Welcome to the 42Crunch Platform
Start protecting your APIs. Please log in below.

SEC-DEV-OPS
AN AUTOMATED APPROACH TO API SECURITY

ISABELLE MAUNY - CTO & CO-FOUNDER
ISABELLE@42CRUNCH.COM
For **2010**, the OWASP Top 10 Most Critical Web Application Security Risks are:

- A1: Injection
- A2: Cross-Site Scripting (XSS)
- A3: Broken Authentication and Session Management
- A4: Insecure Direct Object References
- A5: Cross-Site Request Forgery (CSRF)
- A6: Security Misconfiguration
- A7: Insecure Cryptographic Storage
- A8: Failure to Restrict URL Access
- A9: Insufficient Transport Layer Protection
- A10: Unvalidated Redirects and Forwards

For **2013**, the OWASP Top 10 Most Critical Web Application Security Risks are:

- A1: Injection
- A2: Broken Authentication and Session Management
- A3: Cross-Site Scripting (XSS)
- A4: Insecure Direct Object References
- A5: Security Misconfiguration
- A6: Sensitive Data Exposure
- A7: Cross-Site Request Forgery (CSRF)
- A8: Failure to Restrict URL Access
- A9: Insufficient Transport Layer Protection
- A10: Unvalidated Redirects and Forwards

**Top 10 2017**
- A1:2017-Injection
- A2:2017-Broken Authentication
- A3:2017-Sensitive Data Exposure
- A4:2017-XML External Entities (XXE)
- A5:2017-Broken Access Control
- A6:2017-Security Misconfiguration
- A7:2017-Cross-Site Scripting (XSS)
- A8:2017-Insecure Deserialization
- A9:2017-Using Components with Known Vulnerabilities
- A10:2017-Insufficient Logging&Monitoring
A FEW EXAMPLES
Content Injection Vulnerability in WordPress

FEBRUARY 1, 2017  •  MARC ALEXANDRE MONTES

Security Risk: Severe
Exploitation Level: Easy/Remote
DREAD Score: 9/10
Vulnerability: Privilege Escalation / Content Injection
Patched Version: 4.7.2

As part of a vulnerability research project for our Sucuri Firewall (WAF), we have been auditing multiple open source projects looking for security issues. While working on WordPress, we discovered a severe content injection (privilege escalation) vulnerability affecting the REST API. This vulnerability allows an unauthenticated user to modify the content of any post or page within a WordPress site.

https://blog.sucuri.net/2017/02/content-injection-vulnerability-wordpress-rest-api.html
Vulnerability Spotlight: Multiple Vulnerabilities in Samsung SmartThings Hub

These vulnerabilities were discovered by Claudio Bozato of Cisco Talos.

- Smart locks controlled by the SmartThings Hub could be unlocked, allowing for physical access to the home.
- Cameras deployed within the home could be used to remotely monitor occupants.
- The motion detectors used by the home alarm system could be disabled.
- Smart plugs could be controlled to turn off or on different things that may be connected.
- Thermostats could be controlled by unauthorized attackers.
- Attackers could cause physical damage to appliances or other devices that may be connected to smart plugs deployed within the smart home.

- Remote Command Execution (A1)
- SQL Injection (A1)
- JSON injection (A1)
- Information Leakage (A3)
- Broken Access Control (A5)

The livestream configuration can be changed from the smartphone application. When a user changes the smart camera's password, the following request is sent to api.smarthtings.com via HTTPS:

```
PUT /elder/<location-id>/api/devices/<device-id>/pages/preferences HTTP/1.1
X-BR-API-Version: 3.2
X-ST-Client-AppVersion: 2.14.0
Authorization: Bearer <auth-token>
Accept: application/json
Content-Type: application/json; charset=UTF-8
Host: api.smarthtings.com
User-Agent: okhttp/3.6.0

...{
"completedSetup":true,"label":"SNH-V6414BN","preferences":[
{"name":"wlanMode","value":"PSK"},
{"name":"cameraBrightness","value":"Med"},
{"name":"wlanPassword","value":""},
{"name":"cameraTimeZone","value":4},
{"name":"cameraPassword","value":"<camera-password>"},
{"name":"cameraOverlay","value":"Off"},
{"name":"cameraDayNightMode","value":"Auto"},
{"name":"cameraImage","value":"No Flip - No Mirror"},
{"name":"wlanSite","value":""},
{"name":"cameraWDR","value":"WDR"},
{"name":"cameraAudio","value":"Low/Med"}]
```
Information Leakage (A3)
Broken Access Control (A5)


PAYING THE PRICE FOR DISRUPTION
How a FinTech Allowed Account Takeover

Vincent Haupert  
Friedrich-Alexander University Erlangen-Nuremberg  
v Vincent.haupert@cs.fau.de

Dominik Maier  
TU Berlin  
dmaier@seclab.tu-berlin.de

Tilo Müller  
Friedrich-Alexander University Erlangen-Nuremberg  
tilo.muller@cs.fau.de

well-established security standards of the banking sector are sometimes neglected. This paper presents a range of security vulnerabilities that were identified in N26, a FinTech company. N26, which currently has 500,000 registered users in countries across Europe, offers a smartphone-only bank account. Without access to a user’s smartphone, we were able to not only reveal N26 customer data and manipulate transactions in real-time but also completely take over a victim’s bank account. The bugs, taken together, would have eventually put the customers’ money and data at risk. We believe a security-focused development process would have discovered most of these bugs early on. This raises the question of whether startups should ignore well-established security concepts in favor of a quick-market launch. Also, the focus on usability aspects over security plays a key role in the advancing downfall of conceptual security measures.

We demonstrate how little thought N26 put into security by presenting a variety of practical exploits against it. These exploits were found in all parts of the N26 infrastructure and have since been fixed in the course of a responsible disclosure process. By combining these exploits, we show that even a complete account takeover was possible. We also provide an overview of the state of the art in mobile banking and shed light on the current state of transaction schemes used for banking. With the example of N26, we argue why basic concepts were flawed and how well-established banking standards were disregarded. We also discuss the current legal situation of banks in Europe with regard to transaction security. By highlighting the attacks described in this paper, we hope to raise awareness among investors, state actors, and founders alike in order to emphasize the need to include security analyses early on in the development process.

2 BACKGROUND

In this section, we elaborate on how online transactions and their security mechanisms evolved over time. Today, many FinTech companies like N26 are best described as app-only banks. Everything, including opening accounts, sending transactions, applying for loans and more is done through the central app. Of course, established banks have offered traditional online banking through the browser for much longer. Since N26 operates with a German banking license, we focus on the history and regulations of German online banking. Soon, however, supranational regulations will enforce similar rules throughout the European Union, as outlined in the following section.

2.1 Online and Mobile Banking

In the German market, even in 1980 when online banking was first made public, every transaction had to be secured by an extra
MasterCard ID is printed on the card.

However, each transaction contains the following:

```
{
    "amount": -0.11,
    "cardId": "b8484ca2-a674-4f1c-af81-896e3bfe6d15",
    "linkId": "0123456789-372287",
    "merchantCity": "DUESSELDORF",
    "merchantCountry": 0,
}
```

MasterCard ID is part of every MasterCard transaction!
Failed to properly validate that you can’t input any other number than yours!

T-Mobile Website Allowed Hackers to Access Your Account Data With Just Your Phone Number

The bug exposed customers’ email addresses, their billing account numbers, and the phone’s IMSI numbers. T-Mobile has patched the bug.
How I Hacked BlackHat 2018

Enumerating registered BlackHat attendees with the BCard API

BlackHat is one of the world’s largest cybersecurity events which takes place in the USA in Las Vegas every summer (https://www.blackhat.com/us-18/). Those who have attended BlackHat may have noticed that their badge contains an NFC tag. This NFC tag is scanned at booths in the Business Hall so vendors can collect their marketing data including name, address, company, job title, and phone number. Following BlackHat, attendees who have had their badges scanned by various vendors then receive a barrage of marketing emails. One thing I was not aware of initially was what data was actually contained within the tag itself.
The Apache Software Foundation Blog

Sat, September 09, 2017

Apache Struts Statement on Equifax Security Breach

UPDATE: MEDIA ALERT: The Apache Software Foundation Confirms Equifax Data Breach Due to Failure to Install Patches Provided for Apache® Struts™ Exploit

The Apache Struts Project Management Committee (PMC) would like to comment on the Equifax security breach, its relation to the Apache Struts Web Framework and associated media coverage.

We are sorry to hear news that Equifax suffered from a security breach and information disclosure incident that was potentially carried out by exploiting a vulnerability in the Apache Struts Web Framework. At this point in time it is not clear which Struts vulnerability would have been utilized, if any. In an online article published on Quartz.com [1], the assumption was made that the breach could be related to CVE-2017-9805, which was publicly announced on 2017-09-04 [2] along with new Struts Framework software releases to patch this and other vulnerabilities [3][4]. However, the security breach was already detected in July [5], which means that the attackers either used an earlier announced vulnerability on an unpatched Equifax server or exploited a vulnerability not known at this point in time --a so-called Zero-Day-Exploit. If the breach was caused by exploiting CVE-2017-9805, it would have been a Zero-Day-Exploit by that time. The article also states that the CVE-2017-9805 vulnerability exists for nine years now.
JWT TOKEN VALIDATION

- Failed to properly validate JWT token (A5)
  - Algorithm (set to None)
  - Claims (audience)

Auth0 Security Bulletin CVE 2018-6873

Published: April 4, 2018
CVE number: CVE-2018-6873
Credit: Cinta Infinita

Critical vulnerabilities in JSON Web Token libraries

Which libraries are vulnerable to attacks and how to prevent them.
HOW DID WE GET THERE?
WE ARE HUMANS!

IN THIS CORNER, WE HAVE FIREWALLS, ENCRYPTION, ANTIVIRUS SOFTWARE, ETC. AND IN THIS CORNER, WE HAVE DAVE!!
For the security we said we’d hire someone to...

Hold on, that’s not the priority. Let’s get the project started first, we’ll see about that later...

Hey the project is nearly done, we should do a security audit to...

Too late man, we don’t have the time or the budget for that.

The site’s been online for 10 days, and we haven’t done any security testing, I’m nervous...

Don’t worry, we’ll take care of it later...

DOWN! THE SITE IS DOWN!
WE’VE BEEN HACKED!

How could you let this happen!?
COVERING THE BASICS
KEEP CALM AND TRUST NONE
KNOW YOUR APIS

See: https://www.owasp.org/index.php/Application_Threat_Modeling
VALIDATE AND SANITIZE INPUT

- URL validation
- Verb validation
  ✓ Reject if not valid
  ✓ Reject if user not-authorized
- Query params validation
  ✓ Min / Max / Pattern-based matching
- Content-Type validation
  ✓ Don’t accept as-is!

- Accept Header validation
  ✓ Don’t copy into Content-Type
- Data inbound
  ✓ Format
  ✓ Message Size and complexity
- Data outbound
  ✓ Data Leakage
  ✓ Exception Leakage
  ✓ Use rules against data dictionary

OPEN API to the RESCUE!
Don’t trust the incoming token!
- Validate algorithm (the one you chose!)
  - HS256
  - RS256 (recommended)
- Reject None
- Validate signature
  - Prefer digital signatures over HMAC
  - If not, be careful of key exchange
- Validate standard claims
- Add your own claims

OPEN API to the RESCUE!
EXTERNAL TOKEN VS. INTERNAL JWT

- **Token Server**: Validate/Issue token

- **Reverse Proxy**: Process token

- **Microservices**: Consumer of token
FINE-GRAIN AUTHORIZATION

Who is calling?
✓ Is it your own app?
✓ Is it a trusted user?

What can they do?
Example: T-Mobile number
Scopes are often not enough!
✓ Need ABAC solution
✓ SAML!
FINDING VULNERABILITIES
START EARLY: SHIFT LEFT!
COVER THE BASICS

Automated Analysis
- Static code analysis
- 3rd party libraries validation (CVEs)
  - NPS / Snyk for Node.js
  - GitHub dependencies graph

Manual Analysis
- Code reviews
HACK YOURSELVES!

API Scanning/Fuzzing
- Veracode, Coverity
- 42Crunch (alpha)

Further Analysis
- Bug Bounty
- Pen Testing
PROTECTION
Deploy security measures such as API Security Gateways/Firewalls

- Introduce Rate Limiting
  - Brute force attacks (see N26!)
  - DOS attacks
- Security Policies automatically applied and enforced
- Serves as Virtual Patching for protection
- Deploy at the edge and/or close to APIs (microservices architecture)
Use Development ticketing system for tracking issues

Analyse runtime behaviour and raise issues automatically
42CRUNCH DEV-SEC-OPS CYCLE FOR APIS

**Develop**
Develop and document API with OpenAPI/Swagger

**Deploy**
Deploy to containerized PEP

**Monitor**
Monitor Security Vulnerabilities and runtime behavior

**Test**
Continuous API hardening including API fuzzing

**Assess**
Assess API description and evaluate risk level

**Protect**
Configure and apply security policies from assessed risk
HOW SECURITY PEOPLE FEEL ABOUT APIS
RESOURCES

- OWASP Top 10

- OWASP DevSlop Project

- Chaos Engineering
  - http://principlesofchaos.org
  - https://github.com/dastergon/awesome-chaos-engineering

- OWASP ZAP

- Source Code Analysis

- Code Security reviews

- Systems Scans
RESOURCES

- SSL Setup Scan
  - https://hardenize.com
  - https://securityheaders.io
  - https://www.ssllabs.com/ssltest/

- Threat Modelling
  - https://www.owasp.org/index.php/Application_Threat_Modeling
  - Attacks Type Information
    - XSS: https://excess-xss.com
    - Buffer Overflow: https://www.youtube.com/watch?v=1S0aBV-Waeo
    - SQL injection: https://www.youtube.com/watch?v=ciNHn38EyRc
    - Cookie stealing /XSS: https://www.youtube.com/watch?v=T1QEs3mdJoc

- Pixi / DevSlop
  - https://github.com/DevSlop/Pixi
  - https://devslop.co

- JWT as session data