Mohammed Essehemy 👋

OWASP TOP 10

The 10 most critical web application security vulnerabilities, as identified, and agreed upon, by security experts from around the world.

Me, 🤓

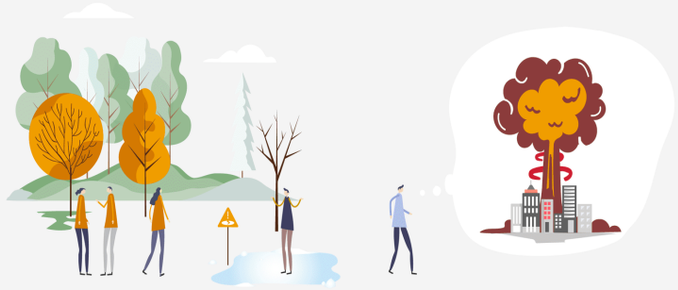
Interests

- Open source 👭
- New Technologies.

WORK 💻

- Node.js Engineer - Vodafone EG (10 months)
- Senior Backend Engineer - Fixed Solutions. (now)

As Node.js developers building web applications, we understand that we live in rough times.



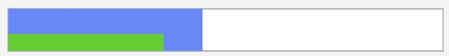
3/4 of us

believe our job requires us to take security seriously

More than 1/3 of us

believe our organizations will be the target of a large-scale attack in the next six months

40% *of us are confident in the security of our application*



100%

31% *of us are confident our code doesn't contain vulnerabilities*

WHAT ABOUT CODE WRITTEN BY OTHERS ?

CONFIDENCE IN
NODE.JS CORE

84%

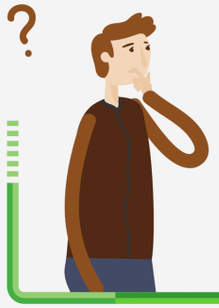
are moderately to very confident in the security of Node.js core.

CONFIDENCE IN
THIRD-PARTY MODULES

40%

of us feel that third-party packages pose the greatest risk to application security.

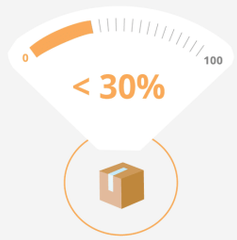
While only **16%** of us are confident the third-party packages we use are vulnerability-free.



Given this healthy skepticism around the security of the code we're using, it would seem logical for us to seek out the best possible tools to help secure our applications.

BUT SURPRISINGLY !
THAT'S NOT WHAT HAPPENS

Less than a third of us **combine manual and automatic code reviews to search for flaws.**



Despite our strong concerns about third-party packages, less than a third of us **use automated tools to discover vulnerable packages.**

How do you make sure your code doesn't contain vulnerabilities ?

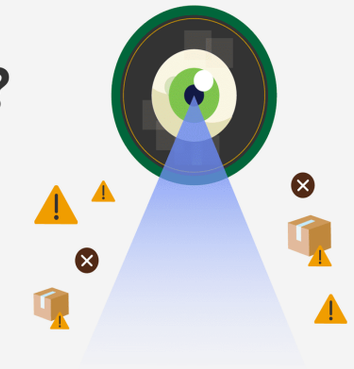
Manual code reviews only	44%
Automated code reviews only	13,5%
Manual and automated reviews	30%
No reviews of any kind	12%

How do you verify there are no known vulnerabilities in your packages ?

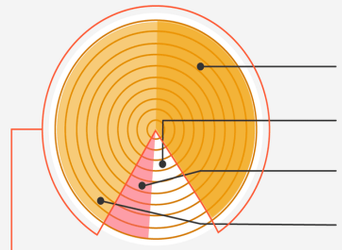
Manual checks	26%
Automated checks	30%
No review of any kind	40%
Other	4%

So while more than two-thirds of us — including 85% of C-suite executives — consider security to be an important part of their respective roles, **we don't always prioritize identifying vulnerabilities in our own code or in third-party packages.**

Out of sight, out of mind ?



Prevention is a key piece of the security puzzle, **but identification and remediation of attacks is also critical**. How do you know if your application is under attack?



44% look at logs

11% look at an APM tool

9% use a SIEM solution

35% don't know for sure

Shockingly,
the vast majority of us have poor to no insight as to when we are actually under attack, less than a quarter of us use any form of real-time protection against attacks.

Conclusion

In short, **we understand well the risks** of operating in the open internet. We understand the complexities of building reliable, secure code (because we are pessimistic that we are actually doing that). **But we are unwilling to take advantage of tools that can help us** identify and mitigate threats. We don't even know if we are under attack. It's hard to square these facts. It's time to put actions to our beliefs, and take charge of our security

!

Powered by

sqreen & NODESOURCE®

A1- Injection

Subtypes

1- Server Side JS Injection

2- SQL/NOSQL Injection

3- Log Injection

eval(), setTimeout(), setInterval(), Function();

while(1), process.kill(process.pid)

db.accounts.find({username: username, password: password});

{“username": “admin”,"password": {$gt: “"}}

A2- Broken Authentication

Subtypes

1- Session Management	No timeout, MITM, hash Password
2- Password Guessing	Password length, complexity, user/password pair

A3- Sensitive Data Exposure

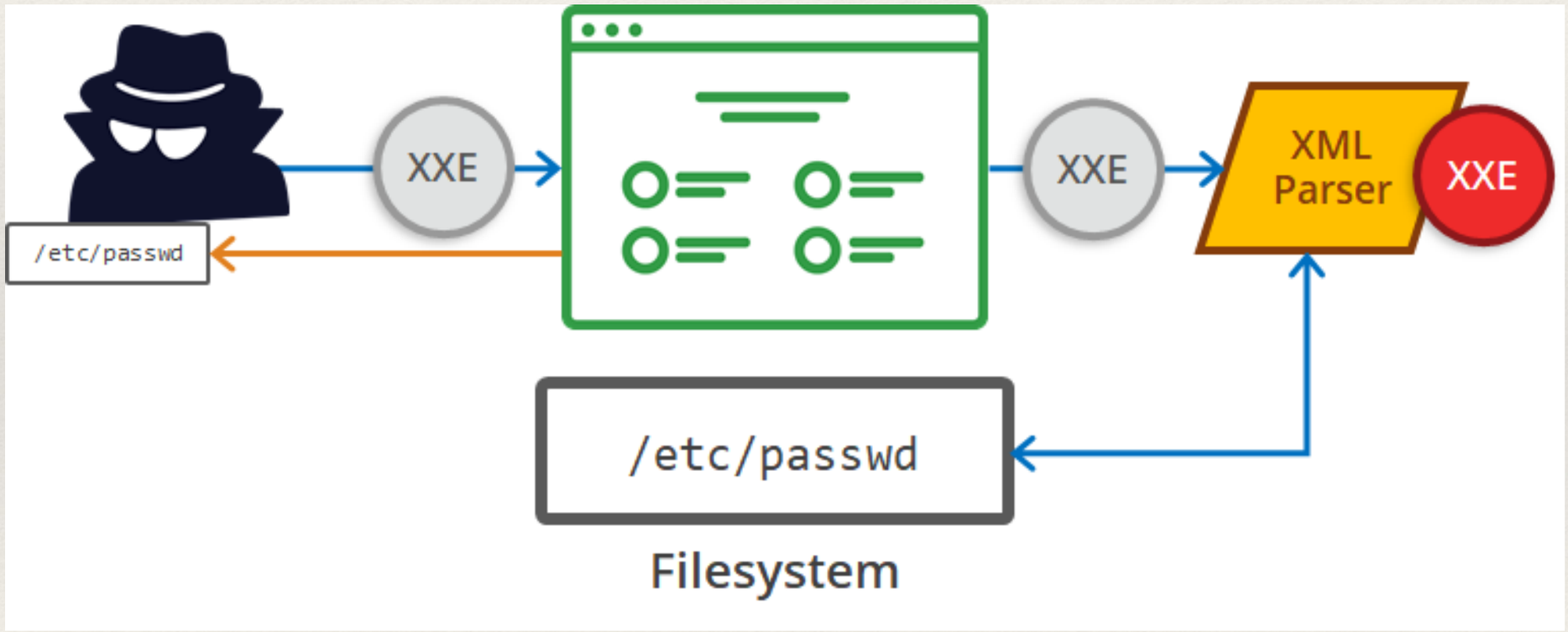
Mitigation

HTTPS

Don’t store sensitive data unnecessarily. Discard it as soon as possible.

Ensure strong standard algorithms and strong keys are used.

A4- XML External Entities

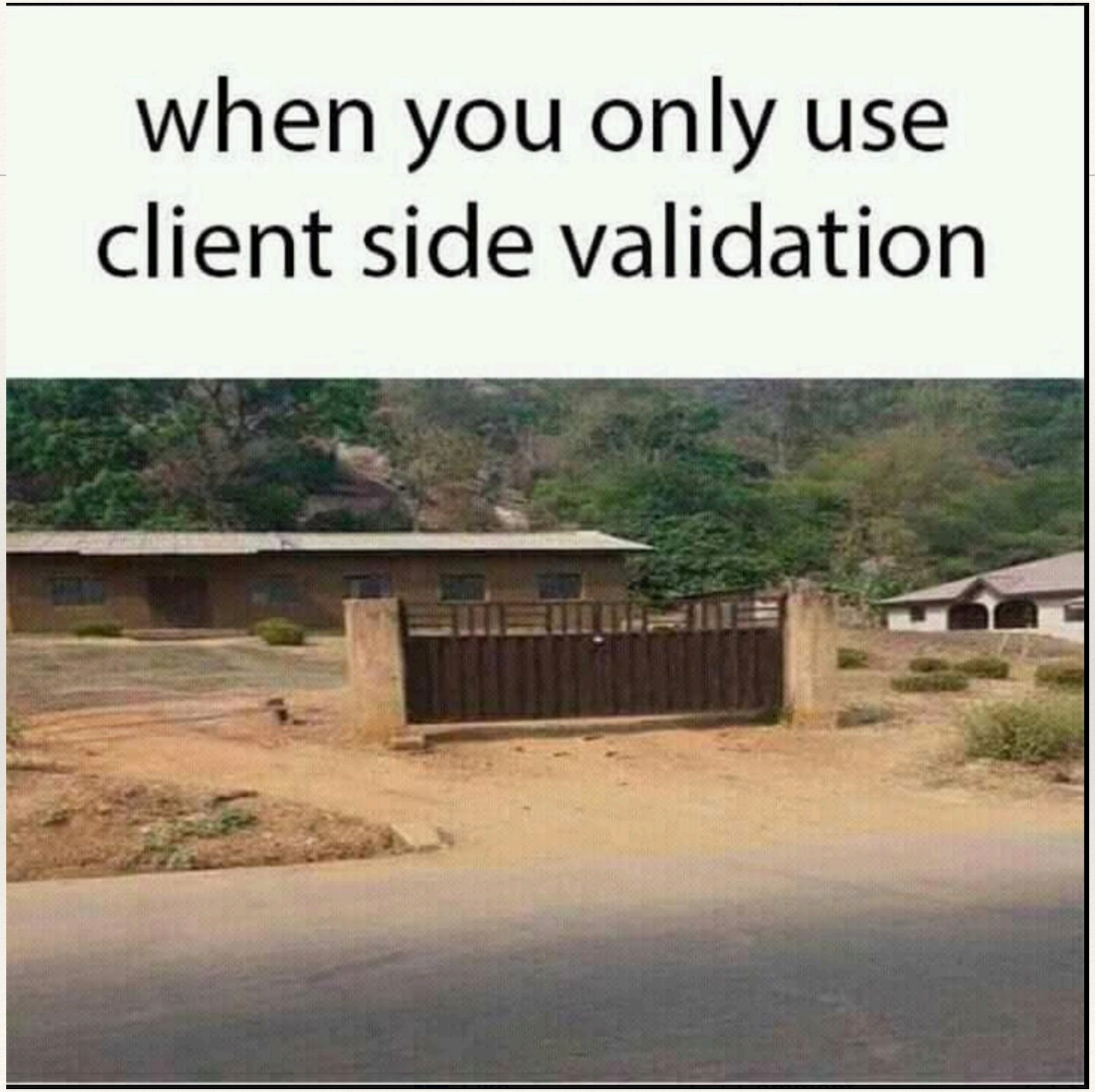


A5- Access Control

Mitigation

Check Access

Don’t ask user for his id


when you only use
client side validation

A6- Misconfiguration

Subtypes

Unused data sources (pages, apis, …)

Default Accounts

App with root privileges

A7- XSS

Mitigation

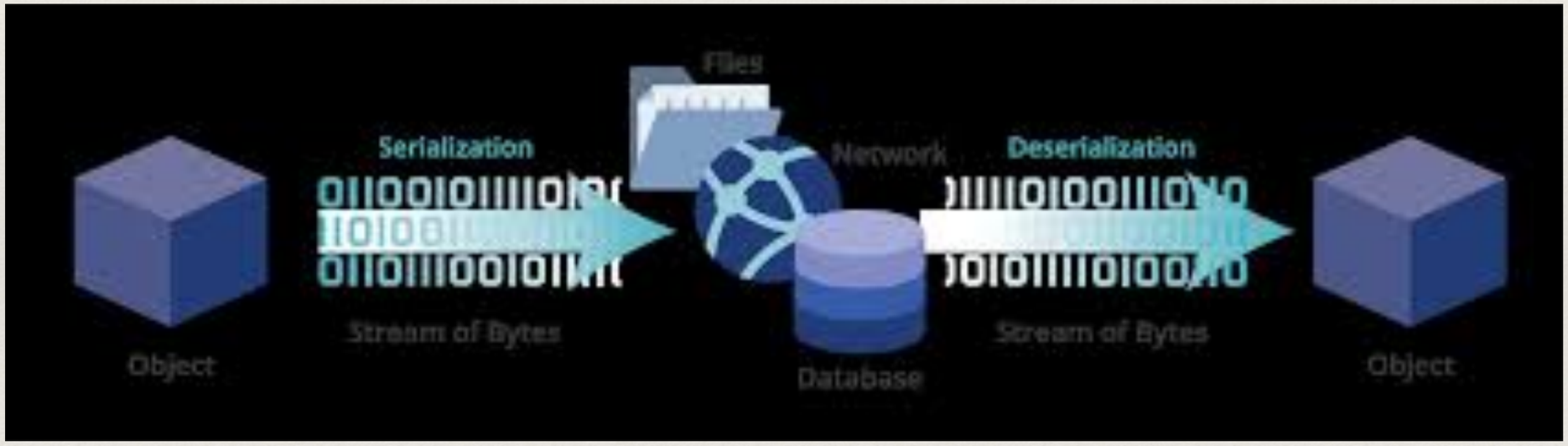
Input validation and sanitization

Output encoding for correct context

HTTPOnly cookie flag

Content Security Policy (CSP)

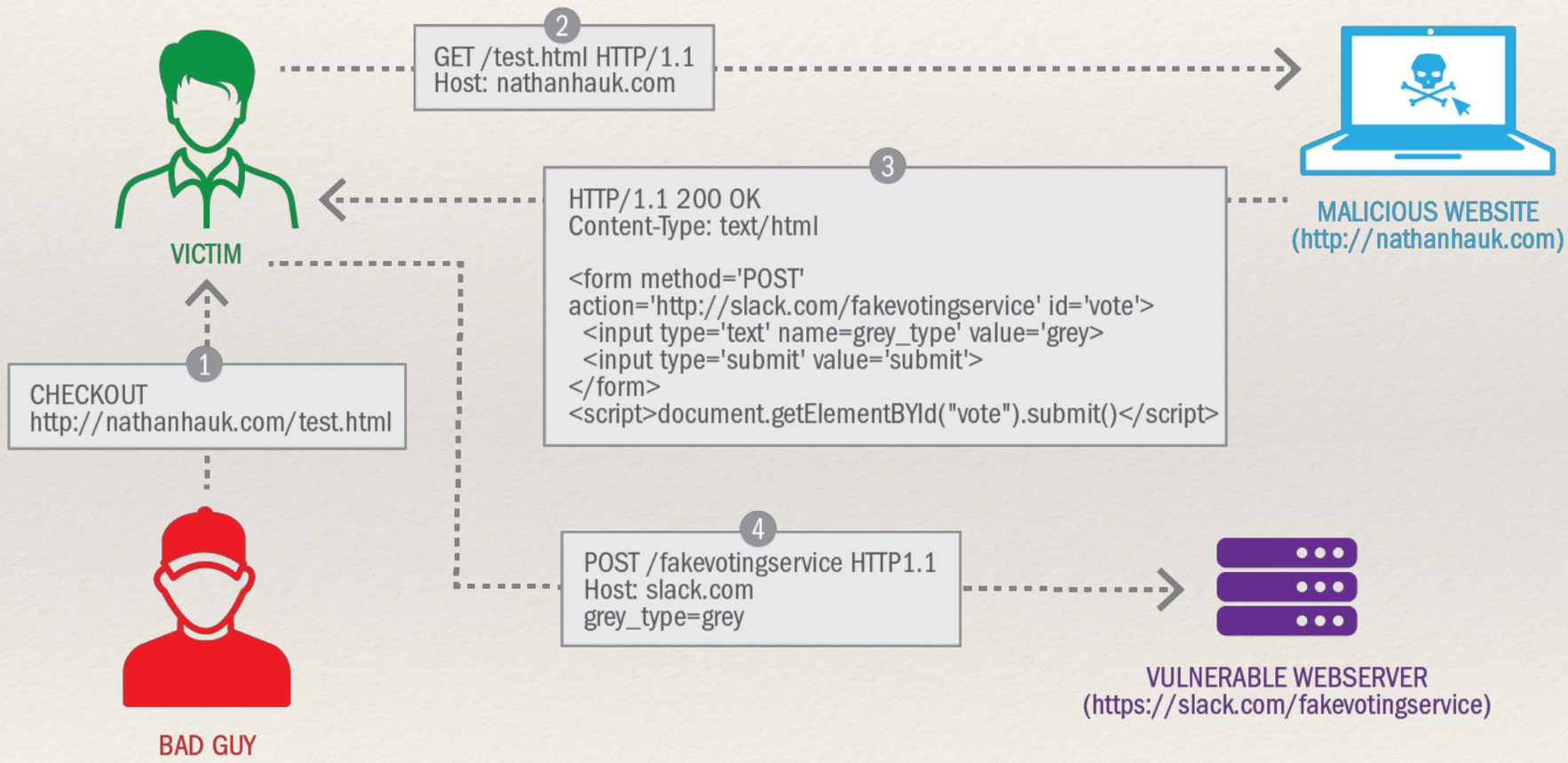
A8- Insecure Deserialization



A9- Components with Known Vulnerabilities

A10- Insufficient Logging & Monitoring

2013-A8- CSRF


2
GET /test.html HTTP/1.1
Host: nathanhauk.com
MALICIOUS WEBSITE
(http://nathanhauk.com)
3
HTTP/1.1 200 OK
Content-Type: text/html
<form method='POST'
action='http://slack.com/fakevotingservice' id='vote'>
<input type='text' name=grey_type' value='grey>
<input type='submit' value='submit'>
</form>
<script>document.getElementBYId("vote").submit()</script>
VICTIM
1
CHECKOUT
http://nathanhauk.com/test.html
BAD GUY
4
POST /fakevotingservice HTTP1.1
Host: slack.com
grey_type=grey
VULNERABLE WEBSERVER
(https://slack.com/fakevotingservice)

2013-A10- Unvalidated Redirects and Forwards