avatao

How to build a security advantage from bugs

5 high-profile security issues and proven ways to turn them into insights for your company

Table of Contents

Are we 100% secure?

Let’s face it, perfect protection doesn’t exist when it comes to securing your company. With the ever-evolving threat landscape and sophisticated hackers, identifying, preventing and continually reassessing potential risks seems like a daunting task.

What if we told you that the future lies in your team’s hand?

One of the secret weapons in winning this battle is developing the security capabilities of your software engineering teams. Hopefully, that is exactly what you already do. You invest in your team and develop their security skills , thus they get familiar with cybersecurity fundamentals, OWASP top 10 and the basics of secure coding.

Then comes the question: **What is the next step after mastering the basic cyber skills?**

How can you upskill your team to a more advanced security mindset?

Playing the cat and mouse game with the right security mindset

Security breaches and bugs have been increasingly common and harmful in the last years. A few stand out, though, as instructive examples of how bugs have evolved and how attackers are able to use these bugs to carry out these attacks.

The future of preventing such security bugs and breaches starts with accepting the possibility of insecure code , learning from other organisations' mistakes and significantly raising the barrier to entry or the resources required to carry one off. To support companies in this endeavour we developed a security training program that equips developer teams with the right security mindset for winning the battles of the future.

Learning from high-profile cases from the frontline

Our team of cybersecurity experts is constantly on the hunt for tricky bugs, unusual cases of exploitations of vulnerabilities and out of the ordinary security breaches. Moreover, they are using their expertise to turn these into interactive learning tutorials and challenges so that advanced security champions, security and developer teams can learn from these high-profile cases.

These hands-on, instructive examples serve now as a key component of our security training program. We continually update our secure coding library with interactive learning modules based on recent security breaches and high-profile vulnerabilities.

They serve as hands-on, instructive use cases that

- enable developers to learn techniques and strategies for creative thinking and problem solving
- encourage them to think outside the box and help them develop a more comprehensive security mindset
- increase their capability to avoid and remediate security issues in coding and deployment to ship products faster

How will your company benefit from it?

Financial

A team prepared to prevent data breaches can save your business from the costs associated with them.

Customer relationship

Continuous security improvement will be embedded in your processes. Teams build the ability to react to security issues faster.

Operational

Your company will deliver a high level of service, security and expertise that will strengthen customer trust.

Education & Growth

Increased team capability to prevent potential future breaches.

In the next few pages we offer a preview on 5 scenarios from our security library that will guide your developer teams towards a security-inclusive development mindset.

1

XSS On Google Search bar

Vulnerability

Cross-Site Scripting (XSS)

Why it's interesting
How an already patched issue can show up again even on the main page of Google

What your team can learn from it
How HTML parsing behaviour can lead to a XSS vulnerability

A security researcher has found a XSS on the Google search bar itself. The Google security team confirmed and fixed the issue before attackers exploited it, however they forgot to write tests. Later on the fix was removed because of a bug, and since it didn't break any tests, it was deployed to production again. In this exercise we take a look at the HTML parsing behaviour that caused the vulnerability.

Security Tips

Browsers are really complex, so avoiding XSS is not trivial in some cases. Learning the whole HTML specification and JavaScript API is clearly an overkill, if it's even possible for a human.

- You can learn a lot by following security news, reading research papers, solving XSS challenges or simply reading and analysing writeups.
- Write tests. Even security tests!
- Try to minimize the attack surface. Do not use user input in HTML context, if it's not necessary.

Take a Deep Dive
Explore Avatao's How to build a security advantage from bugs Module

2

An unexpected HEAD on GitHub

Vulnerability

Cross-Site Request Forgery (CSRF)

Why it's interesting
A peculiar case of a perfectly secure code at first sight and a deadly bug under the surface

What your team can learn from it
How to look for hidden features in your framework

A bug bounty hunter found a CSRF issue on Github. The issue we present here is a great example of a hard-to-catch bug: Your code looks simple, secure and perfect, but a hidden feature in the framework outsmarts you and causes a deadly bug.

Security Tips

- Know the third parties you're working with.
- Take time to read the documentation.
- Configure properly, disable the unnecessary default features if possible, or at least keep them in mind and pay attention to them.

Take a Deep Dive
Explore Avatao's How to build a security advantage from bugs Module

3

Subresource integrity at British Airways

Vulnerability

Subresource Integrity

Why it's interesting
When it comes to third party software, your product is only as secure as the weakest link in the chain.

What your team can learn from it
Best practice on minimizing attack surface while using third party software

Nowadays it's almost impossible to avoid depending on third party software, thus we created a tutorial with a best practice on how to handle third party software securely. It's territory, where even big companies make mistakes. Better write that in stone: Your product is only as secure as the weakest link in the chain.

Security Tips

- Always pin third-party software versions to avoid executing malicious code in your product.

- If the software is a third party JavaScript file in your web application, then you should use subresource integrity checks, which blocks the script execution if it's modified.

Take a Deep Dive
Explore Avatao's How to build a security advantage from bugs Module

4

Vulnerability

Web Cache Poisoning

Web Cache Poisoning - RedHat, Unity3D, Cloudflare blog

Why it's interesting
A really out of the ordinary vulnerability: Turning caches into exploit delivery systems by leveraging unexpected web features

What your team can learn from it
Even if you can't imagine how someone could exploit an issue, it doesn't mean it's impossible.

Web applications are usually using cache, which is really useful to save resources and gain some speed as well. If a visitor requests a page, it's saved to the cache and returned from the cache if another visitor requests the same page. The problem is that attackers could send maliciously crafted requests, which could be really different from the ones the browser sends. This way everything becomes user input, which usually isn't. If some parts (headers) of this request are reflected in the response and this response is cached, then attackers can inject malicious code in the page, which is then returned to other visitors from the cache.

Security Tips

- Sanitize user input even if it seems unlikely to spoof. E.g. e-mail address, HTTP headers, etc.

Take a Deep Dive
Explore Avatao's How to build a security advantage from bugs Module

5

Web Cache Deception - Paypal

Vulnerability

Web Cache Deception

Why it's interesting
Your components seem secure, but what happens when you pair them?

What your team can learn from it
A holistic approach to secure configurations

In this interactive exercise we're tricking the cache to save sensitive, user-specific data to leak it. For example we're telling our victim to visit: https://paypal.com/my_balance/random.jpg. Let's assume PayPal is basically ignoring the 'random.jpg' part and simply returns the balance, however the cache thinks it's an image file. Now if the attacker sends a request to the same URL it won't even reach the servers of PayPal, because the cached version of the page will be returned, which could result in a serious data leak.

Security Tips

- It's not enough to have components that are secure themselves. Make sure the system is secure when the components are connected together.

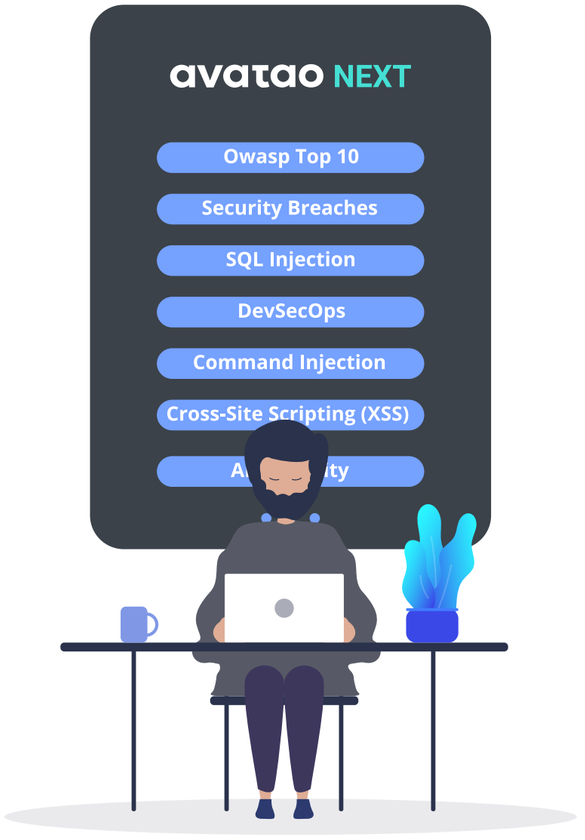
Take a Deep Dive
Explore Avatao's How to build a security advantage from bugs Module

Avatao’s approach to security training

Avatao’s security training goes beyond simple tutorials and videos offering an interactive job-relevant learning experience to developer teams, security champions, pentesters, security analysts and DevOps teams.

With 600+ challenges and tutorials in 10+ languages, the platform covers a wide range of security topics across the entire security stack from OWASP Top 10 to DevSecOps and Cryptography.

The platform goes beyond secure coding, it immerses developers in high-profile cases and provides them with real, in-depth experience with challenging security breaches. Engineers will actually learn to hack and patch the bugs themselves. This way Avatao equips software engineering teams with a security mindset that increases their capability to reduce risks and react to known vulnerabilities faster. This in turn increases the security capability of a company to ship high-quality products.



See how Avatao Works →

Take Avatao on a test drive →