

JAMES BENNETT · DJANGOCON EUROPE · 3<sup>RD</sup> JUNE 2015

THE NET IS DARK AND  
FULL OF TERRORS

# WHO I AM

Working with Django 9 years, 5 at Lawrence Journal-World

Commit bit since 2007

Involved in Django's release and security process ~8 years

# WHAT THIS IS

History of Django and security

How Django tries to protect you

Some ways we've screwed up and what you can learn from that

SECURITY IS HARD

IN THE BEGINNING...

16TH AUGUST 2006  
CVE-2007-0404

# DJANGO'S FIRST VULNERABILITY

## Small security hole fixed in translation helper utility

Posted by **Adrian Holovaty** on August 16, 2006

The Django team discovered and fixed a small security hole in the `django/bin/compile-messages.py` helper script, which is the script that compiles language translation message files (.po files) into binary format (.mo files).

The `compile-messages.py` script uses the name of the .po file to build arguments to a system command, and it didn't sufficiently validate the filename for potentially malicious content.

Users who relied on the language translation files provided with Django, or who wrote and compiled their own translations, were never at risk. Users who never ran the `compile-messages.py` script were never at risk. Only users who compiled third-party translations without examining the filenames first were potentially vulnerable.

No exploit based on this vulnerability, proof-of-concept or otherwise, is known to have existed.

Due to the nature of the vulnerability, we do not feel this merits a new release of Django. However, users who rely on third parties to supply translation files -- such as Django's own i18n maintainers -- are encouraged either to patch their code in one of these ways:

- Upgrade to the latest Django trunk (the Django development version).
- Simply overwrite your copy of `django/bin/compile-messages.py` with the new version. This file has not changed in any backwards-incompatible way since before Django version 0.90, so it's safe to copy over, regardless of which Django version you're using.
- We've applied the patches to Subversion "bug-fix" branches for both

# NINE YEARS AND 48 MORE SECURITY ISSUES LATER...



The web framework for  
perfectionists with deadlines.

OVERVIEW

DOWNLOAD

DOC

## Documentation

### Archive of security issues

Django's development team is strongly committed to responsible reporting and disclosure of security-related issues, as outlined in [Django's security policies](#).

As part of that commitment, we maintain the following historical list of issues which have been fixed and disclosed. For each issue, the list below includes the date, a brief description, the [CVE identifier](#) if applicable, a list of affected versions, a link to the full disclosure and links to the appropriate patch(es).

Some important caveats apply to this information:

- Lists of affected versions include only those versions of Django which had stable, security-supported releases at the time of disclosure. This means older versions (whose security support had expired) and versions which were in pre-release

20TH MAY 2015  
CVE-2015-3982

# DJANGO'S LATEST VULNERABILITY

## Security release issued: 1.8.2

Posted by **Tim Graham** on May 20, 2015

In accordance with [our security release policy](#), the Django team is issuing Django 1.8.2. This release is now available on PyPI and our [download page](#). This release addresses a security issue detailed below. We encourage all users of Django to upgrade as soon as possible. The Django master branch has also been updated.

### **CVE-2015-3982 - Fixed session flushing in the `cached_db` backend**

A change to **`session.flush()`** in the **`cached_db`** session backend in Django 1.8 mistakenly sets the session key to an empty string rather than **`None`**. An empty string is treated as a valid session key and the session cookie is set accordingly. Any users with an empty string in their session cookie will use the same session store. **`session.flush()`** is called by **`django.contrib.auth.logout()`** and, more seriously, by **`django.contrib.auth.login()`** when a user switches accounts. If a user is logged in and logs in again to a different account (without logging out) the session is flushed to avoid reuse. After the session is flushed (and its session key becomes `' '`) the account details are set on the session and the session is saved. Any users with an empty string in their session cookie will now be logged into that account.

Thanks to Sam Cooke for reporting the issue.

---

### **Affected versions**



Security issues are archived in the documentation:  
[https://docs.djangoproject.com/en/dev/  
releases/security/](https://docs.djangoproject.com/en/dev/releases/security/)

2007 · PRE-1.0

INFORMAL SECURITY PROCESS

2008 · DJANGO 1.0

TEMPLATE AUTOESCAPING

2010 · DJANGO 1.2

MODERN CSRF PROTECTION

2012 · DJANGO 1.4

HASHING, CRYPTO, SIGNED  
COOKIES, CLICKJACKING,  
SENSITIVE ERRORS, FORMAL  
SECURITY PROCESS

2013 · DJANGO 1.5 · DJANGO 1.6

HOST HEADER HARDENING,  
INCREASED HASH ITERATIONS,  
HASH TRUNCATION

2014 · DJANGO 1.7

SYSTEM CHECK FRAMEWORK

2015 · DJANGO 1.8

SECURITY MIDDLEWARE,  
DEPLOYMENT CHECK

# DJANGO'S SECURITY PROCESS

TL;DR: email [security@djangoproject.com](mailto:security@djangoproject.com) if you think you've found a security issue in Django.

Full security policy always accessible at  
<https://www.djangoproject.com/security/>

# SECURITY ISSUE VERIFICATION

Try out a proof-of-concept, if provided

Coordinate with reporter for more info if needed

Once verified, begin tracking issue (privately)

# SECURITY PATCHING PROCESS

Patches submitted in private tracker (only core team has access)

Reviewed and, if needed, ported to multiple versions of Django

# SECURITY NOTIFICATION PROCESS

Request a CVE identifier for the issue

One week prior to release, send to our security notification list

Pre-notification can be abbreviated for issues already public/exploited



# SECURITY RELEASE PROCESS

Patches merged from private security branches to public GitHub repository

Releases issued with blog post containing details of issue(s) and CVE identifier(s)

Spam ALL the social media sites!

The goals of our process are to ensure responsible reporting and disclosure of security issues.

# DJANGO vs. THE OWASP TOP TEN

[https://www.owasp.org/index.php/  
Category:OWASP\\_Top\\_Ten\\_Project](https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project)

Injection attacks ✓

Authentication and session management ✓

Cross-site scripting (XSS) ✓

Direct object references ✓

Misconfiguration ✓

Sensitive data exposure ✓

Function level access control ✓

Cross-site request forgery (CSRF) ✓

Components with known vulnerabilities ✓

Unvalidated redirects and forwards ✓

ABOVE AND BEYOND

Django tries very hard to be secure-by-default, and to offer the tools you need to harden your applications beyond the common cases.



However...

**oh god how did this get here**



**I am not good with computers**

WHY DO WE FALL?

WHY DO WE ~~FALL~~?

WHY DO WE **FAIL**?

“Parsing the Accept-Language header is expensive to do every time, let’s do it once per unique value and cache the results!”

–THE DJANGO TEAM, CIRCA 2007

“Let’s use a one-time base36 token to do password resets!”

–THE DJANGO TEAM, CIRCA 2010

“Formsets need to dynamically grow the number of forms they use!”

–THE DJANGO TEAM, CIRCA 2013



“Restrictions on password length are dumb! Long passwords are better!”

–THE DJANGO TEAM, CIRCA 2013

**CVE-2007-5712**

Denial-of-service via arbitrarily-large  
Accept-Language header

**CVE-2010-4535**

Denial-of-service in password-reset mechanism

**CVE-2013-0306**

Denial-of-service via formset max\_num bypass

**CVE-2013-1443**

Denial-of-service via large passwords

**oh god how did this get here**



**I am not good with computers**

**CVE-2007-5712**

Denial-of-service via arbitrarily-large  
Accept-Language header

**CVE-2010-4535**

Denial-of-service in password-reset mechanism

**CVE-2013-0306**

Denial-of-service via formset max\_num bypass

**CVE-2013-1443**

Denial-of-service via large passwords

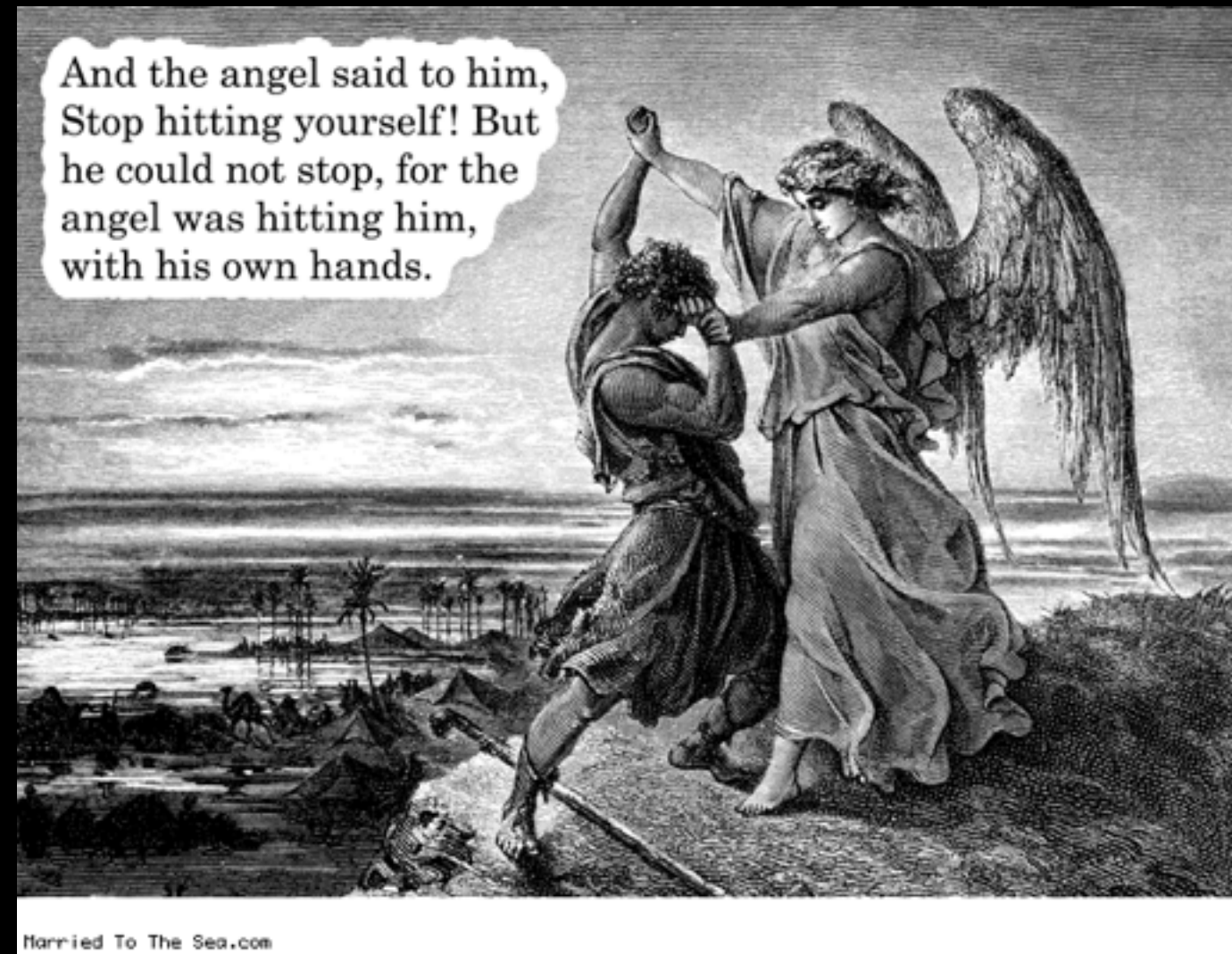


# PYTHON IS GREAT

No buffer overflows

But you can still DoS yourself if you're not careful

We learned this the hard way so you shouldn't have to



# STOP DOS'ING YOURSELF

Sanity-check your inputs for length *before* you start processing them

Yes, even passwords (when appropriate)!

Configure your web server to cap the length of HTTP headers and request bodies

“URLField should really check whether the URL exists before accepting the value!”

–THE DJANGO TEAM, CIRCA 2006

“URLField should accept anything that matches  
the format of a valid URL!”

–THE DJANGO TEAM, CIRCA 2006



“EmailField should accept anything that matches the format of a valid email address!”

–THE DJANGO TEAM, CIRCA 2006

“Checking for corrupt image files is easy, we can just use PIL’s routines for that!”

–THE DJANGO TEAM, CIRCA 2012

“Most image formats store metadata in a header,  
let’s find it by only reading a few bytes at a time!”

–THE DJANGO TEAM, CIRCA 2012

**CVE-2011-4137**

Denial-of-service via `URLField.verify_exists`

**CVE-2009-3965**

Denial-of-service via pathological regular-expression performance

**CVE-2012-3443**

Denial-of-service via compressed image files

**CVE-2012-3444**

Denial-of-service via large image files

**oh god how did this get here**



**I am not good with computers**

**CVE-2011-4137**

Denial-of-service via `URLField.verify_exists`

**CVE-2009-3965**

Denial-of-service via pathological regular-expression performance

**CVE-2012-3443**

Denial-of-service via compressed image files

**CVE-2012-3444**

Denial-of-service via large image files



# THE BIG O

Expresses upper bound on  
your algorithm

Also, apparently, an anime

But more important is the  
“upper bound” bit



“What’s the worst that could happen?”

–ACTUALLY A VERY USEFUL QUESTION



NO REALLY, STOP DOS'ING YOURSELF!

Figure out how much work your code should do

Then figure out whether you can make it do more

Then figure out ways to ensure it does less

Some issues (compressed formats, incremental reads, pathological regex, etc.) have been around forever — read up on them!

[illegible]

“Values of cookies we’ve set can be trusted!”

–THE DJANGO TEAM, CIRCA 2010

“Admin users can be trusted with a bit of the  
lookup API!”

–THE DJANGO TEAM, CIRCA 2010

“We can trust the browser same-origin sandbox!”

–THE DJANGO TEAM, CIRCA 2011

“We can trust admin users with the history log!”

–THE DJANGO TEAM, CIRCA 2013

“Once we’ve validated a value and stored it, we  
can trust it!”

–THE DJANGO TEAM, CIRCA 2013

**CVE-2010-3082**

XSS via trusting unsafe cookie value

**CVE-2010-4534**

Information leakage in administrative interface

**CVE-2011-0696**

CSRF via forged HTTP headers

**CVE-2013-0305**

Information leakage via admin history log

**NO CVE, DISCLOSED 2013-08-13**

XSS via admin trusting URLField values



“We can trust the HTTP Host header now!”

–THE DJANGO TEAM, OVER AND OVER AGAIN...

**CVE-2011-4139**

Host header cache poisoning

**CVE-2011-4140**

Potential CSRF via Host header

**CVE-2012-4520**

Host header poisoning

**ADVISORY, 2012-12-10**

Additional hardening of Host header handling

**ADVISORY, 2013-02-19**

*Additional* hardening of Host header handling

"I did warn you not to trust me."





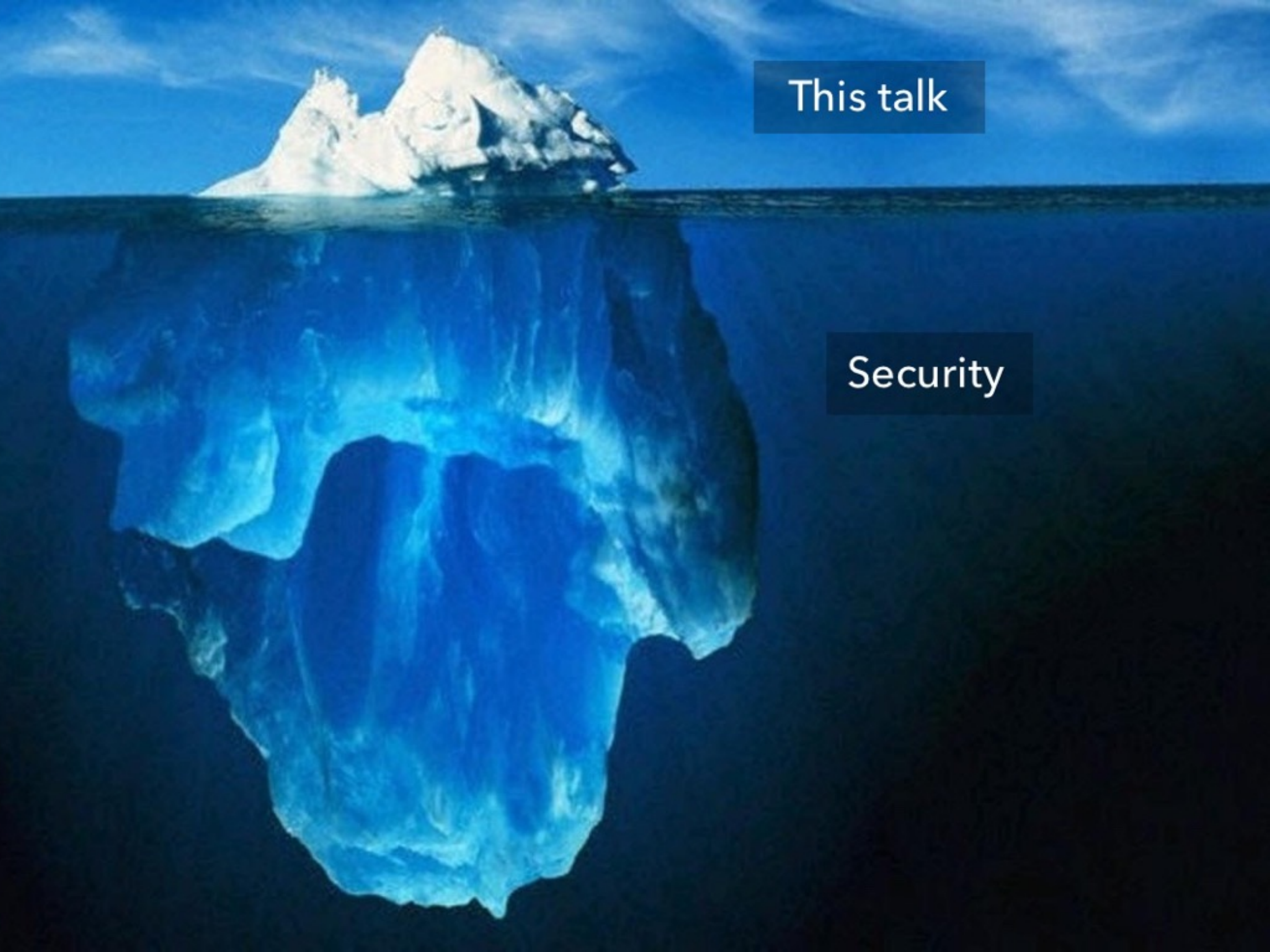
The image features a dark, atmospheric landscape. The sky is filled with heavy, layered clouds in shades of deep blue and grey. In the foreground, the dark silhouette of a hill or mountain slope is visible on the right side, extending towards the bottom of the frame. The overall mood is somber and mysterious.

TRUST NO ONE

THERE IS NO SUCH  
THING AS "SECURE"

WHY DO WE FALL?



A photograph of an iceberg floating in the ocean. The tip of the iceberg is visible above the water line, while the much larger, submerged part is visible below. The sky is blue with some clouds. The water is dark blue. The iceberg is white and jagged.

This talk

Security





# QUOTES/IMAGES

<http://www.marriedtothesea.com/index.php?date=012710>

[http://en.wikipedia.org/wiki/File:Selfmade\\_Big\\_O.png](http://en.wikipedia.org/wiki/File:Selfmade_Big_O.png)

<http://www.ex-parrot.com/pdw/Mail-RFC822-Address.html>

<http://highlighthollywood.com/2015/02/game-of-thrones-actor-aidan-gillen-lord-petyr-baelish-talks-season-5-sansa-and-little-finger-highlight-hollywood-news/>

[http://x-files.wikia.com/wiki/File:Trust\\_No\\_One\\_tagline.jpg](http://x-files.wikia.com/wiki/File:Trust_No_One_tagline.jpg)

<http://www.slideshare.net/ChristofHammel/process-iceberg-21703547>