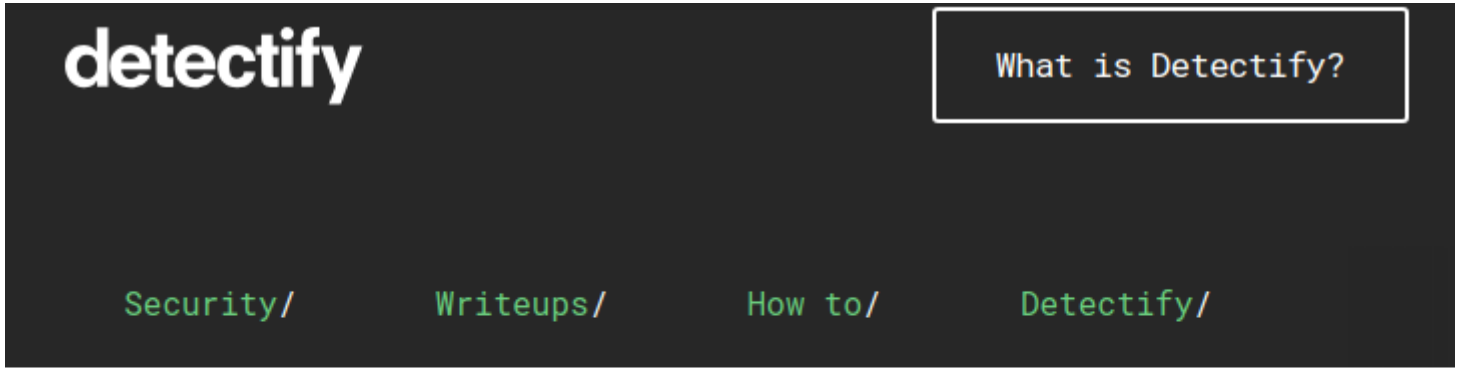
# MIME Types, XSS as a module and XSS as a standard

Hanno Böck
https://hboeck.de/
Twitter: @hanno




# XSS using quirky implementations of ACME http-01

September 4, 2018

# ACME http-01 validation

*http://example.org/.well-known/acme-challenge/TOKEN1*

**Response:** *TOKEN1.TOKEN2*

Some implementations reflect TOKEN1, thus this can lead to XSS.

But only if the browser interprets it as HTML.

*However, there is an old mod to Apache called Magic MIME that tries to figure out the content-type depending on the first bytes of the response. [...] For example <b> would lead to content type text/html [...]*

# Wait, what?!?

# Apache mod_mime_magic

It's a module that enables XSS attacks.

# Apache mod_mime_magic

*This module determines the MIME type of files in the same way the Unix file(1) command works: it looks at the first few bytes of the file.* (Apache documentation)

# mod_mime_magic parser

Parser code is based on an old fork of the "file" utility.

# What does that mean?

- If file extension is in /etc/mime.types use that.
- Else try to guess MIME type.

Any web application that allows uploading files with an unusual extension not in /etc/mime.types has Cross Site Scripting.

(Found multiple examples, disclosure pending.)

# Why?

Upload file containing HTML and Javascript.

Server will guess MIME type (e.g. if it starts with <html>) and send it as text/html.

# Can we disable mod_mime_magic?

Only globally, no option to disable it per host or directory (can't be disabled by customers on shared hosting).

# But if we disable mod_mime_magic we're good?

Not so fast...

# There's still the browser

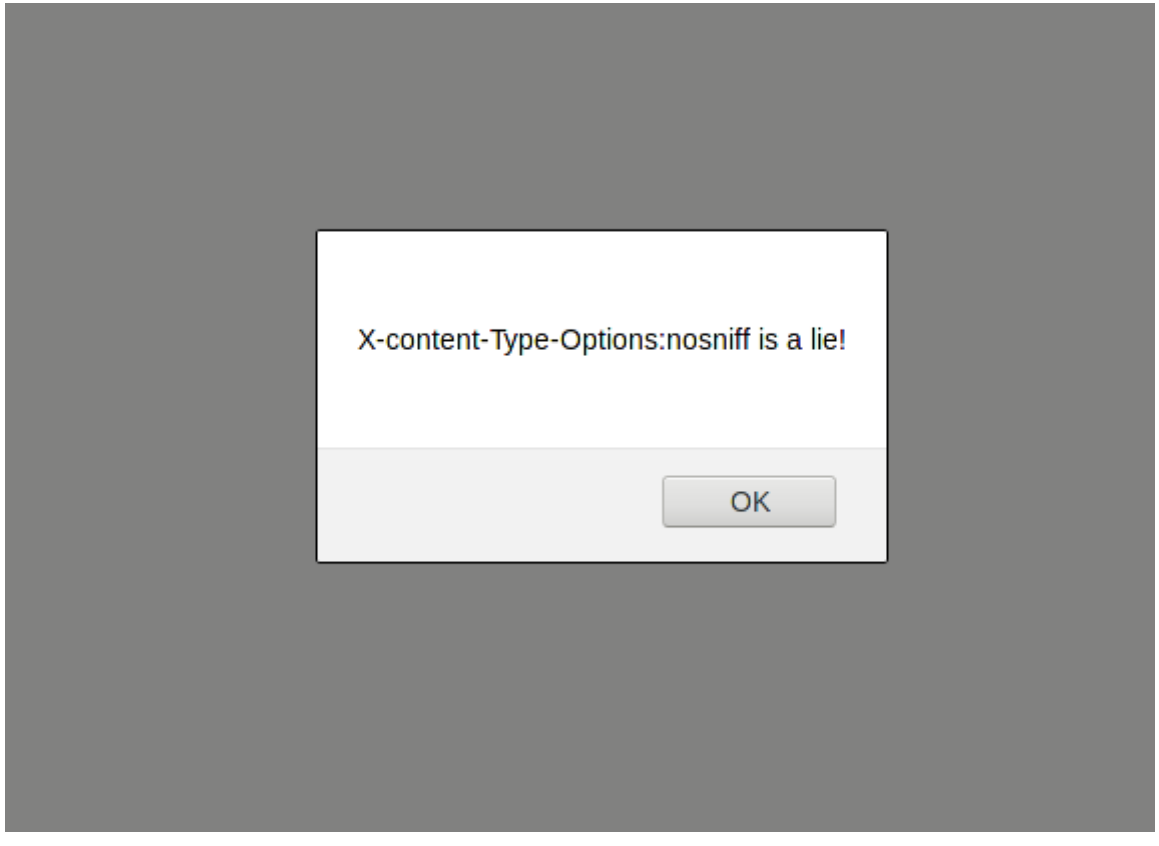
It can guess MIME types, too!

If file with HTML is sent without a MIME type the browser will render it.

# But there's

X-Content-Type-Options: nosniff

# So we can disable MIME sniffing in the browser?


X-content-Type-Options:nosniff is a lie!
OK

Firefox and Edge will render HTML without a MIME type even with "X-Content-Type-Options: nosniff".

# What can web applications do?

# Only allow file extensions that are in /etc/mime.types

Good luck with that: Every Linux distribution has its own version of mime.types.

# What could server administrators do?

# Always send a MIME type?

Let's set a safe MIME type (e.g. text/plain or application/octet-stream) for every unknown file extension.

# Apache "DefaultType" Directive

Has been removed in Apache 2.4.

# WHY???

# W3C Standard Authoritative Metadata

A standard to enable Cross Site Scripting.

# W3C Standard Authoritative Metadata

| | |
|---|---|
| Good Practice | Server software designers (implementers) SHOULD NOT specify default representation metadata, such as media type, character encoding, or content language, within the standard configuration shipped with the server. |

Instead of specifying a default for metadata, it is better for representations to be sent without that metadata. That allows the recipient to guess the metadata instead of being forced to either accept incorrect metadata or be tempted to violate Web architecture by ignoring it.

| | |
|---|---|
| Good Practice | Server managers (webmasters) SHOULD NOT specify an arbitrary Internet media type (e.g., "text/plain" or "application/octet-stream") when the media type is unknown. |

It is better to send no media type if the resource owner has failed to define one for a given representation.

# Software doesn't have to follow stupid standards

nginx sends application/octet-stream by default.

# Conclusions

# MIME sniffing - server and client side - can easily lead to XSS.

# Disable mod_mime_magic. It's inherently bad.

**Web application developers have no easy way of avoiding this issue.**

# X-Content-Type: nosniff doesn't help in half of the browsers.

# W3C standards tell us we aren't allowed to mitigate this server-side.

# This is a big mess

I'm hoping to get some ideas from you what to do about it.