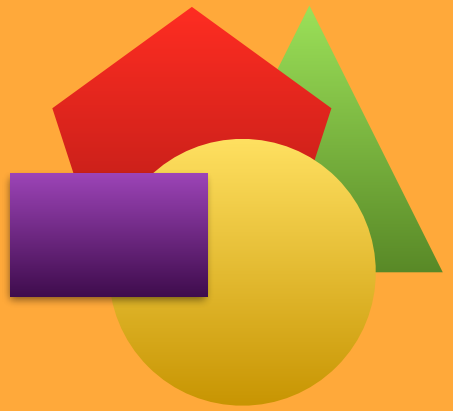
OWASP - Top10 2013 - A4: Insecure Direct Object References (IDOR)

Hochschule Mannheim
SSE, WS2016
Prof. Dr. Sachar Paulus
Jan-Philipp Willem, 1314162

Agenda

- Beschreibung
- Hands-On!
- How to fix it?
- Kategorisierung



„Unsichere direkte Objektreferenzen“

Beschreibung

„A *direct object reference* occurs when a developer exposes a reference to an internal implementation object, such as a *file, directory, database record*, or key, as *a URL or form parameter*. An attacker can manipulate direct object references to access other objects without authorization, unless an access control check is in place."

https://www.owasp.org/index.php?title=Top_10_2007-Insecure_Direct_Object_Reference&oldid=81713

„unsicherer direkter Objektzugriff?“ (1/2)

- hier: Objekt ≠ „Instanz einer Klasse“
  - *Resource*
- Zugriff *ohne Mapping* von Parameter zu:
  - Datei
  - Verzeichnis
  - DB-Record, ORM-/DM-Object

ORM: Object-Relational-Mapper
DM: (noSQL-) Document-Mapper

„unsicherer direkter Objektzugriff?“ (2/2)

- Probieren von Parameter-Kombinationen
- Eventueller Zugriff auf nicht korrekt abgesicherte *Resources*, ..
- .. welche *eigentlich* für den aktuellen User verborgen bleiben sollten.

Hands-On!

Architektur

- Docker
- Ruby on Rails
- *REST(-full URLs)*
- Scaffold-App:
  - Posts
  - Dokumente-Download

Sourcecode

```
$ git clone https://github.com/jwillem/
owasp_t10-2013_a4-idor_ruby.git
```

How to fix it?

„sicherer indirekter Zugriff?“ (1/2)

- Mapping von Parameter zu *Resource*
  - *Whitelist*
  - Integer / Direkt / Hash
  - fest / pro User / zeitbasiert

```
{ 4 => ‘umsatz_2013.pdf’,
  ‘umsatz_2014.pdf’ =>
  ‘umsatz_2014.pdf’,
  ‘js2dek§h8$’ =>
  ’umsatz_2015.pdf’
}
```

- defaults

```
{ wenn User mit Id nicht gefunden..
  dann liefere Session-User }
```

„sicherer indirekter Zugriff?“ (2/2)

- *sanitize*-function / Regex auf Parameter anwenden
- Bei *jedem* Zugriff sollte eine Rechte- oder Rollenprüfung vorgenommen werden.
- Eventuell bietet dies schon das genutzte Framework von Hause aus?

Kategorisierung

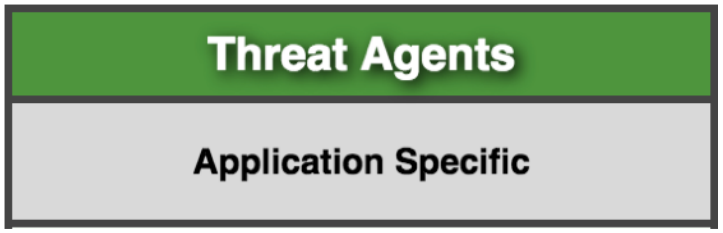
OWASP Risk Rating Methodology

https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology

Threat Agents	Attack Vectors	Weakness Prevalence	Weakness Detectability	Technical Impacts	Business Impacts
App Specific	Easy	Widespread	Easy	Severe	App / Business Specific
	Average	Common	Average	Moderate	
	Difficult	Uncommon	Difficult	Minor	

Risks

„Das individuelle Gesamtrisiko kann nur durch die Betrachtung aller relevanten Faktoren abgeschätzt werden."



Bedrohungsquellen

- *Anwendungsspezifisch*
- Gibt es verschiedene Arten an Benutzern?
- Gibt es Daten die nur gewisse Benutzer sehen sollten?

Angriffsvektoren

- *Ausnutzbarkeit*: einfach
- Zugriff auf nicht autorisierte Daten durch Parameterwechsel

Schwachstellen (1/2)

- *Verbreitung*: häufig
  - gerade bei REST kann URL-Schema leicht verstanden werden
  - *automatisierte* Suche nach passenden Kombinationen: trivial

Schwachstellen (2/2)

- *Auffindbarkeit*: einfach
  - Ob ein geeigneter Zugriffsschutz genutzt wird, ist schnell zu erkennen
  - *Code-Analysis*?
  - ORM/DM mit Rollenschutz pro Zugriff

Technische Auswirkungen

- *Auswirkung*: mittel
  - alle Daten, welche per Parameter Empfangen werden können sind gefährdet

Geschäftliche Auswirkungen

- *Anwendungs-/Geschäftsspezifisch*
  - Welcher Wert besteht hinter den ungeschützten Daten? **individual Business-Value**
  - Was sind die Auswirkungen für die Anwendung, nach Veröffentlichung der Schwachstelle?

Danke für die Aufmerksamkeit.

– Fragen?