



RequestRodeo: Client Side Protection against Session Riding

Martin Johns / Justus Winter
University of Hamburg, SVS
johns@informatik.uni-hamburg.de
0049-40-42883-2510

OWASP
AppSec
Europe
May 2006

Copyright © 2006 - The OWASP Foundation
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License.

The OWASP Foundation
<http://www.owasp.org/>

Me, Myself and I

- Martin Johns
- johns at informatik.uni-hamburg.de
- Security researcher at the University of Hamburg
- Member of the secologic project
 - ▶ Research project carried out by SAP, Commerzbank, Eurosec and the University of Hamburg
 - ▶ Sponsored by the German Ministry of Economics (BMWi)
 - ▶ Goal: Improving software security
 - ▶ Visit us at <http://www.secologic.org>



Agenda

- **Web Application Authentication**
- **Session Riding**
- **Client Side Protection**
- **Conclusion**



Agenda

- **Web Application Authentication**
- Session Riding
- Client Side Protection
- Conclusion



Agenda

■ **Web Application Authentication**

- ▶ **Explicit authentication**
- ▶ Implicit authentication

■ Session Riding

■ Client Side Protection

■ Conclusion



Explicit Authentication

- The authentication credentials are communicated by the web application
 - ▶ URL rewriting: Session token is included in every URL
 - ▶ Form based session tokens
- Immune against Session Riding



Agenda

■ **Web Application Authentication**

- ▶ Explicit authentication
- ▶ **Implicit authentication**

■ Session Riding

■ Client Side Protection

■ Conclusion



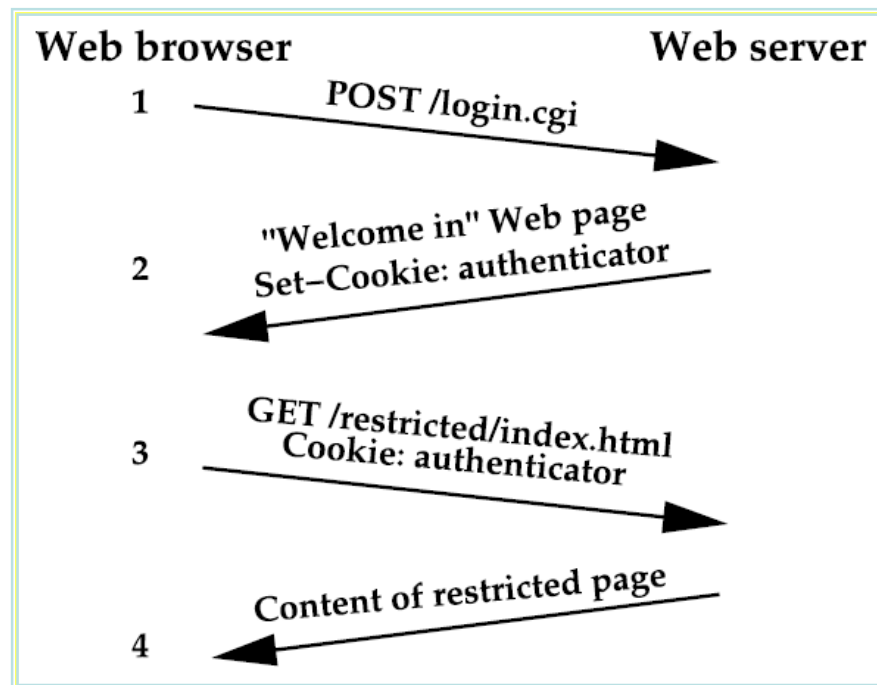
Implicit Authentication

- Automatically executed by the browser
 - ▶ **Cookies**
 - ▶ **http authentication (Basic, Digest, NTLM)**
 - ▶ **IP based schemes**
 - ▶ **Client side SSL**
- Potentially vulnerable against Session Riding

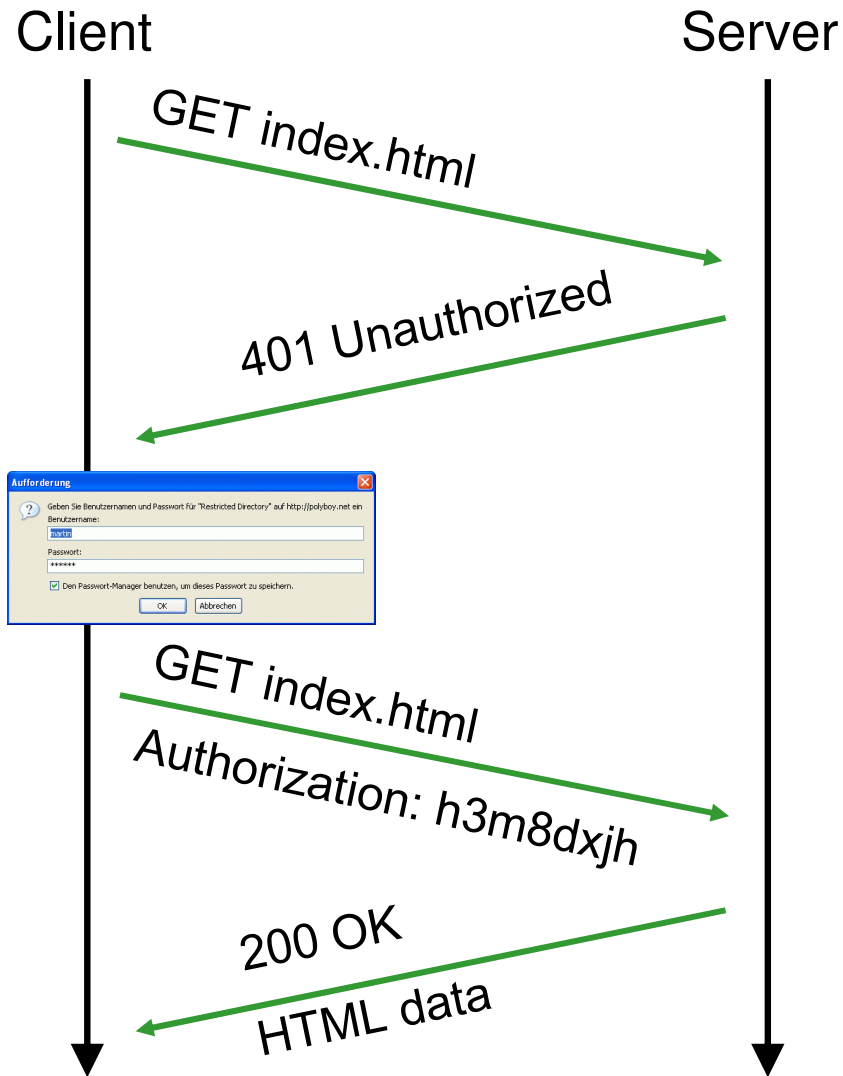


Session management with cookies

- After the authentication form the server sets a cookie at the client's browser
- As long as this cookie is valid, the client's requests are treated as authorized



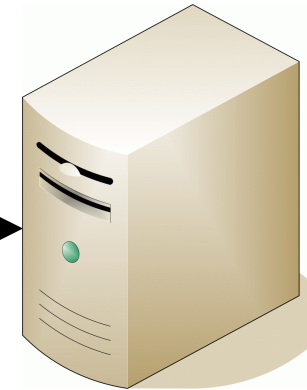
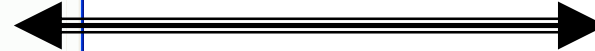
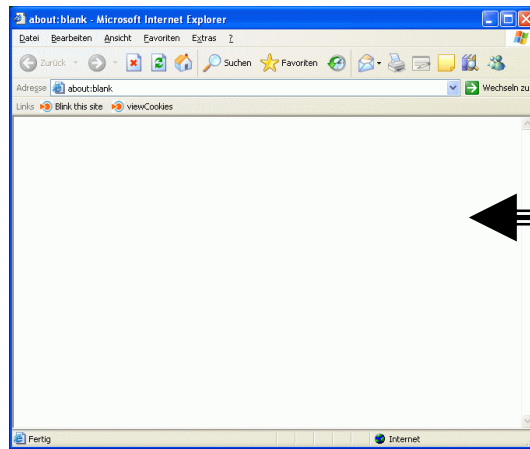
http authentication (Basic, Digest)



- ▶ The client requests a restricted resource
- ▶ The server answers with a "401 Unauthorized" response
- ▶ This causes the client's browser to demand the credentials
- ▶ The client resends the request
- ▶ The user's credentials are included via the "Authorization" header
- ▶ Every further request to that authentication realm contains the credentials automatically

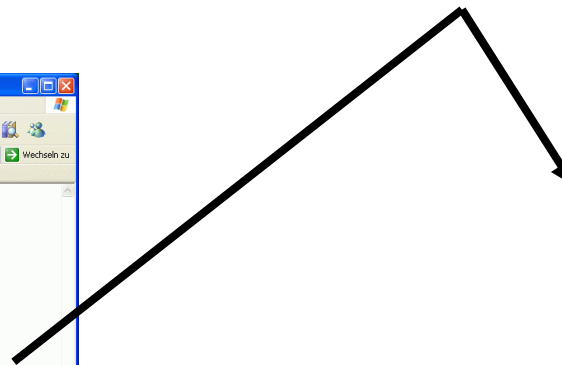
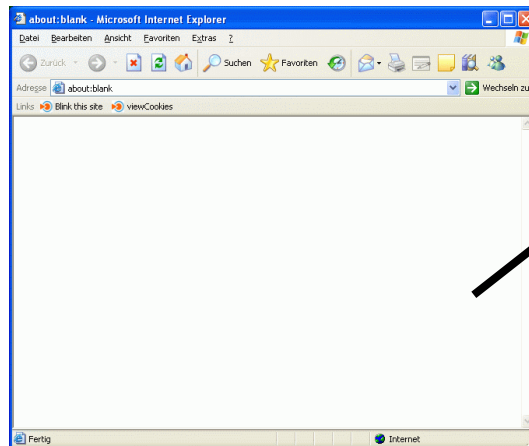


IP based authentication



Intranet webserver

Firewall



Agenda

- Web Application Authentication
- **Session Riding**
- Client Side Protection
- Conclusion



Agenda

- Web Application Authentication
- **Session Riding**
 - ▶ **The attack**
 - ▶ Server side countermeasures
- Client Side Protection
- Conclusion



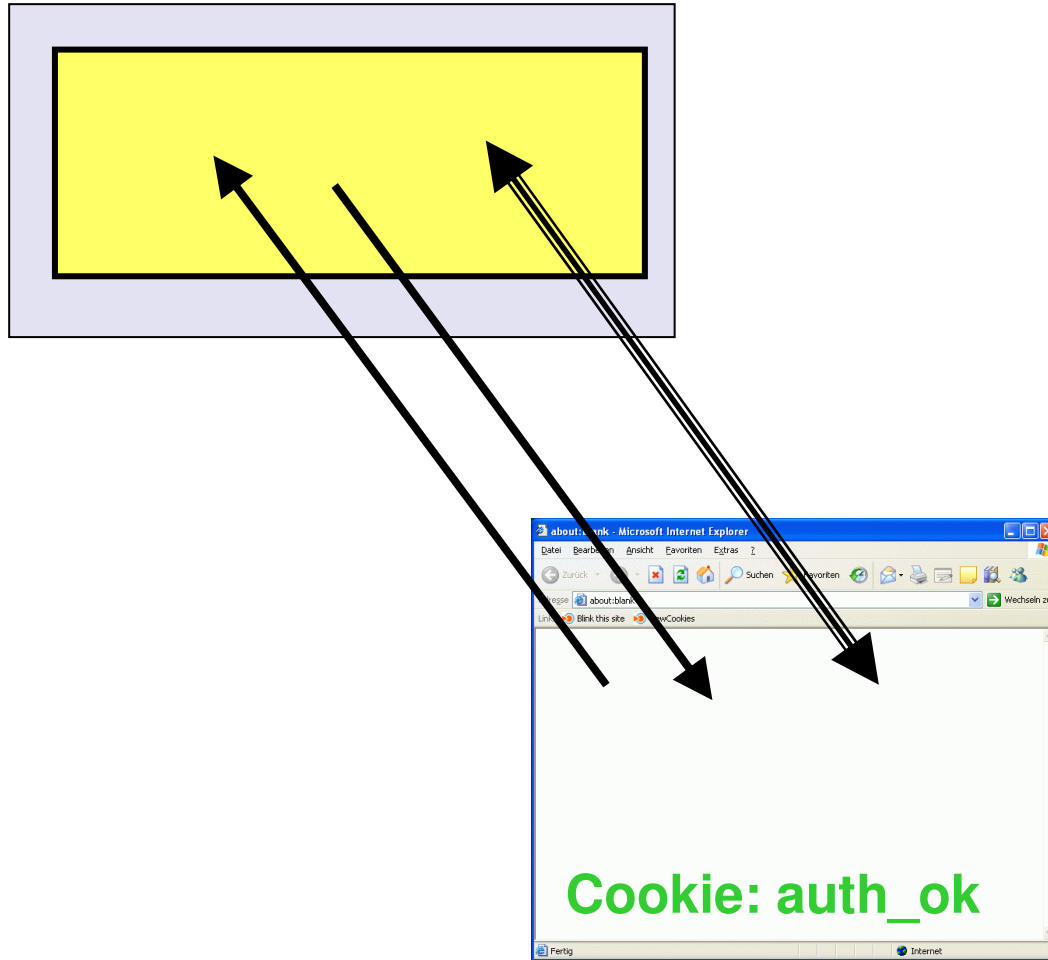
Session Riding

- Exploits implicit authentication mechanisms
 - ▶ Known since 2001
 - ▶ A.k.a “Cross Site Request Forgery” (XSRF / CSRF)
 - ▶ Unknown/underestimated attack vector (compared to XSS or SQL injection)
- The Attack:
 - ▶ The attacker creates a hidden http request inside the victim’s web browser
 - ▶ This request is executed in the victim’s authentication context



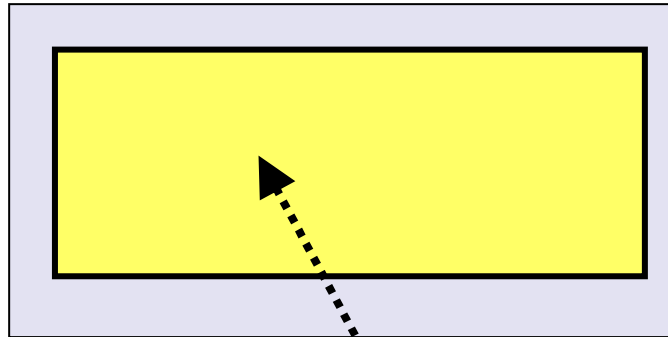
Session Riding (II)

www.bank.com

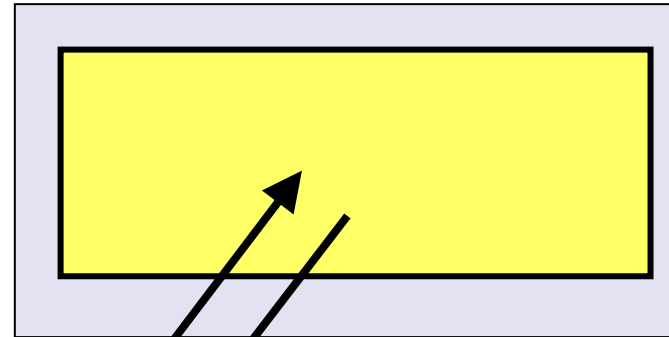


Session Riding (II)

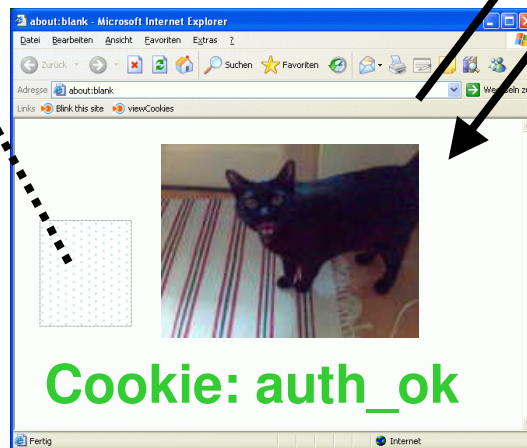
www.bank.com



www.attacker.org



GET transfer.cgi?am=10000&an=3422421



Session Riding (II)

- Problem: The web application does not verify that state changing request were created “within” the web application
- Methods
 - ▶ Image (GET)
 - ▶ Frame (POST)
- Realworld exploits
 - ▶ Deletion / fabrication of data
 - ▶ State alteration of network devices
 - ▶ JavaScript code inclusion
 - ▶ Execution of arbitrary PHP-code



Session Riding: Attack origin

- Reflected: Attacker has to manufacture a website that hosts the origin of the hidden request
- Local / stored: Origin of the malicious http request is hosted on the attacked website
 - ▶ Example: Users are allowed to post images with foreign URLs



Session Riding: Classification

■ Attack origin

- ▶ Reflected: Attacker has to manufacture a website that hosts the origin of the hidden request
- ▶ Stored: Origin of the malicious http request is hosted on the attacked website
 - Example: Users are allowed to post images with foreign URLs

■ Target

- ▶ Broad: Every user of a certain web application is potentially affected by the same request
- ▶ Explicit: The attack is tailored for one special victim



Misconceptions

- Only accepting POST requests
 - ▶ Defends against local attacks
 - ▶ On foreign web pages hidden POST requests can be created with frames
- Referrer checking
 - ▶ Some users prohibit referrers
 - referrerless requests have to be accepted
 - ▶ Techniques to selectively create http request without referrers exist



Agenda

- Web Application Authentication
- **Session Riding**
 - ▶ The attack
 - ▶ **Server side countermeasures**
- Client Side Protection
- Conclusion



Server Side Countermeasures

■ Misconceptions (don't)

- ▶ Only accepting POST requests
- ▶ Referrer checking

■ Do:

- ▶ Use one-time tokens for state changing requests
- ▶ Mirror all foreign content / do not allow users to use arbitrary URLs
- ▶ Alternative: Switch to explicit authentication



Agenda

- Web Application Authentication
- Session Riding
- **Client Side Protection**
- Conclusion



Agenda

- Web Application Authentication
- Session Riding
- **Client Side Protection**
 - ▶ **Concept**
 - ▶ Identification of fraudulent requests
 - ▶ Removal of implicit authentication
- Conclusion



RequestRodeo: Concept

- Client Side Proxy
- Identification of potential fraudulent requests
- Removal of implicit authentication



Identification of suspicious requests

- The origin determines the state
- Definition: **entitled**

An http request is classified as **entitled** only if:

- It was initiated because of the interaction with a web page (i.e. clicking on a link, submitting a form or through JavaScript) and
- the URLs of the originating page and the requested page satisfy the “same origin policy”

- Only entitled requests are permitted to carry implicit authentication information.



Agenda

- Web Application Authentication
- Session Riding
- **Client Side Protection**
 - ▶ Concept
 - ▶ **Identification of fraudulent requests**
 - ▶ Removal of implicit authentication
- Conclusion



Processing http responses

■ Adding tokens to URLs

- ▶ HTML code in http responses is processed
- ▶ Identification of elements that potentially causes http requests
- ▶ The target URLs of these elements are enhanced with a URL token
- ▶ The tokens are kept together with the repose's URL in a table

→ This way the proxy is able determine a request's origin

■ *"a reliable referrer"*



Processing http responses (II)

■ Example:

▶ ``

→

``

■ Augmenting JavaScript:

▶ Impossible to find all URLs in JS

▶ Instead: Altering the code that creates the requests:

`document.location = url;`

→

`document.location = __rrt_addToken(url);`

▶ Unreliable, therefore additional referrer checks



Checking http requests

- Every http request is checked for a URL token
- If such a token exists, the originating URL is retrieved
- If the domains of the request and its origin do not match, implicit authentication information gets removed



Agenda

- Web Application Authentication
- Session Riding
- **Client Side Protection**
 - ▶ Concept
 - ▶ Identification of fraudulent requests
 - ▶ **Removal of implicit authentication**
- Conclusion



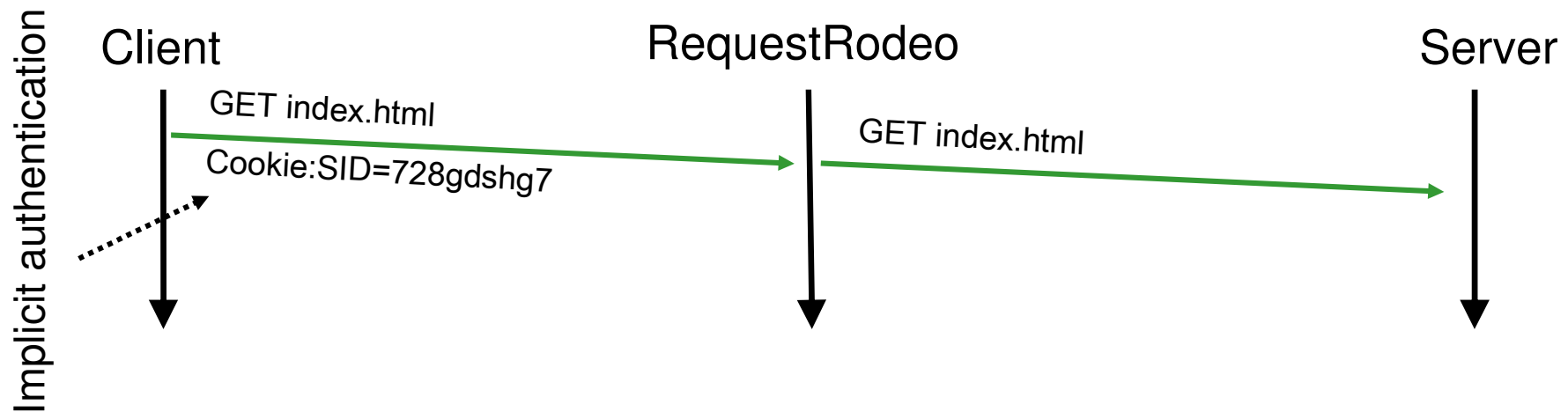
Removal of implicit authentication: Cookies

- Cookies are communicated via the http header field "Cookie"
- If a not "entitled" request contains such a field, it is removed by the proxy before forwarding the request



Removal of implicit authentication: Cookies

- Cookies are communicated via the http header field "Cookie"
- If a not "entitled" request contains such a field, it is removed by the proxy before forwarding the request



Removal of implicit authentication: Cookies

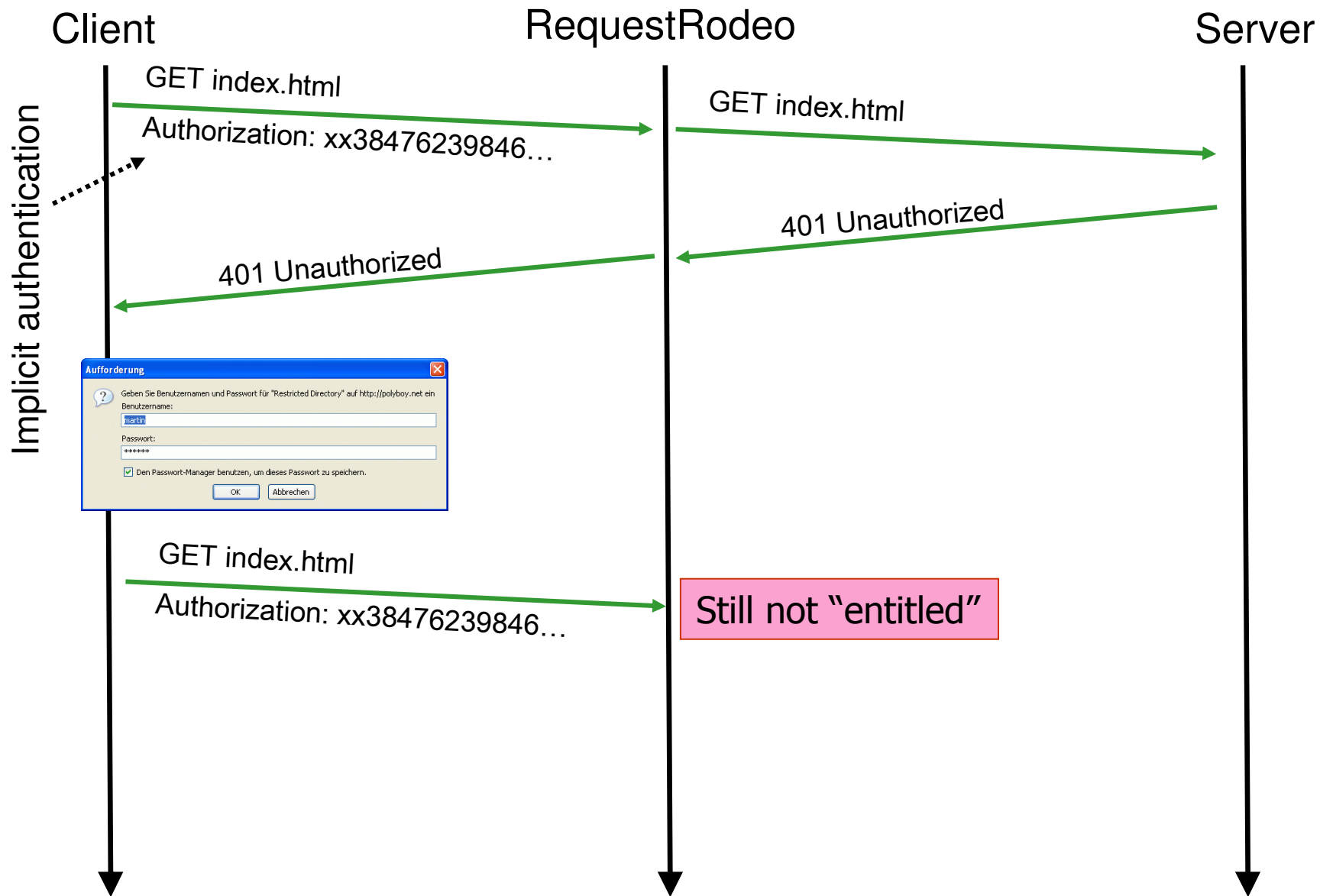
- Cookies are communicated via the http header field "Cookie"
- If a not "entitled" request contains such a field, it is removed by the proxy before forwarding the request
- A cookie's domain value should be respected
- Example:
 - ▶ cookie's domain value: "foo.org"
 - ▶ a request from "www.foo.org" to "order.foo.org" is allowed to carry the cookie



Removal of implicit authentication: http auth

- Simply removing unentitled “Authorization” headers does not work

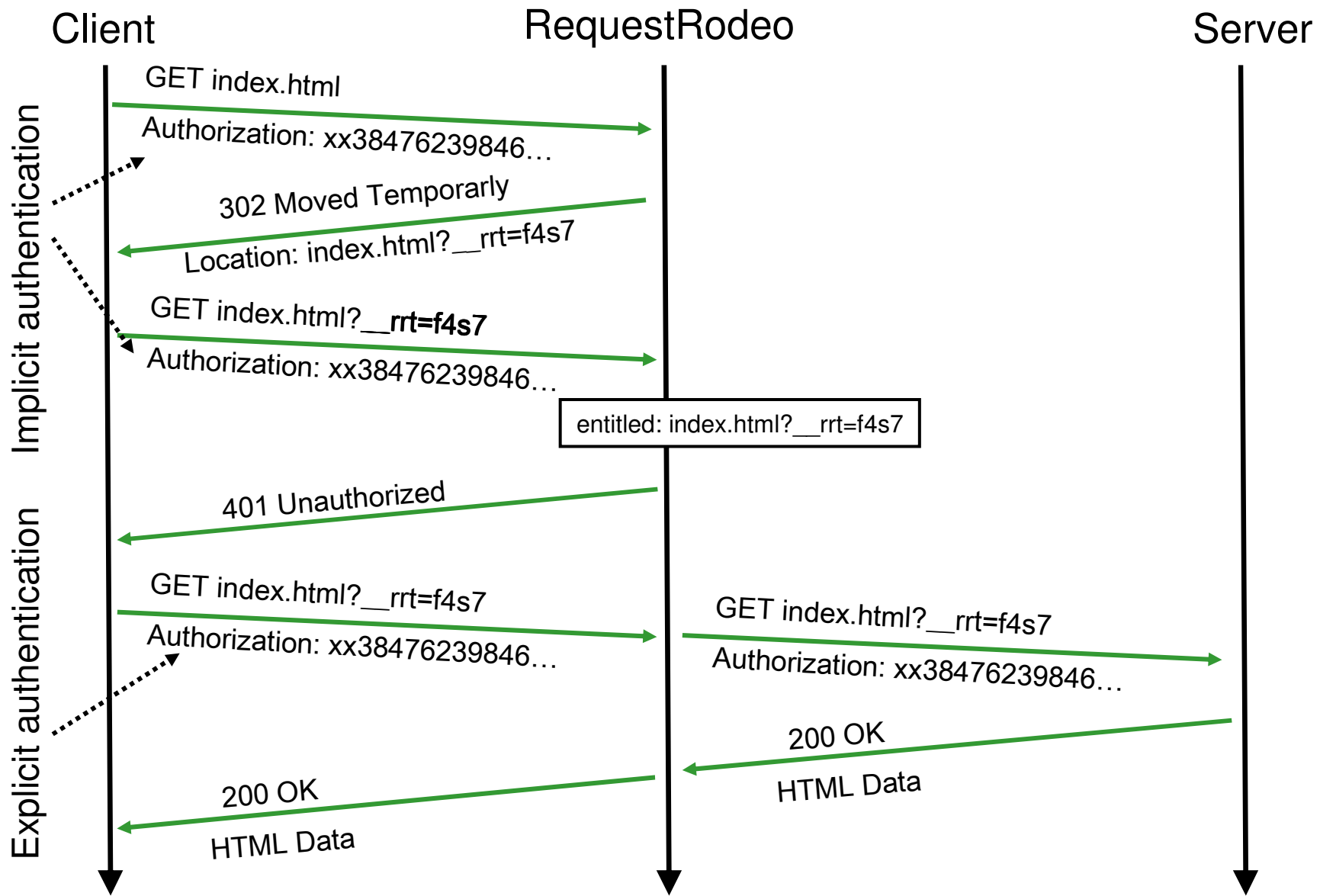




Removal of implicit authentication: http auth

- Simply removing unentitled "Authorization" headers does not work
- In this case the browser re-requests the same URL → The URL is still not entitled
- The proxy has to add a token to the request's URL
- This is done using a 302 "Temporary moved" response



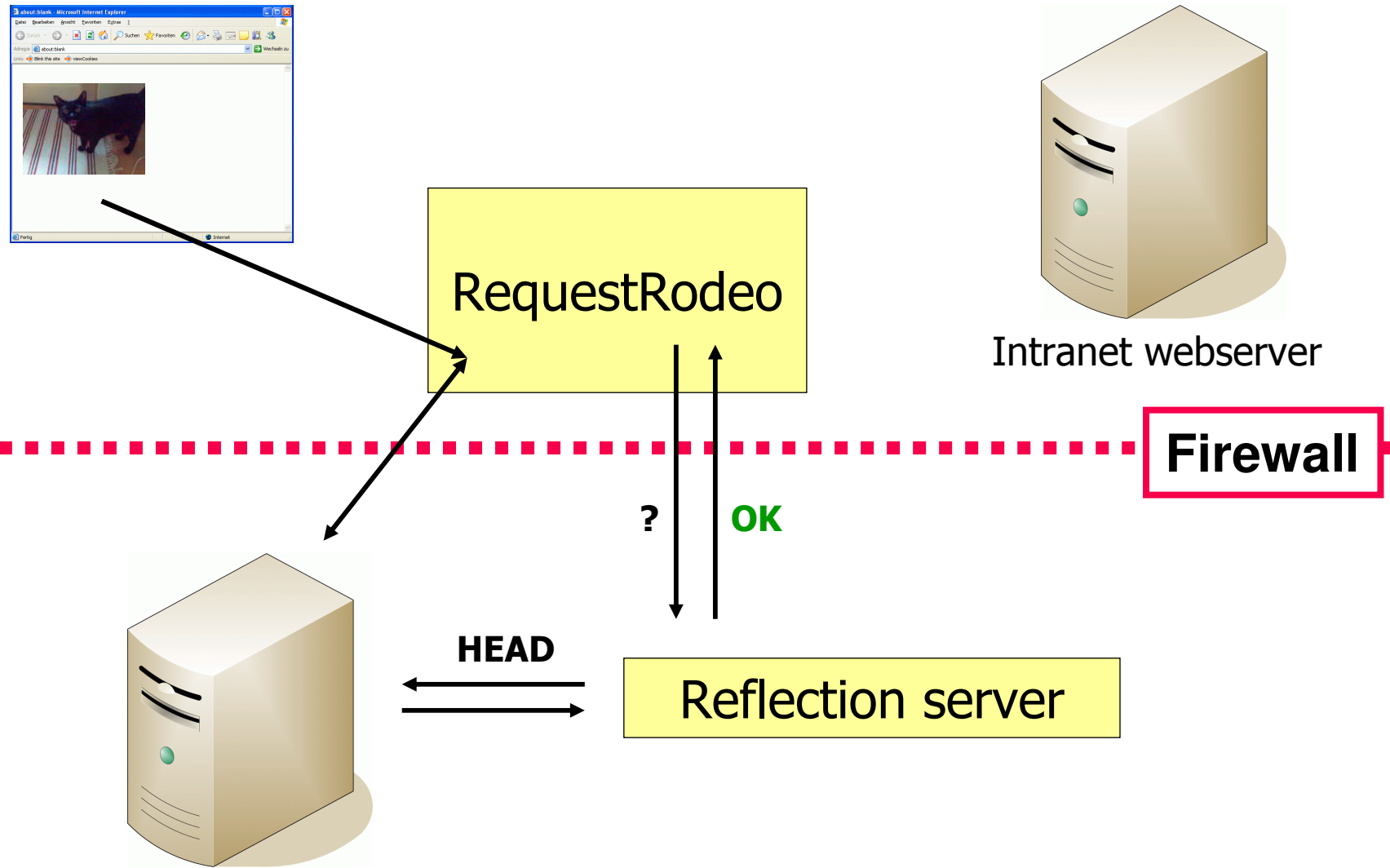


IP-based authentication

- Unentitled requests are only allowed if their target is worldreachable
- For this reason a “reflection sever” is employed
- The reflection server is hosted offsite (i.e. on the other side of the cooperate firewall)
- Before allowing an unentitled request, the proxy verifies that the reflection server is able to access the request’s target
- Caching of checked IPs to minimize the performance penalty



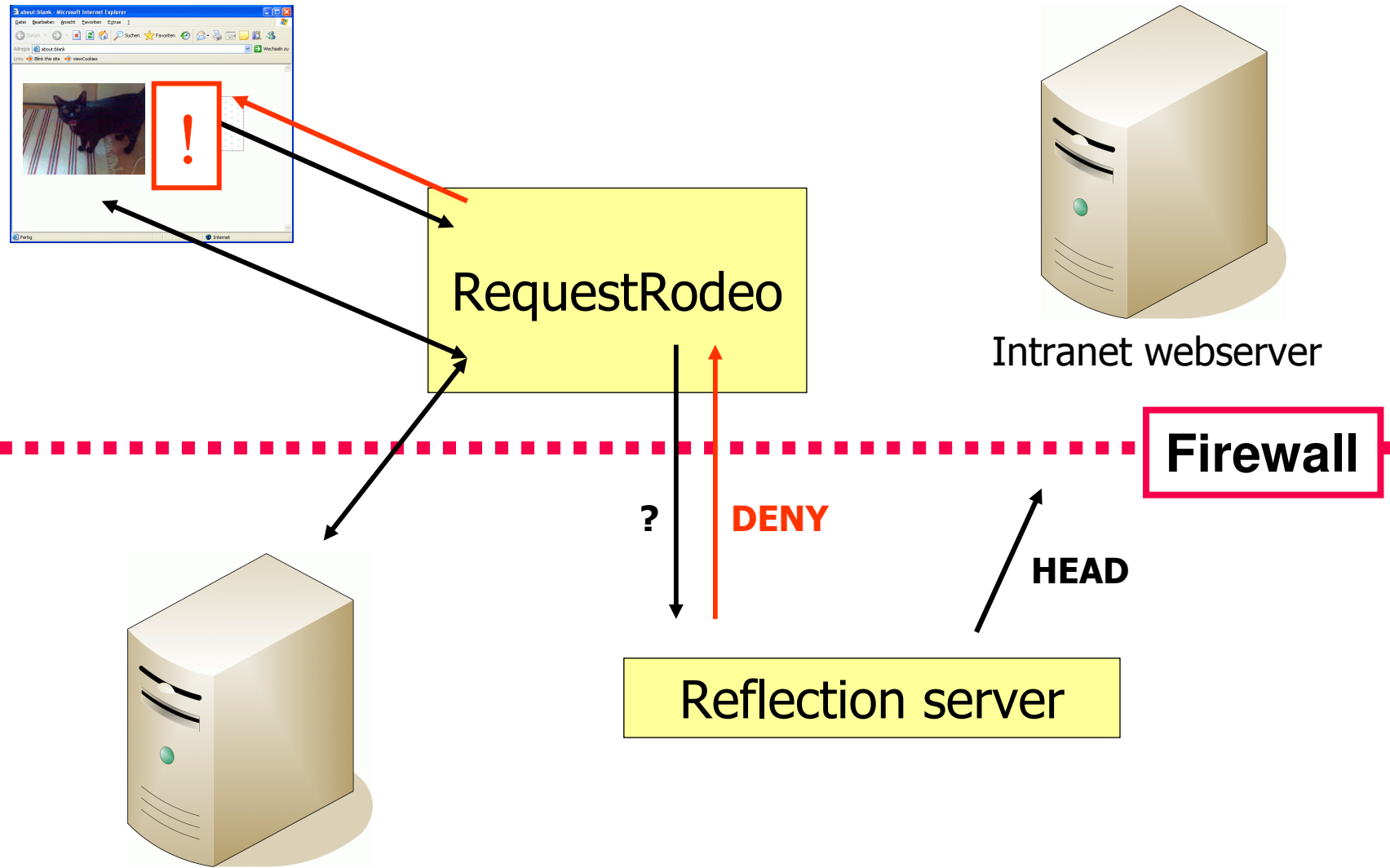
IP based authentication



Malicious site



IP based authentication



Malicious site



Agenda

- Web Application Authentication
- Session Riding
- Client Side Protection
- **Conclusion**



Implementation

- Proof of concept
- Implemented in Python
- Using the “Twisted” framework
- (will be) released under the GPL



Discussion

■ Conservative Approach:

- ▶ Connections are allowed
- ▶ Only removal of implicit authentication
- ▶ Request to public resources are uninhibited

■ Limitations:

- ▶ No protection against “local” attacks
- ▶ NTLM / Client Side SSL not implemented

■ Future work

- ▶ Browser integration
- ▶ Addition of anti XSS techniques



The end

- Thank you for your attention
- Questions?
- Comments?

