) in the burp suite filter. See if it finds anything. Then do [AWS - S3 Directory Listing](onenote:Checklist%20Comment%20Notes.one#AWS%20-%20S3%20Directory%20Listing&section-id={AAEA25AC-58AA-4F5A-9FB4-CC302A3872B5}&page-id={10960ED1-F7D3-424E-B5BD-6810640F55E4}&end&base-path=https://netspi-my.sharepoint.com/personal/mallen_netspi_com/Documents/Marissa%20@%20Netspi)

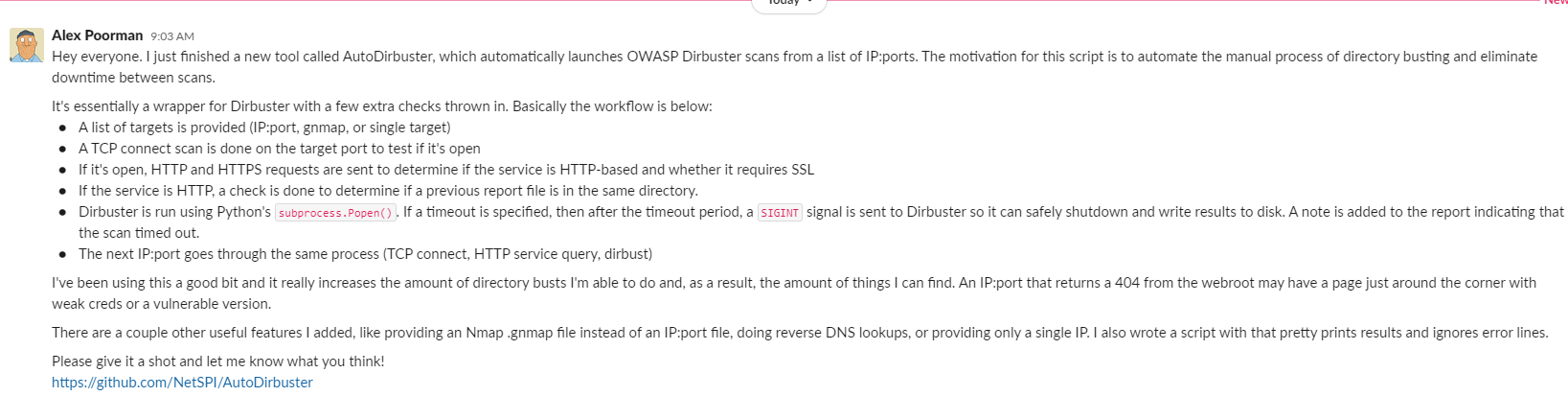
Search for:

* AKIA
* ASIA
* ey - the beginning of a JWT token. Might be being used as cookies.

AKIA and ASIA are the beginning strings of AWS access keys. AKIA means programmatic access ASIA means assumed role

Of course you still need the secret key id but those are still a big deal to find by themselves.

1. Use [GoBuster](onenote:#GoBuster%20using%20SecLists%20to%20discover%20directories&section-id={CC6FB3C2-CD8F-4D64-BCAE-3DC744447654}&page-id={1535452C-9199-4F42-9CFA-4F1992827396}&end&base-path=https://netspi-my.sharepoint.com/personal/mallen_netspi_com/Documents/Marissa%20@%20Netspi/Resolve7%20WAPEN%20Checklist.one), Dirsearch, [Dirbuster](onenote:#If%20allowed%20to%20do%20dirbuster-type%20stuff&section-id={CC6FB3C2-CD8F-4D64-BCAE-3DC744447654}&page-id={889388DE-8D59-4598-A9EA-56E84807775D}&end&base-path=https://netspi-my.sharepoint.com/personal/mallen_netspi_com/Documents/Marissa%20@%20Netspi/Resolve7%20WAPEN%20Checklist.one) to fuzz for directories. Subdomain enumeration, example if there's [www.you.integration.com](http://www.you.integration.com) and [www.auth.integration.com](http://www.auth.integration.com), or any link like [www.subdomain.integration.com](http://www.subdomain.integration.com) then make sure to also enumerate the subdomain with Dirbuster and you might find [www.admin.integration.com](http://www.admin.integration.com). Brute force subdomains for usbank.com with amass in Docker:   
   ~/tools/amass/amass enum -brute –d usbank.com  
     
   In tools folder:

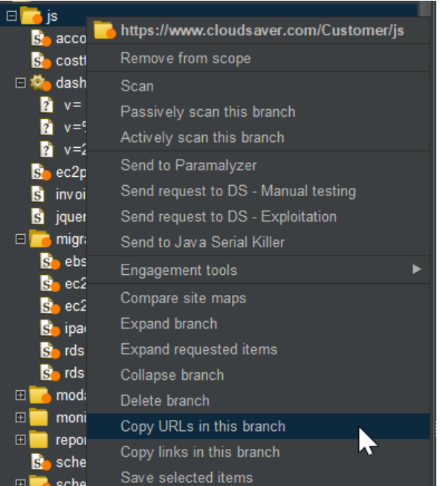


1. Search in GitHub using GitHub’s search feature:  
   “@usbank.com” for GitHub  
   <https://github.com/NetSPI/OSINT/tree/master/GitHub> Or get access to the github search tool by asking Thomas

1. Check the BurpJSLinkFinder Extension, look for interesting links, see if there is anything you can exploit (to carry out exploit see [JSLinkFinder Demo & Response modification](onenote:#JSLinkFinder%20Demo%20%20Response%20modification&section-id={CC6FB3C2-CD8F-4D64-BCAE-3DC744447654}&page-id={DD6DA265-4450-443C-BA10-0174288E19ED}&end&base-path=https://netspi-my.sharepoint.com/personal/mallen_netspi_com/Documents/Marissa%20@%20Netspi/Resolve7%20WAPEN%20Checklist.one)). Check for common or default directories for the technology being used and append them on to stuff. You do this by using the BurpJSLinkFinder and going through and visiting all those URLs. Export the log because the log can accidently be written over in burp for some reason. These will also often show up by doing gobuster.

Download all the javascript so now you can search through it.

Right click and get all the URLs.  Then put them into a file called like javascript.txt



And then open git bash anywhere, desktop, folder.  And run the command:

wget -I javascript.txt

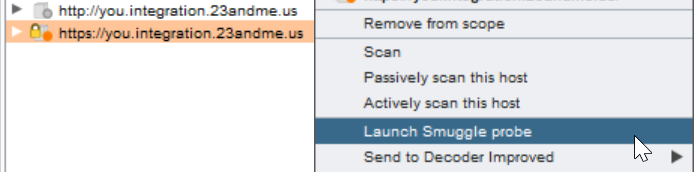
And it will download all the files in the URLs to a text file in that location.

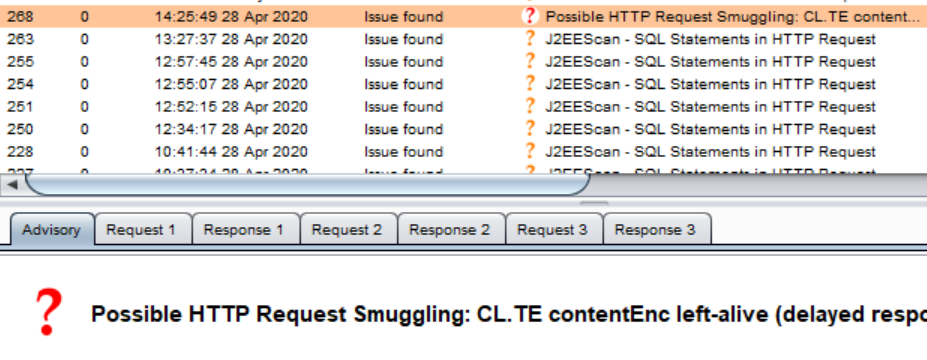
And then you can search through all the javascript at once by running the command:

grep -E 'SeachStringHere' \*

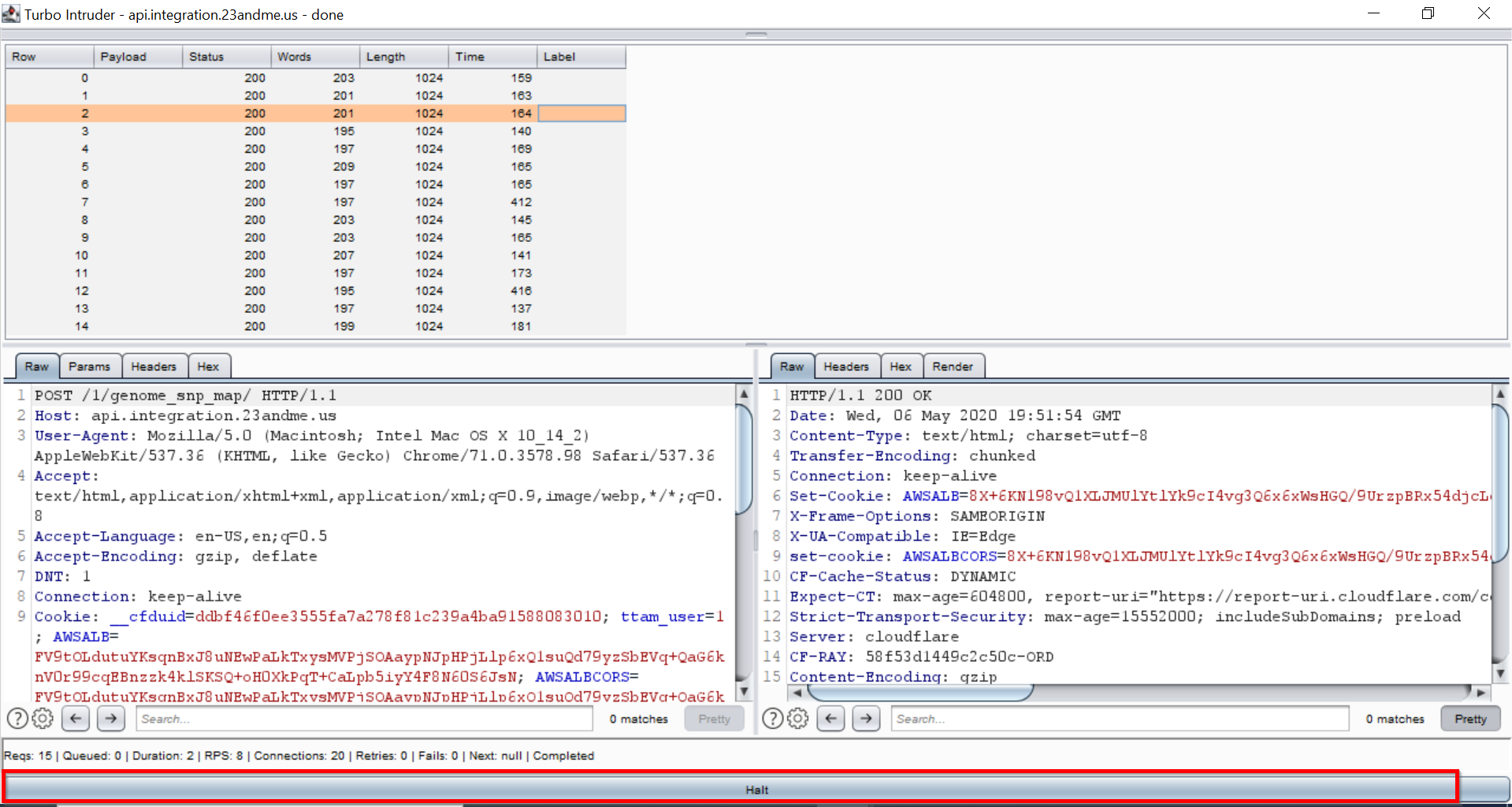
Can search for href, s3bucket [amazonaws.com](http://amazonaws.com/), or whatever you want in the 'SearchStringHere' section.

1. **Check for HTTP Request Smuggling.** Right click on the site map folder and click launch smuggle probe. Check the Dashboard and check issues for a possible http request smuggling. If there is one, right click on a request and smuggle the attack for its matching advisory CL.TE or TE.CL. Click attack and halt, repeat this process a few times. Looking for any 400 errors to indicate request smuggling. Otherwise it says Error: Referer checking failed - no Referer.





Machine generated alternative text:
Advisory Requ%t 1 Response 1 
Raw Params Hesders Hex 
1 POST / l/genome snp_map/ HT 
GZIP encode body 
LAunch Smuggle probe 
Smuggle sttsc:k (CL. TE) 
Smuggle attack (TE_CL) 
Send to Decoder Improved 



1. **Check for Content Security Policy (CSP**) by using the CSP-Bypass Burp Extension. It will scan for it, then just upload burp findings to Resolve. Verify the real from the false positives.

In the work space click the bug to add a manual instance. Click to change the master finding. Need to get rid of the blue filter by clicking x. And search content-security and then you get Misconfigured content-security-policy. That is the one that we want to select to put all CSP Issues under unless it's missing the CSP header itself.

Three kinds of CSP things we check for:

1. Any kind of XSS sources that are allowed like 'unsafe-inline' 'unsafe-eval'
2. Consider the following misconfiguration scenarios:
   * It can be an overly permissive policy if default-src is not set or set to wildcard and/or other directives are set to wildcard.
   * Multiple instances of this header are allowed in same response. A development team and security team might both set the header but one may use one of the deprecated names. While deprecated headers are honored if header with latest name Content Security Policy is not present, they are ignored if policy with content-security-header name is present. Older versions only understand deprecated names. Hence, in order to achieve desired support it is essential that the response include an identical policy with all three names.
   * If a directive is repeated within the same instance of the header, all subsequent occurrences are ignored.
   * If either the unsafe-inline and/or unsafe-eval configurations are used, a site's client-side security could be damaged as they allow scripts from event attributes and inline scripts to execute.
3. If the CSP header is missing in the application. (Need to confirm if we check for this, but we would need to put this under a different finding)

<https://www.netsparker.com/blog/web-security/negative-impact-incorrect-csp-implementations/>

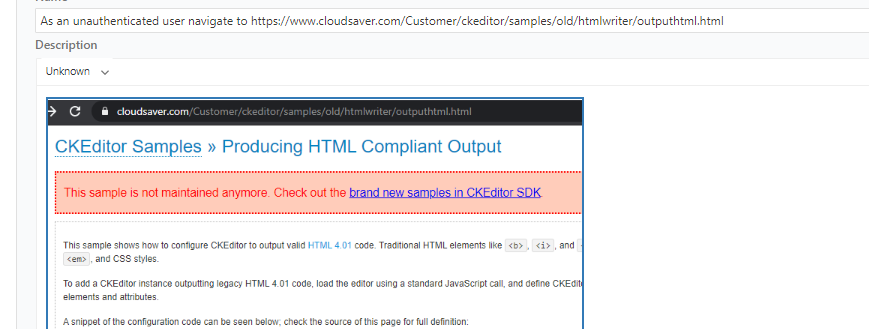
<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy>

1. Nessus scan, Shodan, Nmap. Test the given main in-scope urls and check if a site can be loaded on a different **port.** Ex: scan shows that port 8443 is open, then check it like: <https://you.integration.23andme.us:8443/>  
   Just checking to see if it's different from the normal site. If it is different, it's not a specific finding. It's just dependent on if you find anything at all on it that there is a finding now on that port for the site.

If the port is a really important open port like tomcat, then try this one if the first method doesn't work:

[Start up SSH Server for SSH Tunnel](onenote:Expen%20Labs.one#Start%20up%20SSH%20Server%20for%20SSH%20Tunnel&section-id={02289F0E-155A-492D-9EED-F9FC39B865CB}&page-id={0C93ADEF-5AEE-4EBE-B361-C210B19E875E}&end&base-path=https://netspi-my.sharepoint.com/personal/mallen_netspi_com/Documents/Marissa%20@%20Netspi)

1. Use the **Wappalyzer** and **Netcraft** extensions to detect technology used on the website. Research if the technology versions used have any known vulnerabilities, call out those vulnerabilities in the report. In the console try and find commands like $().jquery to



Machine generated alternative text:
Analytics 
Ana I ytics 
O New Relic 
JavaScript frameworks 
t 3.3 
Tag managers 
Tag 
JavaScript libraries 
L Odash 
jauery 33Ä 

1. Use waybackurls to find old urls from the Way Back Machine. To use make sure Docker is running, in powershell run the command: **echo "**<http://www.example.com>**" | docker run -i cylab/waybackurls**

1. If you need to test forgot password or checkout anything then use: [Netspi Email + Credit card](onenote:#Netspi%20Email%20+%20Credit%20card&section-id={CC6FB3C2-CD8F-4D64-BCAE-3DC744447654}&page-id={1E6BDC93-DA71-4EF7-A2A2-BBBA465956E3}&end&base-path=https://netspi-my.sharepoint.com/personal/mallen_netspi_com/Documents/Marissa%20@%20Netspi/Resolve7%20WAPEN%20Checklist.one)

1. Check Docker if there's time <https://hub.docker.com/search?q=usbank&type=image>

1. Check for any additional stuff here  
   <https://medium.com/@vignesh4303/collection-of-bug-bounty-tip-will-be-updated-daily-605911cfa248>

1. Looks like Nessus checks for PUT for us. So after we have Nessus we won't need the curl method unless you want to be really thorough. One of the informational findings it had was about methods, check those when you import the Nessus file. If webdav isn't running, I don't think I've ever seen PUT.

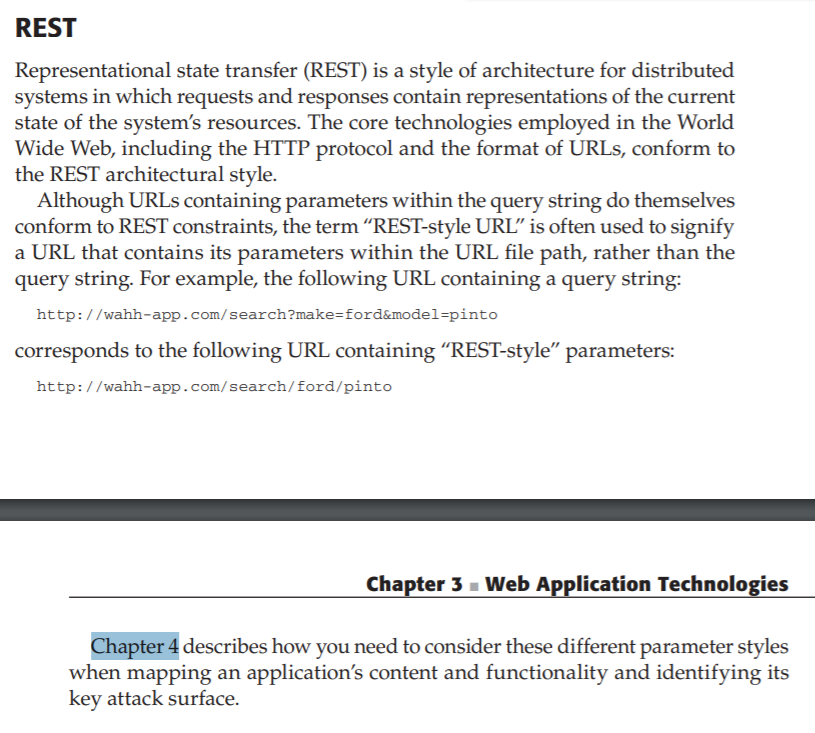
[Test for POST and PUT](onenote:#Test%20for%20POST%20and%20PUT&section-id={CC6FB3C2-CD8F-4D64-BCAE-3DC744447654}&page-id={A2A0E30B-4DB1-4C5C-BD49-FD4324BEF650}&end&base-path=https://netspi-my.sharepoint.com/personal/mallen_netspi_com/Documents/Marissa%20@%20Netspi/Resolve7%20WAPEN%20Checklist.one)

1. <https://cheatsheetseries.owasp.org/>

1. If open redirect: Use the open\_redirect\_wordlist.txt and the payload all the things master open word list.
2. <https://resources.infosecinstitute.com/web-application-pentest-guide-part/#gref>
3. <https://resources.infosecinstitute.com/web-application-pentest-guide-part-ii/#gref>
4. USE THIS ONE FOR **ALL INJECTION TESTING**: <https://portswigger.net/support/burp-testing-methodologies>
5. Use Burp to test for access control issues: <https://portswigger.net/support/using-burp-suites-site-map-to-test-for-access-control-issues>
6. Burp test for missing function level access controls: <https://portswigger.net/support/using-burp-to-test-for-missing-function-level-access-control>
7. If you're having trouble with figuring out what's getting filtered when testing an app.

Machine generated alternative text:
Dan Norte 
10:35 AM 
1.) Throw unexpected input into everything. 
2.) Check for errors. 
3.) Evaluate how your unexpected input was processed/changed. 
4.) Adjust as necessary. 
5.) Pop sells, get paid. (edited) 
If you've got git bash, you could always use diff to really see what's being filtered. It would take an extra few 
seconds, but at least it would be a nice visual. 
Save that long string into payload.txt 
Then save just what gets returned in response.txt 
diff payload.txt response.txt (edited) 
But just copy and paste and save the part between the <span> tag. Not the whole response. 

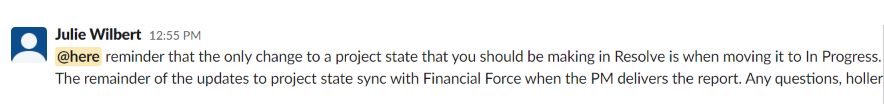
1. Machine generated alternative text:
   payload.txt 
   response.txt 
   03/25/2020 10:42 
   03/25/2020 10:42 
   Text Document 
   Text Document 
   norte@NETSPI-494-DNO MING"'64 4 Documents/ netspi_labs/altoromutual 
   diff payload. txt response. txt 
   " S{2+2} (O) % 
   norte@NETSPI-494-DNO MINGW64 4 Documents/ netspi_labs/altoromutual 



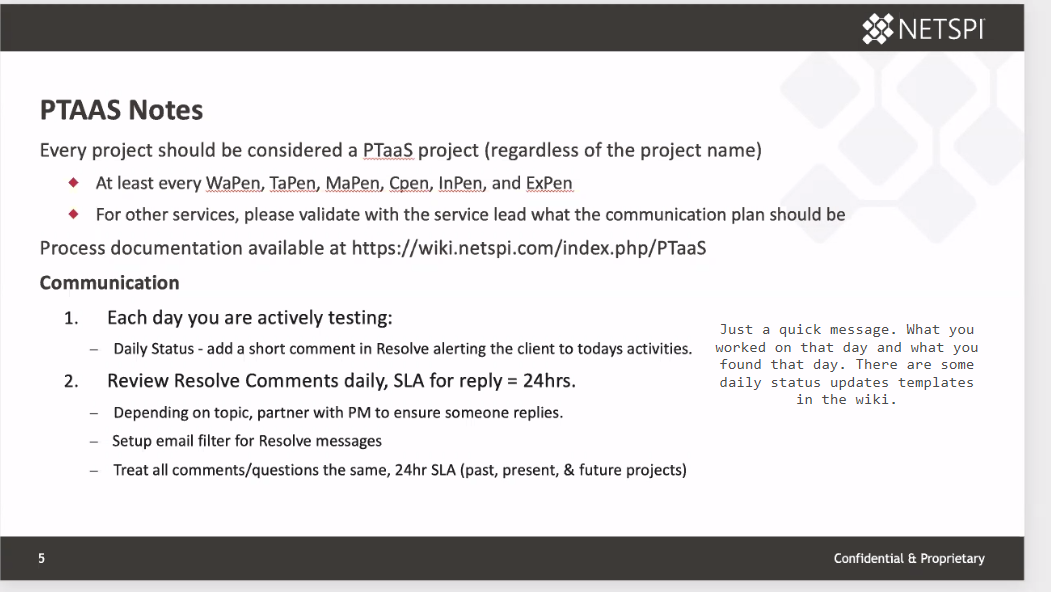
Resolve Daily Communication (DO DAILY)/ Additional Reporting Standards

Thursday, March 11, 2021

11:06 AM

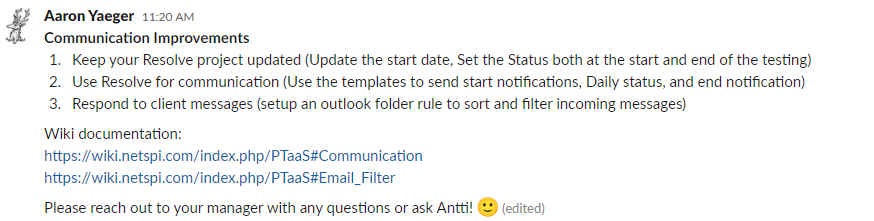


Generating Reports 
(edit I source) 
• Always generate Microsoft Word report• 
• The reporting process may change as more clients move to PTaaS, but as Of this writing (03/05/2021), continue generating MS Word 
• Always the CSV,'XLSX 
• Filenarnes should be as follows: 
• Example: A report for a client named XYZ Capital for a WA Pen done in the year 2021 on an application named Test Application should have a filename Of: XYZCapitaI_2021 
• Check [Page#CIientsIthis client Wiki page for more hformation on client requ*ements 



<https://wiki.netspi.com/index.php/PTaaS#Communication>

<https://wiki.netspi.com/index.php/PTaaS#Email_Filter>



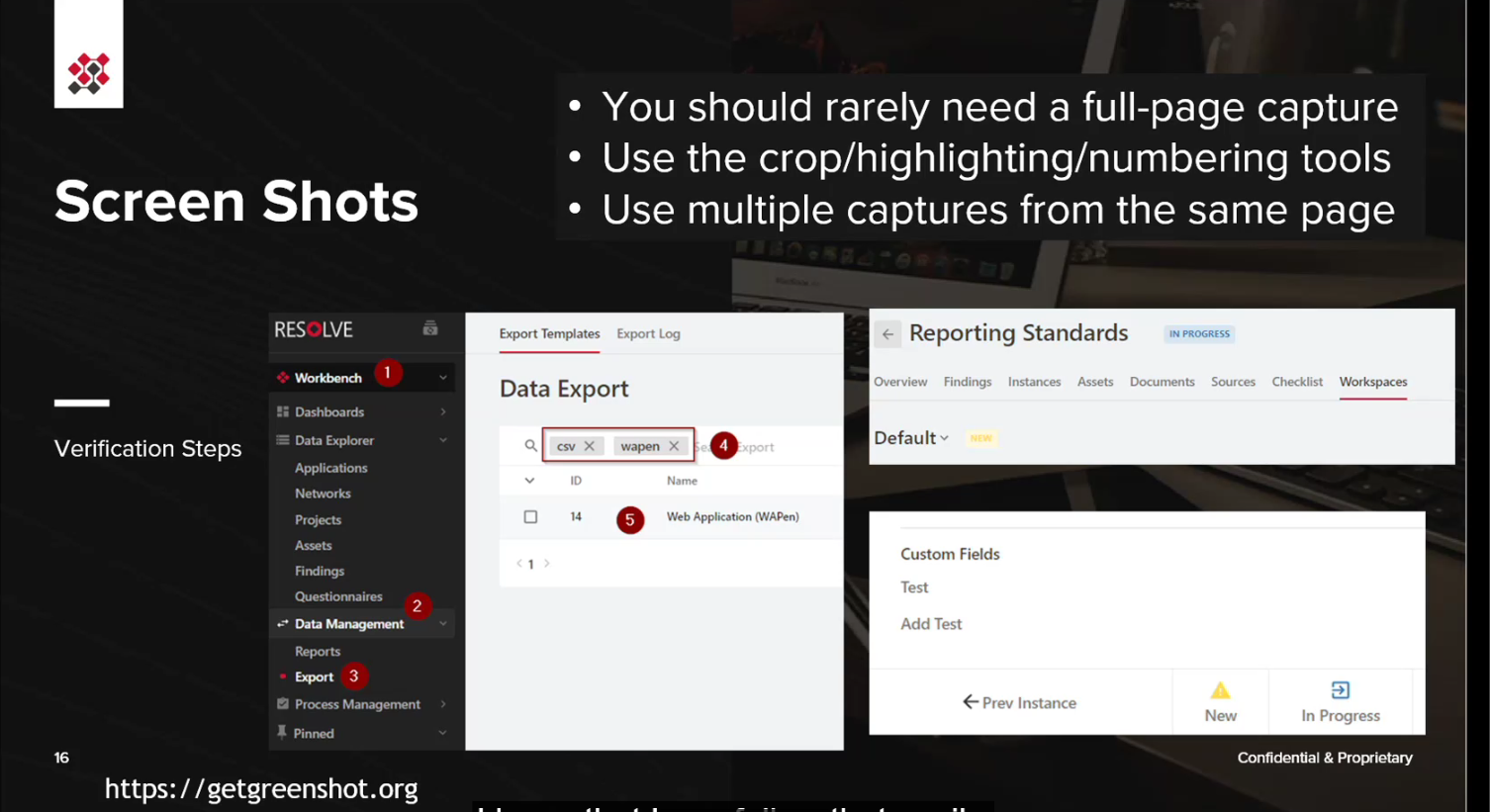
Communication 
be to R.I. B diff—t during kk:koff 
• Test starting the be#ning of the project (Sam* Start template) 
• Daily Daily stab." - Wepn Daily - 
• Test finished "—sage at the end the testing complete template) 
details t%bng to ttw 
Project Start 
When starling your mako to do 
• Set the Start and end dates 
Set In 

<<NetSPI Services Documentation\_Other\_Lunch and Learns\_Reporting\_Standards\_LunchAndLearn.pdf>>

* They want all of the Burp Suite findings to be imported into the workspace now and just FPed.
* They want your screenshots in a perfect square now
* Utilize the Greenshot numbering tool
* No bookmarks or extensions in the screenshot (just take a Greenshot color block and remove them, or do the puzzle piece method in Chrome to hide them)
* Make multiple screenshots of the same page and add them to a verification step if the screenshot is too big. I guess because Resolve makes it really small in the report if you take a large screenshot

From <[*https://app.slack.com/client/T0B7H2E59/GTKUYM0HE*](https://app.slack.com/client/T0B7H2E59/GTKUYM0HE)>

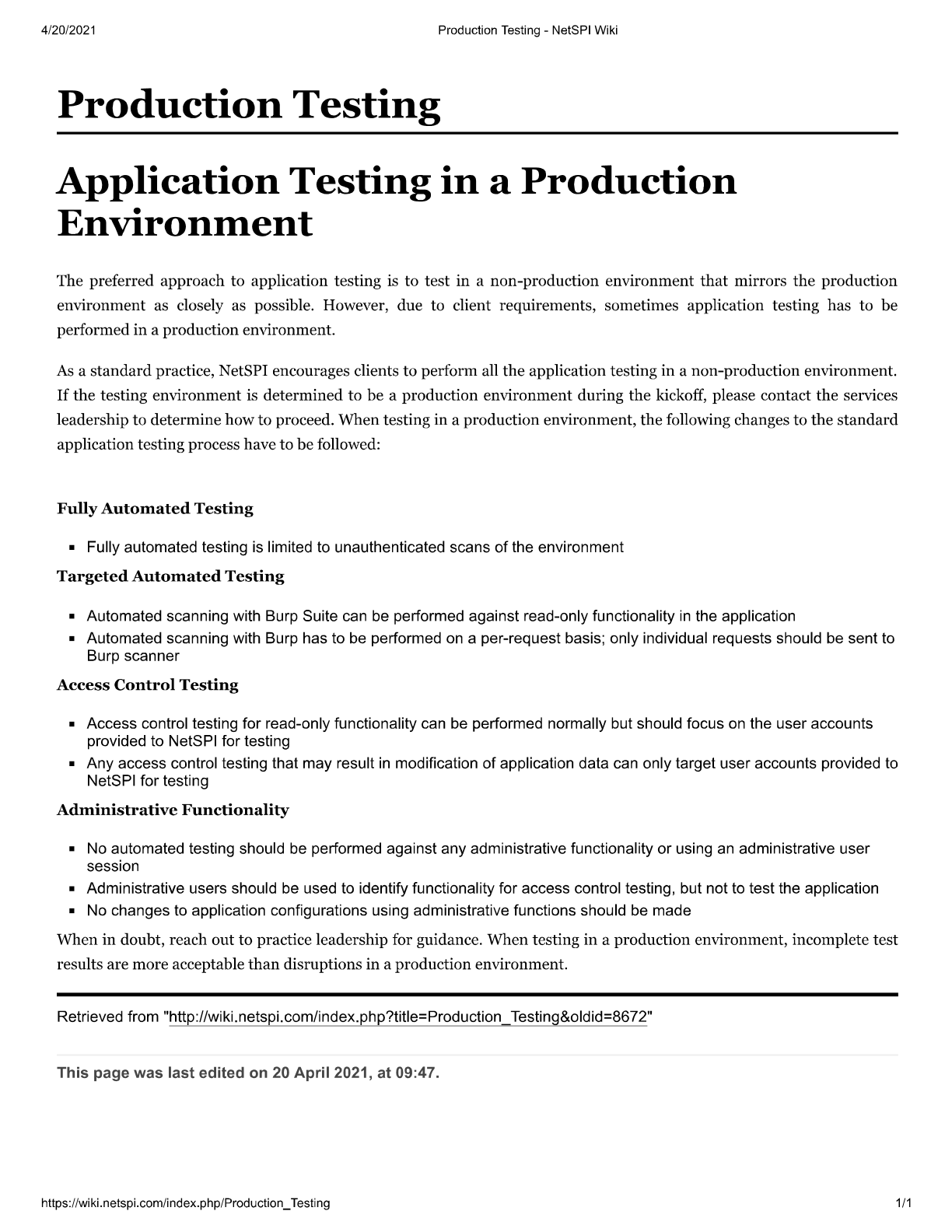
Screen Shots 
Verification Steps 
• 
• 
• 
Ensure that text is legible 
Must fit nicely in the report 
• Square rather than rectangle 
No unnecessary extensions 
Goo le 
Cmfidenual & Procyietary 



How to test in a production environment

Tuesday, April 20, 2021

1:58 PM



(READ b4 making report) Report attack params

Wednesday, April 15, 2020

10:02 AM

Standard WAPen reporting guidelines

Verification notes

Instance description for High and Medium severity findings on the Instance details tab

Multiple Verification steps vs all in one.

Headings are descriptive but concise

Text for HTTP Original request/ HTTP Modified request / HTTP response verifications steps

Site wide issues - fp bulk, finalize one, update url to base landing page, update the description to "This issue is site wide.", add verification items.

**Reporting and Testing Process Notes** Screenshots - limit the size of screenshots to make them more readable

* Add highlights / arrows / etc. to screenshots to point out the functionality in question as needed

Burp requests / responses - these are only needed in the verifications steps when they add value; no need to include every request / response pair

* But when in doubt, too much is better than too little

Use inline screenshots in verifications, not attachments

Ensure that every verification step has a clear description in the "Name" field

* Just saying "HTTP Request and Response" is not descriptive enough

Ensure that affected assets have affected ports listed, if applicable

Affected URLs

* Single URL per affected instance - avoid duplicate URLs/instances when just parameter values are different

Checklists

* Mark items completed as the testing progresses
* Many checklist items can be found on most pages and have to be tested on most pages; these should be marked complete after the whole test has been completed
* Add findings manually via the checklist to the workspace ("Add Manual Finding" on checklist items); this way everyone uses the same findings for most common vulnerabilities

Machine generated alternative text:
v Remote Management Interface - SSH 
v El 3.92.42.157 - Resolve 7 
v An SSH server is listening on this port. 
An SSH server is listening on this port. 
An SSH server is running on the remote host. 
> Weak Session Management - Insufficient Session Expiration 
Low 
Information 
Ignore 
Instance Details Verification Items Similar Instances 
Verification items (optional) 1 Total 
Preview or add verifications for this instance 
Findings Details 
Comments 
Copy verifications to... 
Include in Report 
U 
Name 
NESSUS Output 
Imported on May 19, 2020 
NESSUS Output 
Description 
9 
SSH version : SSH-2.O-OpenSSH_7.4 
SSH supported authentication : publickey,keyboard-interactive 
SSH banner : 
22 
- WARNING This system is for the use of authorized users only. Individuals 
using this computer system without authority or in excess of their authority 
are subject to having all their activities on this system monitored and 
recorded by system personnel. Anyone using this system expressly consents to 
such monitoring and is advised that if such monitoring reveals possible 
evidence of criminal activity system personal may provide the evidence of such 
monitoring to law enforcement officials. 
Attachment 

You find the functionality (form) on <https://www.cloudsaver.com/Customer/reports/monthlyinvoicevariance.aspx>, but the name parameter is submitted to /Customer/ajax/reportlayout/save.ashx

[8:59](https://netspi.slack.com/archives/CTJENQ5PY/p1586959196070000)

Call out the /Customer/ajax/reportlayout/save.ashx url and the name parameter in resolve, not monthlyinvoicevariance

So report where the vulnerability is not where the payload shows up in the URL field.

**Database/verbose error** messages will have attack parameters if you modified a value!

We put parameters, cookies, or anything in "attack parameters" and not in "query parameters" or "cookies".  Otherwise it doesn't show in the report.

If a website uses HTTPS then mark TCP and ports as 443, if it runs over HTTP mark as TCP and ports as 80 in Resolve. When doing webapps everything is going to be TCP. Probably for web apps it's going to be TCP and the port 99.9% of the time unless there is a really weird api.  The HTTP thing is probably used on thick apps or mobile apps more.

If you have DOM XSS you must write the source line it's on in the report.

Machine generated alternative text:
Attack Parameters (optional) 
Affected Source (optional) 
poorlywritten.js 
External Link (optional) 
Affected source line 
(optional) 
512 

How do we know it's that one segment of code and not another? I mean the 4 things are nice but we can't really know for certain, they could have multiple things in their code like that.

If you're lucky burp might say what lines are vulnerable if you do passive or active scan, so then you just have to trace through it on the actual app's source code to see which one it is.

Debugger in chrome and find when it executes. Might have to set some break points and stuff. Look for the obvious DOM XSS ones and set some break points. If it hits that when it executes then you're golden.

Machine generated alternative text:
Data is read from window.location.hash and passed to $() via the following statements: 
• var innerLink = window. location. hash. substring (1) ; 
• innerLink = innerLink. replace(/ / g, "820") ; 
• var frameTit1e = decodeURI (innerLink) ; 
• var matchFromHash = $ ("#whTocTree a [title = "' + framelitle + 
"'l: first"); 

Machine generated alternative text:
Verification 
Scenario 1 
It is possible to deliver an payload 
entering the p 
Dari d N 
XSS caps it 
ad into the search input field 

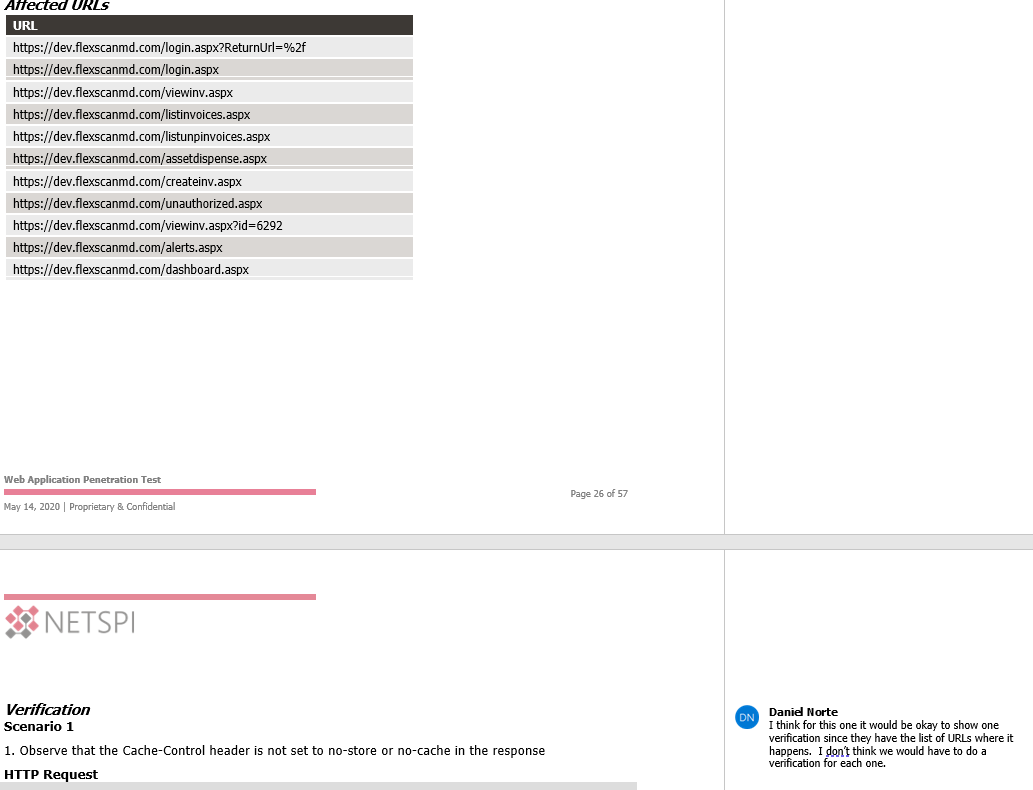
Machine generated alternative text:
3. Enter the payload into the notes box and dick update 
Place the payload: in the instructions box. 
Place payload in a script block or inline script 

**In the reports in Resolve we do not add findings to the report that are lower than a low. So no informational or ignore findings.**

**For Nessus findings look in** [What to do](onenote:#What%20to%20do&section-id={CC6FB3C2-CD8F-4D64-BCAE-3DC744447654}&page-id={051C4EDB-B55E-414F-AB98-9699DE9CE1A7}&end&base-path=https://netspi-my.sharepoint.com/personal/mallen_netspi_com/Documents/Marissa%20@%20Netspi/Resolve7%20WAPEN%20Checklist.one)

**Make sure to just click Duplicate in Resolve for a generated finding. So much easier than going into burp suite and trying to find the code to copy.**

**(Unless it's site wide, which just verify one and write site-wide in description)For stuff like X-Frames-Options, Inadequate cache control, mflac with multiple links that're just forced browsing.**



Good morning/evening everyone,

Today we're rolling out Live Reporting! In the hopes of keeping this short:

* Live reporting means publishing your vulnerabilities to the client at the end of each day (Only for PTaaS clients).
* If you are an SC 1 or SC 2, reach out to your manager to QA your vulnerabilities before publishing each day.
* **If PMO says your current project is a PTaaS client, go read the wiki here**[**https://wiki.netspi.com/index.php/PTaaS**](https://wiki.netspi.com/index.php/PTaaS)
* Remediation testing will see some process changes in the near future, I'll send out another email for that next month. For now shoot me a message if the client is requesting remediation testing in an unusual way.

Live reporting has been a large selling point to our new customers, we will be delivering vulnerabilities sometimes a month faster than any of our competitors could. This will also help push the client discussions into the engagement, instead of 3 weeks after the engagement during the readout when everyone has moved on to other work.

As always, this is a new process and will help us provide an improved client experience, but if you notice any ways this process can be improved/changed please don't hesitate to reach out. If you really don't like it, send complaints to [@Antti Rantasaari](mailto:Antti.Rantasaari@netspi.com) please.

Thanks,

Jake