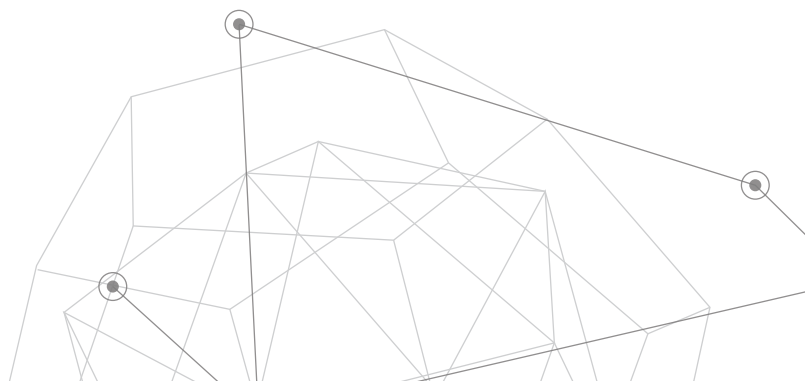



# The Curious Case of API Security

**Solving the Top 11 API Threats**

By Gunnar Peterson





By channeling the spirit of Sherlock Holmes and Hercule Poirot, we'll explore how to learn from failure, improve pattern recognition of security quality, and detect possible security vulnerabilities.

Just as a detective studies a crime scene for clues, we will follow a methodical approach to investigating and solving the Top 11 API Threats. Our basic process is as follows:

- 1 Understand the context in which APIs exist**
- 2 Look for clues that point to possible vulnerabilities**
- 3 Catalog the tools used to identify and track vulnerabilities**
- 4 Identify countermeasures to fix vulnerabilities**
- 5 Provide evidence that can measure the efficacy of the countermeasures**



APIs are simultaneously a great benefit and a significant threat for the enterprise. Serving as gateways to important functionality and data, APIs enable enterprises to make key resources available to developers, mobile apps, consumers and other companies. This benefit is what also makes APIs a threat, because they represent an extremely valuable target for attackers.

The job for security architects is to ensure that APIs offer the right functionality without also giving attackers a key to the enterprise kingdom. That's trickier than it sounds. This paper explores the most critical API threats and the corresponding countermeasures to help you thwart them.

# 01 | The Curious Case of Unprotected APIs

UNPROTECTED APIS	
<b>CONTEXT</b>	
Most enterprise cores are as soft and chewy as the center of a candy bar. That means that once inside, an attacker has free reign. Therefore, the API layer is a table-pounding, must-have security priority.	
<b>VULNERABILITIES</b>	<b>IDENTIFICATION &amp; TRACKING</b>
REST, SOAP and other APIs that make access available to back-end systems lack access control, monitoring and management	Build a service repository or API catalog Keep repository up to date to reflect changes
<b>COUNTERMEASURE(S)</b>	
Enforce access policy to all APIs through a central chokepoint such as an API gateway. Implement an API gateway to: <ul style="list-style-type: none"><li>• Mediate and monitor all access requests to the API layer</li><li>• Enforce API access control policy</li><li>• Ensure the system does not expose unprotected assets via APIs</li></ul>	
<b>ASSURANCE</b>	
Use dynamic scanning tools to look for exposed APIs. These scanning tools should run continuously.	





### **Case in Point: Unprotected APIs**

**APIs often introduce unwanted second- and third-order effects to the internal enterprise core. The U.S. National Weather service<sup>1</sup> developed an Android application that connected to its core systems via an API. The API layer had the ability to make unfettered requests to the internal core systems, which resulted in the internal core National Weather Service system going down due to an external Denial of Service attack.**

**This unprotected API threat should be a wake-up call for every security architect. Unlike “What happens in Vegas stays in Vegas,” what happens on the external API layer does not stay external. APIs are not a blocking layer, they are an admission layer. Anything admitted to the enterprise core needs strict scrutiny, and that begins with managing the API layer with an API gateway.**

<sup>1</sup> <http://www.forbes.com/sites/jameslyne/2014/08/26/android-app-causes-national-weather-service-website-blackout/>

## 02 | The Curious Case of Weak Authentication

### WEAK AUTHENTICATION

#### CONTEXT

Once companies bring their APIs under management via an API gateway, the next job is to answer the foundational question — Who are you?

APIs are designed to be exposed externally, so they cannot trust who is calling them. APIs have to authenticate users to be able to tell friend from foe.

#### VULNERABILITIES

There are numerous vulnerabilities in authentication protocols. Exploring weaknesses should begin with:

- Unauthenticated access (open APIs)
- Poorly protected secrets and tokens
- Use of password-based authentication
- Hard-coded secrets
- Lack of replay protection
- Guessable secrets and tokens

#### IDENTIFICATION & TRACKING

Follow the authentication lifecycle from end to end, from login request to verification to usage to termination.

Look for use of approved authentication standards.

#### COUNTERMEASURE(S)

API authentication is best analyzed in two parts:

1. Login authentication and minting the API token
2. Validating the calling application's token (This is subtly different from initial login because it's the token that is authenticated by the API layer.)

To implement a stronger API authentication approach, consider SAML and OAuth over TLS as a way to issue and verify API authentication for API consumer applications.

#### ASSURANCE

Use application-level testing to verify the use and strength of approved API authentication protocols.

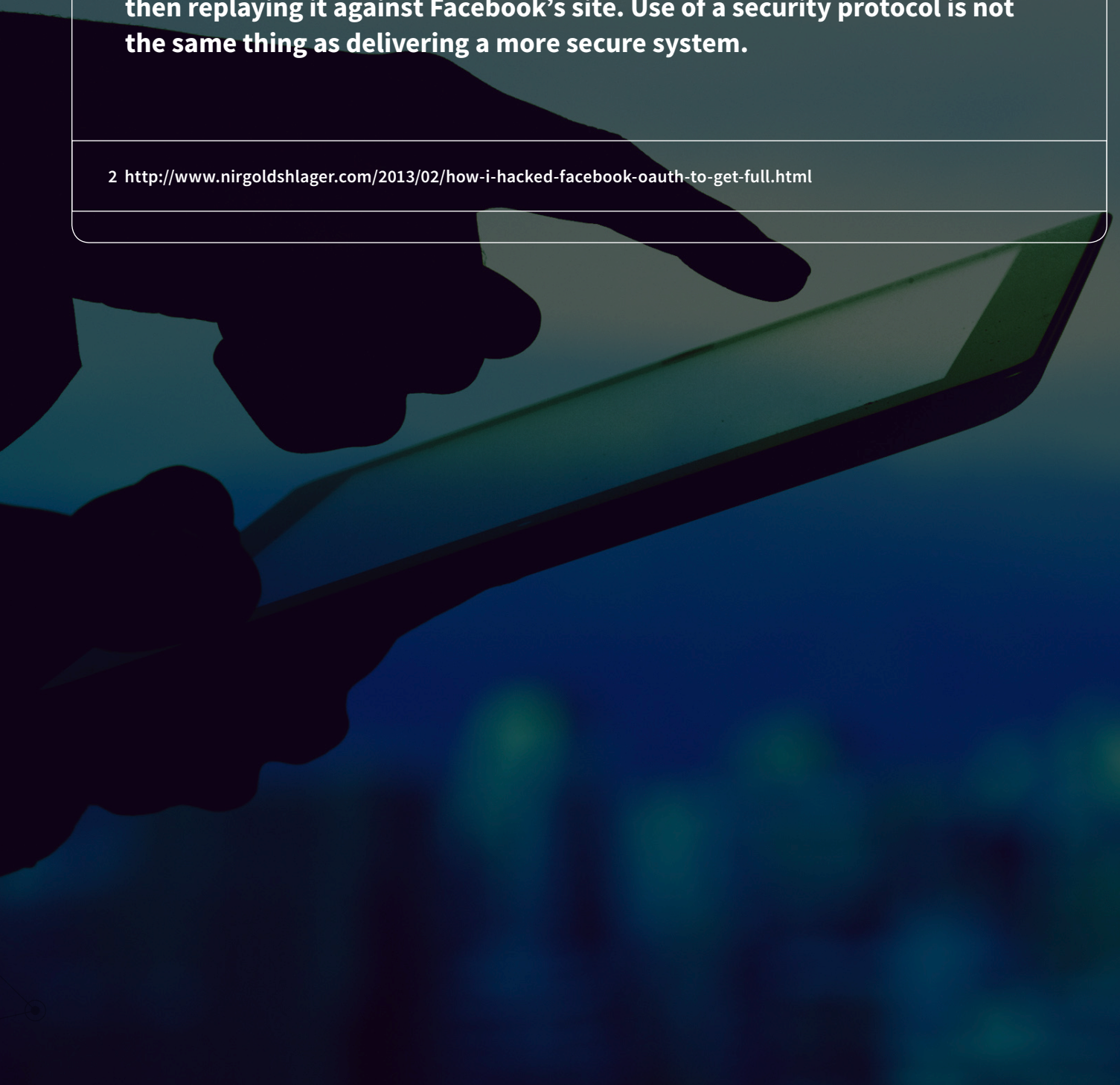
All protocols have vulnerabilities. Non-standard protocols should be heavily scrutinized. Even if industry standards like SAML and OAuth are implemented, thorough testing should be performed to check for replayability, session bugs, storage, scoping issues and protocol-level vulnerabilities.



### **Case in Point: Weak Authentication**

**Facebook's OAuth implementation was vulnerable to an attack<sup>2</sup> whereby an attacker could gain full access to any Facebook user account. The attack worked by using a covert redirect, stealing the user's token and then replaying it against Facebook's site. Use of a security protocol is not the same thing as delivering a more secure system.**

<sup>2</sup> <http://www.nirgoldshlager.com/2013/02/how-i-hacked-facebook-oauth-to-get-full.html>



## 03 | The Curious Case of Brute Force

### BRUTE FORCE

#### CONTEXT

Unfortunately, computers are fairly stupid, and this is especially true of security protocols. In the real world, if you were to punch someone in the stomach, they would do one of two things: punch you back or run away. The one thing they would not do is say “try again.” Yet this is how most security protocols function.

Authentication systems are built on at least one secret that the user knows and the attacker does not. Attackers leverage misplaced trust that secrets cannot be reverse-engineered.

#### VULNERABILITIES

Attackers use brute-force replay and retry attacks to impersonate or gain access to a legitimate user’s authenticated session

#### IDENTIFICATION & TRACKING

The authentication system must log the number, type and time of access requests

#### COUNTERMEASURE(S)

Rate-limiting services can be delivered via an API gateway to throttle access requests and detect potential malicious patterns like brute-force attacks. This should be done at network, application and user levels.

#### ASSURANCE

Build test harnesses that are designed to lock out users after multiple failed attempts with invalid credentials and tokens.





### **Case in Point: Brute Force**

**In September 2014, a large number of famous Apple iCloud users<sup>3</sup> were surprised to find intimate photos and other sensitive information leaked on the Internet. The culprit according to Apple was weak passwords. To help address the long list of issues with password security and brute-force guessing, Apple introduced an important rate-limiting backstop to prevent attackers from deploying scripts that gain unauthorized access.**

**Another example was Github's breach in November 2013. In this case, Github's repository was hit with access attempts from 40,000 different IP addresses<sup>4</sup>. These large-scale, brute-force attacks cannot be thwarted using passwords alone. Rate limiting, fingerprinting and other techniques must also be used to protect users and data.**

<sup>3</sup> <https://www.washingtonpost.com/news/the-switch/wp/2014/09/02/apples-basically-blaming-hack-victims-for-not-securing-their-own-icloud-accounts/>

<sup>4</sup> <http://www.theguardian.com/technology/2013/nov/21/github-accounts-compromised-in-brute-force-attack>

## 04 | The Curious Case of Injection

### INJECTION

#### CONTEXT

Injection is one of the most prevalent high-impact attack vectors. Injection vectors like SQL injection lead directly to most data breaches and serious security failures.

APIs are the gateway to the enterprise core, and they often let attacks slip right through their grasp. That is because the target of the injection attack is the back-end database (as in SQL injection) or a directory service (as in LDAP injection) or an ERP system or even an HVAC system. The attackers can leverage those resources directly and/or use them as launch pads to further attacks.

#### VULNERABILITIES

The root problem is a result of the mixture of the control plane and the data plane.

The results here are many, but the two most well-known are SQL injection and cross-site scripting.

#### IDENTIFICATION & TRACKING

The security architect must traverse the system to gather sources and sinks (where they are used and written). For the inputs of the system, trace them to their sinks and examine how they are handled. Is the data user controllable? Is it validated in the system?

For output data written back to the API caller, what is the source of that data? How is that data encoded?

#### COUNTERMEASURE(S)

Safe input/output handling is the key countermeasure.

For input handling, this is generally a mix of:

- Data sanitization and escaping that removes or overwrites control characters
- Input validation that examines a known good and/or known bad list of words, characters and data types and blocks access based on regular expression failures

For output handling, the system must encode the data in a way that prevents user-controllable from being propagated through the system in an executable format. This means encoding output data as JSON, HTML or other format, depending on how the client is set up to consume it.

#### ASSURANCE

This is an area where the security industry is particularly rich with testing tools. SQLMap is a powerful tool for testing your system for SQL injection<sup>5</sup>. Fuzzing tools like Burp Suite<sup>6</sup> can be instrumented with data from FuzzDB and other sources to identify SQL injection, cross-site scripting and other input/output-handling vulnerabilities.



## Case in Point: Injection

In October 2014, Drupal announced a SQL injection vulnerability<sup>7</sup> that allowed attackers to copy data and otherwise use the victim's Drupal site maliciously. The severity was grave, as attackers immediately began taking over Drupal servers.

The rapid reaction from the attacker community led to this dire warning from the Drupal team (emphasis added):

**“You should proceed under the assumption that every Drupal 7 website was compromised unless updated or patched before Oct 15th, 11pm UTC, that is 7 hours after the announcement.”**

The event illustrates how easy it can be for an attacker to exploit injection vectors once found; the extreme scale of those attacks; and the catastrophic, system-altering impact they can have on a company and its customers. There's no way to sugarcoat the threat of injection. Safe input/output handling is at least as important as access control and any other security service anywhere in the system, yet it is treated as almost entirely optional in many systems. Consider all input guilty until proven innocent by data sanitization and input validation, and make sure you have strict output encoding in place.

5 <http://sqlmap.org>

6 <https://portswigger.net/burp/>

7 <https://www.drupal.org/PSA-2014-003>

## 05 | The Curious Case of Lateral Movement

### LATERAL MOVEMENT

#### CONTEXT

Attackers are adept at manipulating security tokens and adapting them to find ways to move laterally across a system. Lateral movement is a powerful attack technique, because an externally accessible API server may be programmed by developers to talk to a database (let's call it database A). Attackers can leverage vulnerabilities to craft API requests that access databases B,C, and D, app servers E,F, and G, and enterprise message servers X, Y, and Z.

#### VULNERABILITIES

Direct object reference, cross-site request forgery and open redirects are three examples of vulnerabilities where lateral movement is used as a vector to bypass authorization.

#### IDENTIFICATION & TRACKING

This is a challenge for security architects, because the expected request will likely work as planned. But detecting lateral movement means that unexpected sequences and combinations must be attempted.

#### COUNTERMEASURE(S)

API gateways should enforce a strict token scoping and validation policy that limits what is accessible to each API caller. The API server should validate the inbound request against the token scope. The gateway should dispatch any request to a list of approved services.

#### ASSURANCE

Lateral movement is not an area where automated tools are particularly beneficial. Manual testing and fuzzing, along with penetration testing, is likely required for a security architect to sleep at night.





### **Case in Point: Lateral Movement**

**Target's credit-card data breach in the fall of 2013 was traced back to its HVAC system<sup>8</sup> — hardly on the security architect's short list of hot-button areas to tackle. This episode exposed the new kinds of dependencies introduced by expanding ecosystems of "things" like HVAC, printers, cameras and anything else with an IP address. Unfortunately, many are vulnerable and so have appealing entry points from which attackers can launch attacks. Don't let your API be one of them.**

<sup>8</sup> <http://krebsonsecurity.com/2014/02/target-hackers-broke-in-via-hvac-company/>

## 06 | The Curious Case of Session Promiscuity

### SESSION PROMISCUITY

#### CONTEXT

Historian Niall Ferguson posits, “Money is not metal. It is trust inscribed.” The same goes for session tokens including cookies, one-time-use URLs, SAML tokens and OAuth tokens. These session identifiers are the main (and often only) method for letting the API server know who is calling it. If these tokens are corrupted, replayed or spoofed, it’s difficult, if not impossible for API servers to distinguish valid access from malice.

#### VULNERABILITIES

Tokens may be tampered with or replayed and privileges may be altered

#### IDENTIFICATION & TRACKING

Review security token specification and implementation

#### COUNTERMEASURE(S)

Implement token-protection schemes that sign and hash tokens when they are issued. The API gateway must authenticate the signature and verify the hash to ensure the request is from an authorized source, and has not been tampered with.

Ensure the tokens are fresh, and use a one-time-use code (nonce) and/or verified timestamp.

#### ASSURANCE

Session identifiers are visible and they can be reviewed at design and dynamic-testing levels (proxy tools like Burp work well). A test suite should be developed to ensure that the tokens are tamper evident, resilient to replay, and only accepted from authorized servers.





### **Case in Point: Session Promiscuity**

**In 2008, Google's implementation of SAML-based Single Sign-On protocol opened up a hole that allowed malicious service providers to access Google user accounts<sup>9</sup>. A couple of things to note here: First, the problem was not with the SAML protocol, it was with Google's implementation. As the NSA likes to say, "We don't break standards, we break implementations." No matter what protocol you use, the standard is only partially relevant; you can build a weak system on a strong protocol.**

<sup>9</sup> <https://www.identityblog.com/?p=1011>

## 07 | The Curious Case of Invisible Attacker

INVISIBLE ATTACKER	
<b>CONTEXT</b>	
Information security has long relied on access-control technologies that are necessary, but not sufficient. Access control divides the system into known-good and known-bad states. These partitions are useful for defining and enforcing authorized access, but they do not hold up in all cases when deliberate malice is involved.	
<b>VULNERABILITIES</b>	<b>IDENTIFICATION &amp; TRACKING</b>
Attackers inject false messages into log files, find events that are not tracked, and/or tamper with log messages	Ensure a reliable event stream reports log messages to a central secure log server
<b>COUNTERMEASURE(S)</b>	
Network-only logging won't cut it; logging and monitoring must be done at an application level. Application sensors should be deployed at boundary-crossing layers like the API gateway. These sensors should record access, exception, malicious and related events.	
<b>ASSURANCE</b>	
Red team testing should be done to ensure that the logging systems do in fact detect malicious use.	





### **Case in Point: Invisible Attacker**

**In 2014, an employee of a regional medical center in Alabama was accused of unauthorized access and tampering with data from a data clearing house<sup>10</sup>. In cases like this, log files may be used for incident response as well as forensic evidence. In addition, the logging and monitoring services should limit the window of time attackers have to compromise a system.**

<sup>10</sup> <http://www.beckershospitalreview.com/healthcare-information-technology/dch-regional-medical-center-employee-allegedly-stole-data-from-computers.html>

## 08 | The Curious Case of Broken SSL/TLS

### BROKEN SSL/TLS

#### CONTEXT

There is no other security protocol as widely used as SSL/TLS, but this does not mean that it's always deployed correctly. In fact, checking your site against SSL Labs<sup>11</sup> is a wake-up call for many companies. Additionally, the client side has to build SSL/TLS protections that function correctly to avoid known vulnerabilities.

#### VULNERABILITIES

BEAST and Poodle are two recent examples of SSL/TLS weaknesses. In addition, certificate naming, chain validation, and protocol issues can open up man-in-the-middle, information-disclosure, and broken-authentication vulnerabilities.

#### IDENTIFICATION & TRACKING

Certificate authorities and key management systems can be used to shepherd, scale, and track SSL/TLS configurations.

#### COUNTERMEASURE(S)

SSL should be replaced with TLS. TLS should be upgraded to the highest level your organization can support (TLS 1.2). The provisioning, design, implementation and deployment should be carefully reviewed and tested. The API gateway can play a role as the central choke point for terminating and validating TLS traffic.

#### ASSURANCE

Excellent free testing tools exist like SSL Labs scanner, SSLyze, and most vulnerability-management scanners include checks for SSL/TLS weaknesses. However, most of these are on the server, so make sure to scan your API clients as well.





### Case in Point: Broken SSL/TLS

A study<sup>12</sup> done by University of Hannover in Germany found that around 8% of publicly available Android applications had broken SSL/TLS implementations. This included trusting all certificates and hostnames, trusting all certificate authorities, and using both encrypted and non-encrypted mixed-mode sessions.

11 <https://www.ssllabs.com>

12 <https://www.dfn-cert.de/dokumente/workshop/2013/FolienSmith.pdf>

## 09 | The Curious Case of Inversion of Control

### INVERSION OF CONTROL

#### CONTEXT

Unlike in the old days when they were strictly request-response client-server protocols, APIs are now a glue layer. With mobile, HTML5 and other technologies, we are seeing many applications where the server pushes data to the client.

Since most security protocols are set to trust servers and distrust clients, this turns the security protocols upside down as well.

#### VULNERABILITIES

Clients lack the protection and isolation of a DMZ

#### IDENTIFICATION & TRACKING

Build a set of allowed client redirects and sites that are authorized to connect

#### COUNTERMEASURE(S)

A full client-side DMZ is impractical. However, a client-side sandbox with server-side restrictions such as session management and control flow is possible. Clients should only accept pushes from authorized servers over strongly authenticated and encrypted channels.

#### ASSURANCE

Pentesting is generally a server-centric operation, but mobile and other push-based applications require a synthetic test suite to exercise applications end to end in order to identify client side issues.

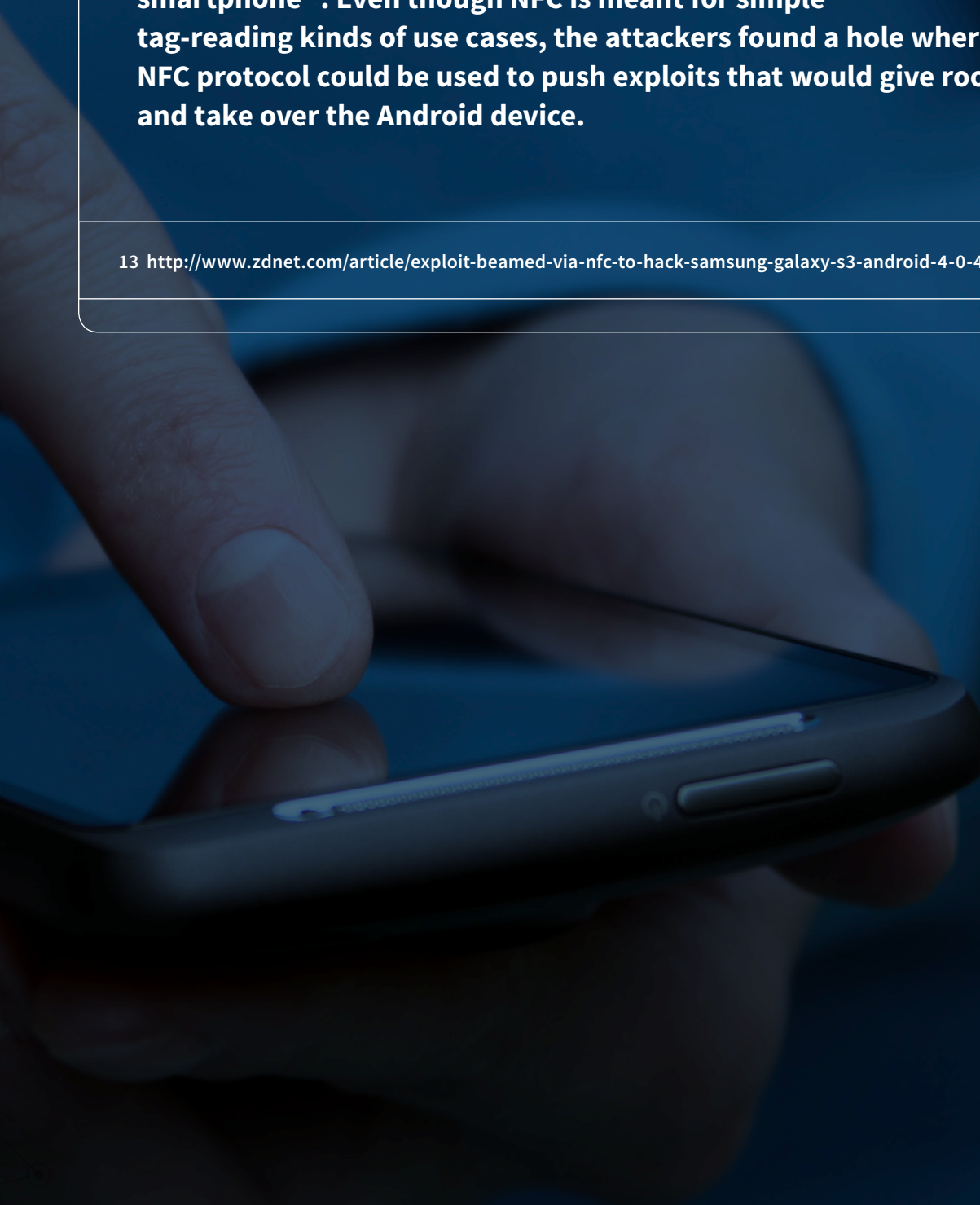




### **Case in Point: Inversion of Control**

**In 2012, a team from MWR Labs beamed exploits to an Android Samsung smartphone<sup>13</sup>. Even though NFC is meant for simple tag-reading kinds of use cases, the attackers found a hole whereby the NFC protocol could be used to push exploits that would give root access and take over the Android device.**

<sup>13</sup> <http://www.zdnet.com/article/exploit-beamed-via-nfc-to-hack-samsung-galaxy-s3-android-4-0-4/>



## 10 | The Curious Case of Order of Operations

ORDER OF OPERATIONS	
<b>CONTEXT</b>	
APIs can appear to be a static set of getters and setters, but once they are built into other applications, the combinations and permutations can drive unexpected behavior on the enterprise back end.	
<b>VULNERABILITIES</b>	<b>IDENTIFICATION &amp; TRACKING</b>
Old-school security folks may call this one Madame TOCTOU for time-of-check, time-of-use vulnerability or race conditions	Race conditions are notoriously difficult to identify. Manual synthetic testing under load is one of the better options here.
<b>COUNTERMEASURE(S)</b>	
Granular control on the server side for full-session state management is the main countermeasure.	
<b>ASSURANCE</b>	
TOCTOU vulnerabilities are among the hardest kind to detect because they require dynamic testing at loads to appear.	





### **Case in Point: Order of Operations**

**In 2014, the dominant Bitcoin marketplace, Mt. Gox, fell victim to a race condition<sup>14</sup>. Attackers found a vulnerability whereby they could force the Bitcoin marketplace to accept an altered transaction and block a legitimate transaction. The net result was that Mt. Gox was drained of Bitcoins and put out of business. (More detail on this fascinating case can be found at the link provided below.)**

<sup>14</sup> <http://www.tripwire.com/state-of-security/security-data-protection/cyber-security/the-fall-of-mount-gox/>

# 11 | The Curious Case of “Trusted ≠ Trustworthy”

TRUSTED ≠ TRUSTWORTHY	
CONTEXT	
<p>The security architect’s most dangerous suspect is probably not a malicious attacker, but rather him or herself.</p> <p>Microsoft’s John Lambert says it well, “Defenders think in lists, attackers think in graphs. As long as that is true, attackers win.”</p>	
VULNERABILITIES	IDENTIFICATION & TRACKING
Humans have cognitive biases including overconfidence, blind spots, and being susceptible to seductive details, data, and security-conference presentations.	Root out fuzzy concepts in your security architecture. Are you using the terms “trust” or “principle of least privilege?” What do these mean? Can you be more specific?
COUNTERMEASURE(S)	
Don’t rely on lists alone. Think of ways an API gateway and other security can make it more difficult for an attacker to access your enterprise’s resource graph.	
ASSURANCE	
Continually stress-test your assumptions and update your security architecture. It should be a living, breathing exercise — more like practicing a martial art than admiring a painting in a frame hung on a wall.	





### **Case in Point: “Trusted ≠ Trustworthy”**

**The U.S. Government’s Office of Personnel Management (OPM) breach<sup>15</sup> showed why the distinction between trust and trustworthiness is so important to information security. In the event, records containing sensitive personal information were breached. While very, very bad, this by itself was not so different from many other breaches.**

**What made OPM much worse is that it’s a system trusted by some of the most critical U.S. security organizations. Once that system was broken, a whole raft of other operations was impacted. The single biggest difference in a breach of this kind is that foundation-level dependency on a trusted system means that once breached, there is a cascade effect across the system.**

<sup>15</sup> <http://krebsonsecurity.com/tag/opm-breach/>



## **Don't Let the Trail Go Cold**

The curious case of API security will never be definitively closed, but that doesn't mean you need to let the trail go cold.

To keep the enterprise safe from API threats, security architects should:

- **Catalog and manage all APIs endpoints**
- **Ensure effective access controls**
- **Ensure the API implementation matches the architecture intent**
- **Monitor the system to detect and mitigate threats before they cause harm**

By taking a methodical approach to investigating and adjudicating API threats, and employing an API gateway to implement security capabilities, your security team can protect your business, your users and your data.



## About Gunnar Peterson

Gunnar Peterson, Acting Principal of Arctec Group, is a well known industry advisor focused on providing guidance — from a technology and vendor-independent perspective — to help improve business results. He blogs ([1raindrop.typepad.com](http://1raindrop.typepad.com)) and tweets (@oneraindrop) about distributed systems, security, and software that runs on them.

## About Axway

Axway (Euronext: AXW.PA) is a catalyst for transformation. With Axway AMPLIFY,™ our cloud-enabled data integration and engagement platform, leading brands better anticipate, adapt and scale to meet ever changing customer expectations. Our unified, API-first approach connects data from anywhere, fuels millions of apps and delivers real-time analytics to build customer experience networks that unite not only employees, but suppliers, partners and developers into an agile force for innovation that's as fast and fluid as today's digital customer. From idea to execution, we help make the future possible for more than 11,000 organizations in 100 countries.

**[www.axway.com](http://www.axway.com)**

**Follow us on Twitter:** <https://twitter.com/axway>

**Read more on our blogs:** <http://www.axway.com/en/blog>



**TRY IT NOW**

Free API Management Trial  
[www.axway.com/api-management-trial](http://www.axway.com/api-management-trial)

