Video Series
To Do list:

add a github page for remediation, selling it as the app team can customize this page for their own apps and what they know they need so the devs are checking for these things as needed. For example, could show how to implement param queries owasp article dos and donts.

Each youtube vid description can link to the checklist, summary payloads, sources for that specific video, and overall sources.

Start each video by explaining what checklist items we are covering and why they are risks. Do the tests, then show remediation.

Links:

https://github.com/swisskyrepo/PayloadsAllTheThings

to start machine: open postman, burp suite, start juice shop, open PS, start apache2 and mysql services.

Video 1: Introduction to the video series – what we are doing and why. (5-10 mins)

In this video show the list of videos and explain that more material will be linked throughout if you want to go learn and try more. Show the checklist and payload list.

Mention that this testing methodology is not a full pentest meaning your app is secure. Our goal is to do a quick and dirty sanity check on our app to prevent exploitation from an attacker and in a lot of cases ensure that the pentesting team isn't going to find a lot of low hanging fruit in your app. The best thing to do is get a full pentest or run a Burp scan over your app. For the sake of this entire video series, we are assuming that we are testing lower lane environments and we do not have access to pentesting tools because most dev teams do not. This would be a great task to give to a junior developer or someone on your team that wants to transfer into security. A junior would learn a lot about writing secure code. Make sure to mention that if you can't test your app, you can do the payload free sections and the SCA section.

Video 2: Intro to the tools – Show the tools that can be used to complete this series (10 mins)

open curl, postman, insomnia, burp. Show how we can get a request from the browser into each, then show sending a random payload in each so people have an idea of the flow.

-explain why we need a tool using encoding lab and change password in juice shop.

-show how to import a request into postman

-show how to use the browser with fetch

Optional: why we don't use other browsers video bc encoding on send. (5 mins)

Show how the different browsers encode payloads, so you have to be careful with your testing tool.

Video 3: Best coding practices and remediation overview. (10 mins)

This is not a secure coding practices course, it is a pentesting course designed for developers to find vulnerabilities in their own apps. However, I have set up sort of a template page for the app teams to fill out themselves with language specific links. We wouldnt have a lot of these vulns if the app teams coded securely, so this will try to nip our issues in the bud.

First, i recommend the entire dev team read the high level coding links.

Second, your code leads should all read the deeper links.

Then, your app team should work together to take the important pieces to build your own secure code development guidelines based on your companies risk appetite and security policies.

Again, this is more of a template with some examples, but here is how it could be used.

link to good coding practices and discuss blanket remediation like input validation, output encoding, fixing response headers and rate limiting will severely help. We can also discuss the basics of SCA and review. Show off the in progress github page.

https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/assets/docs/OWASP_SCP_Quick_Reference_Guide_v21.pdf

https://owasp.org/www-project-code-review-guide/assets/OWASP_Code_Review_Guide_v2.pdf

Testing section

Video 4: Picking endpoints to test and covering continuous review.(25 mins)

what endpoints are important to us and what can we see before sending payloads. We can mention cookies and response headers here, but really we are focused on the continuous review section of the checklist. Those might be priority targets for our payloads section. Can also cover what we can ignore, like js files and non sensitive endpoints.

Navigate around the app pointing out important endpoints and mention that we will use this method to select what we will use in future videos along with random one off labs.

If you know that user input is making its way into operating system commands, being reflected back on the page, or being included in SQL queries make sure to include these in your testing.

Aim: bwapp xml/xpath lab: click any button and see fatal error. This is bad from a user experience perspective, but also this helps an attacker figure out that they need to test out this endpoint. simplexml_load_file() tells us as the attacker what we should be trying here. Verbose error messages often lead to XSS as well.

Next show Juice shop login page, show the response itself to see the full error in inspect element network tab. The response shows an sqlite error. We can see exactly where our input is put into the SQL query.

Auto complete disabled and sensitive information in the URL. Show when we register it saves our inputs. If you logged in from the library someone could come behind you and see your answers from the registration page.Search for autocomplete=off

So if you ever get the google pop up to save credentials, that is a key indicator that autocomplete is not set to off.

Show that the post request to login is POST request meaning its encrypted over HTTPS and no machine in the middle can see it. However on the password change page we can see it in the URL meaning even over HTTPS anyone can see the new password

Video 5: Response headers. (15 mins)

Make sure to explain why at the start of each item

Ensuring secure response headers will help reduce impact of some attacks and even prevent execution of certain payloads.

There are nuances with apis, but we will mostly ignore that here.

Cache controls - talk about back button capability. And using a library computer may store things locally that an attacker could see afterwards.

CORS - if you had vulnerable cors on your app, facebook could send a request to your app then read the info in the response. Allowing them to enumerate sensitive information or even see if their attack is working depending on the setup of their exploit and what other vulnerabilities were chained with this.

Video 6: Cookies (10 mins)

Make sure to explain why along the way

Video 7: Cross-Site Scripting (XSS): 15 mins

Start with what can be done with XSS and if you want to learn more you can go to XYZ places. Go into Payloads, then jump into remediation.

This will be where we explain how to use the payload list to quickly scan over endpoints looking for weird interactions. Remember to point out the content type in the response for XSS or checking for CSP as helpers to see if its even possible to exploit.

Video 8: SQL Injection 15 mins

Cover risk, do a few labs showing how to use the full list of basic payloads because behavior can be different. Explain remediation. Make sure to cover the fact this requires a connection to SQLi, even if it's an API to another tool/team’s app.

Video 9: Directory Traversal 15 mins

Explain how this can lead into OS command injection and what parameters would be vulnerable.

Video 10: OS Command Injection 15 mins

Explain how this vulnerability arises and what parameters to test.

Video 11: SQL Injection on a login page 15 mins

Explain how this is different from the other SQL injection section.

Video 12: Login Page Issues 15 mins

Explain that account lockout can help prevent password brute forcing. Discuss time based username enumeration and explain that this requires tools showing response times.

Video 13: Session Management 15 mins

Video 14: Clickjacking 5 mins

Provide 2 HTML files on github.

Video 15: Malicious File Upload 5-10 mins

Provide XXE, SVG, HTML files.

Video 16: Outro 5 mins

Optional: NoSQL Inj 10 mins

Optional: XXE 10 mins

Optional: SSRF 10 mins

Youtube video descriptions:

Links in every video: Our github