Brandenburg
University of Technology
Cottbus

PADIOFIRE

Firewalls für das WEB 2.0

Hartmut König, René Rietz

PADIOFIRE

Firewall für das Web 2.0

Brandenburg
University of Technology
Cottbus

I. Ziele des Vorhabens

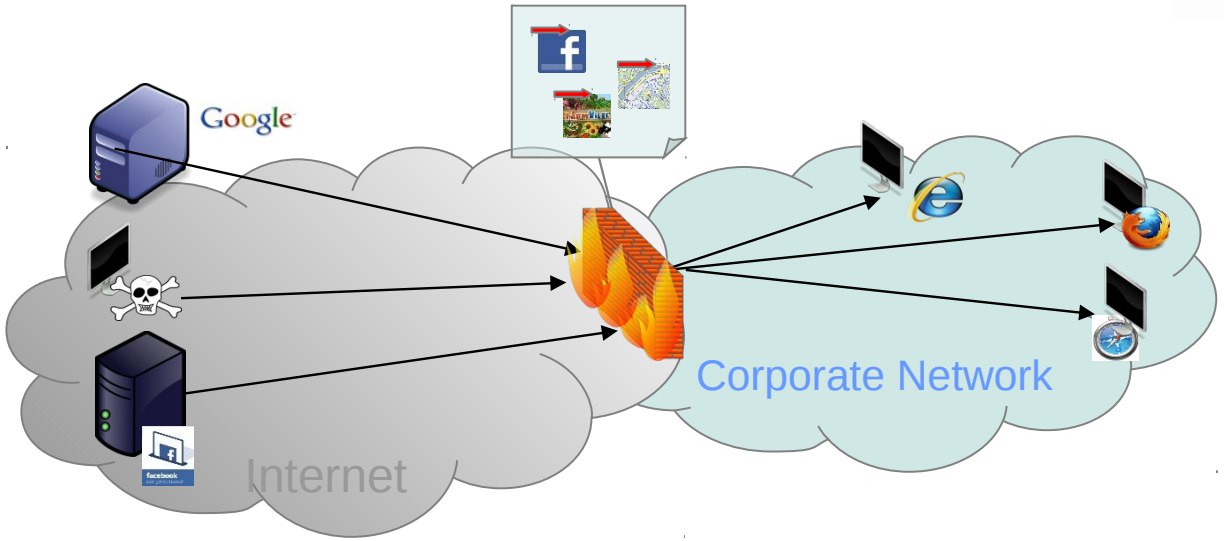
II. Stand der Teilprojekte

Brandenburg
University of Technology
Cottbus

I.

Ziele des Vorhabens

Problemstellung



- **Problem klassischer Firewalls:**
  - Webverkehr oft generell erlaubt:

```
ACCEPT tcp -- anywhere anywhere state RELATED,ESTABLISHED

ACCEPT tcp -- anywhere anywhere tcp:dpt http
```

  - Application Level Gateways können aktive Inhalte wie JavaScript erkennen und entfernen
    → mit Verlust von Funktionalität verbunden und nicht immer zuverlässig

Meldungen vom 14./15.5.2012

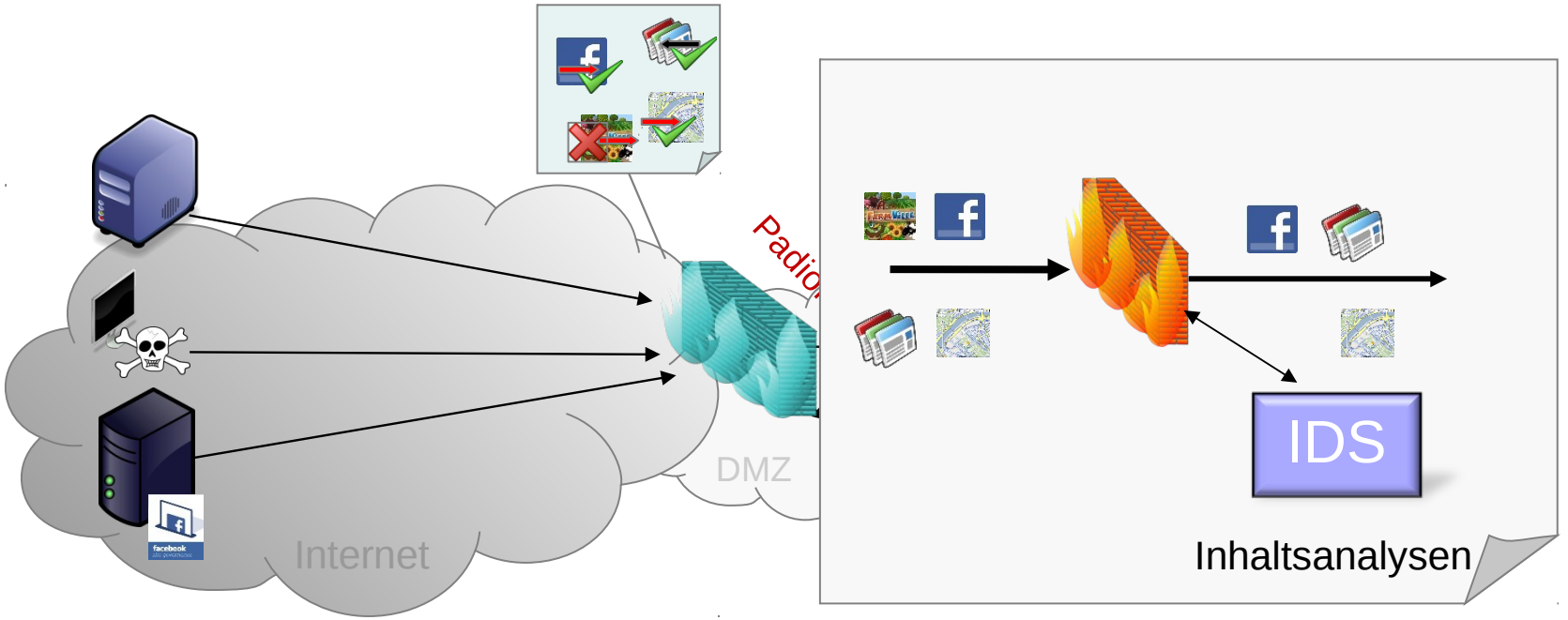
Brandenburg University of Technology Cottbus

Quelle (Screenshots): http://www.heise.de

PADIOFIRE

Parallized Application Detection in Overlay(s / - Networks) for Firewalls

Brandenburg University of Technology Cottbus



Grundidee: Kopplung von Firewalls mit Intrusion Detection Methoden für die Analyse der Inhalte der Web-Protokolle

☞ *Identifizierung* und *selektive Weiterleitung von Webanwendungen* (z.B. Google Maps) *innerhalb* von Webprotokollen (HTTP, SPDY, Websockets)

Projektpartner PADIOFIRE

Brandenburg University of Technology Cottbus

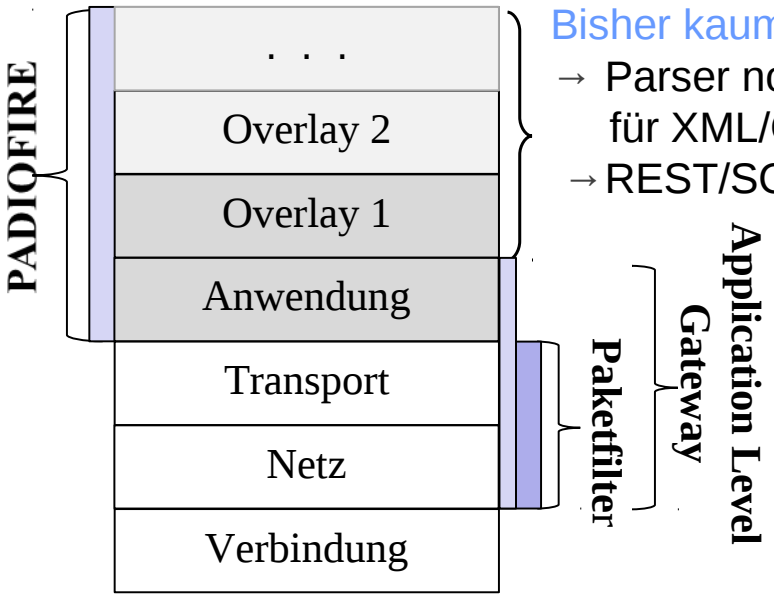
- genua mbH Kirchheim b. München
  Hochsichere (zertifizierte) Firewallsysteme
- BTU Cottbus, LS Rechnernetze und Kommunikationssysteme
  Intrusion Detection Systeme (IDS), Host-, Netz-, Verteilte Einbruchs-erkennung, Signaturgenerierung, statische Programmanalyse
- FAU Erlangen LS 1 für Informatik (I1)
  IT-Forensik, IT-Sicherheit, Penetration Testing

Assoziierte Partner

- Universität Innsbruck, LS Technische Informatik (CCS)
  Netzmonitoring, Verkehrsklassifizierung, Angriffserkennung, Anomalieerkennung
- IXIA
  Leistungsmessung von Netzwerkkomponenten/Sicherheitslösungen

Ziele

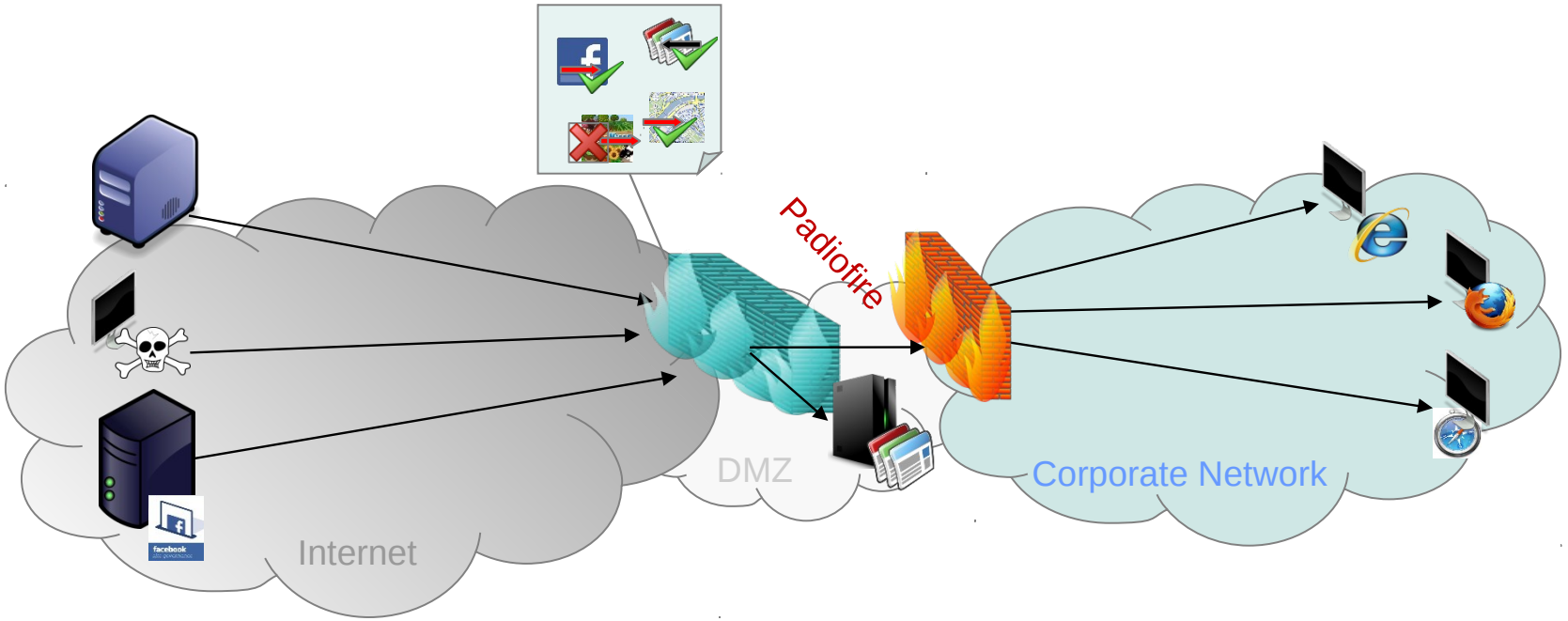
- Entwicklung einer Web-Application-Firewall
  - Overlayanalyse
  - Parser für genaue Analysen von Webinhalten
    → XML, SVG, CSS, JavaScript
  - Ableitung eines Modells der Webanwendung
    → Prüfen, ob aktiver Code noch dem ursprünglichem Code entspricht
    → Bewertung der Gefährlichkeit der aktiven Inhalte
- Demonstration der Funktionalität an ausgewählten Web-Anwendungen



Bisher kaum adressiert:
→ Parser notwendig für XML/CSS/JS
→ REST/SOAP/AJAX

Web 2.0 Protokollstack

Lösungsansatz (1)

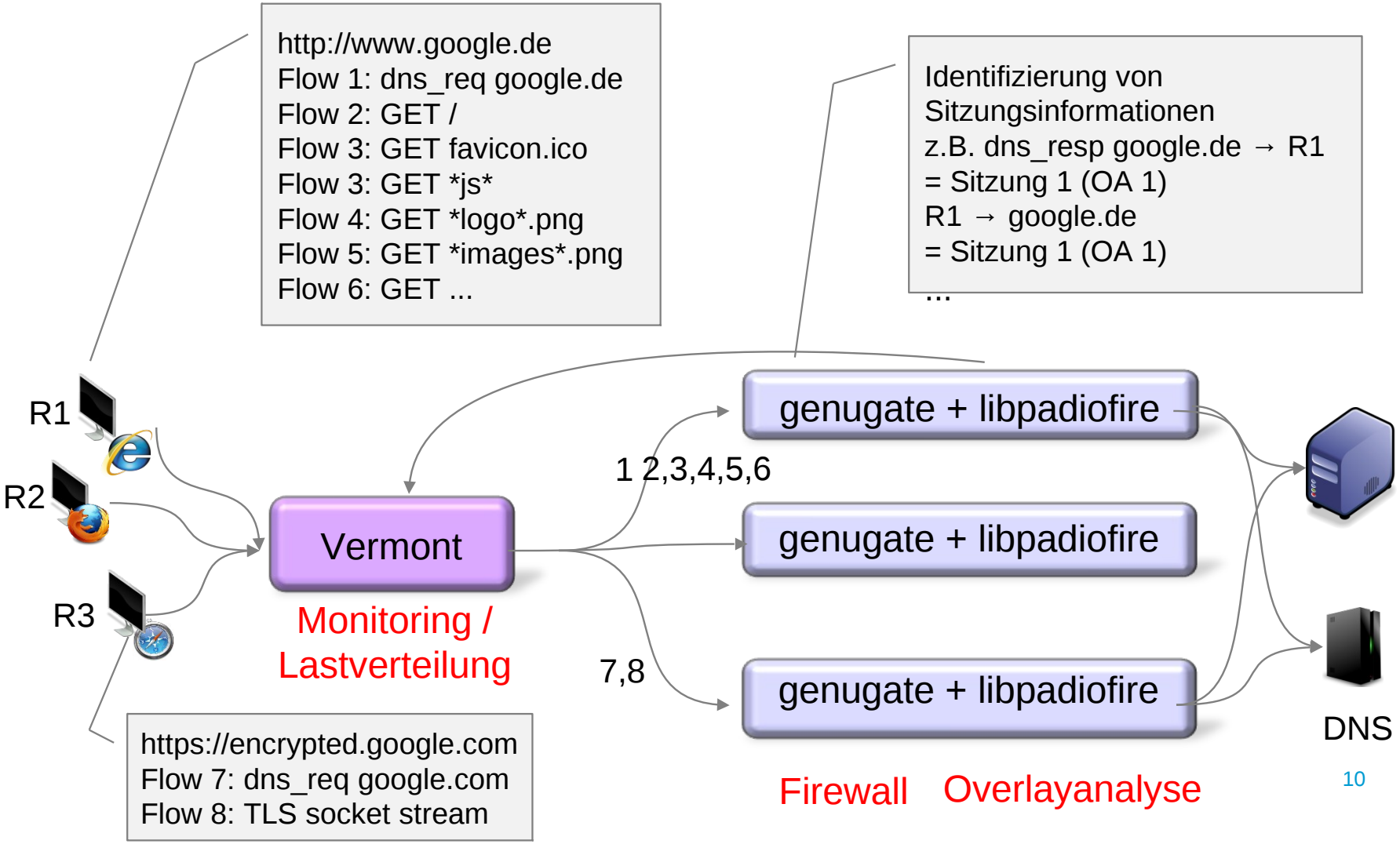


Padiofire Komponenten:

- Lastverteilungskomponente (*Vermont*)
- Firewall/ALG (*genugate*)
- Overlayanalyse (*libpadiofire*)

Lösungsansatz (2)

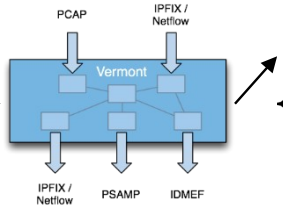
Brandenburg University of Technology Cottbus



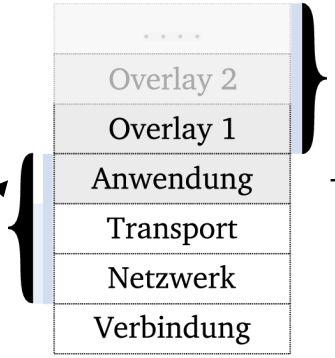
Teilprojekte

Brandenburg University of Technology Cottbus

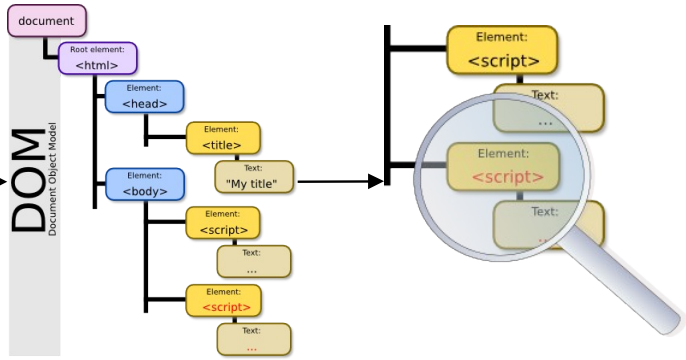
Lasterzeugung, Einspielen protokollierter HTTP-Sitzungen
→ Leistungsmessungen



Flow-basierte Anwendungserkennung, Lastverteilung, Aggregation



Strombasierte HTTP-Analyse + Schnittstelle zur Overlay-analyse



Modellgenerierung für Web-Applikation (HTML/CSS/JS) / Signaturanalyse

Einschätzung der Code-Bösartigkeit (Malware-, Bug-, Manipulationserkennung)

Eingangsklassifizierung von WebApps
(HTTP-Analyse)

Tiefenanalyse → libpadiofire
(Overlayanalyse)

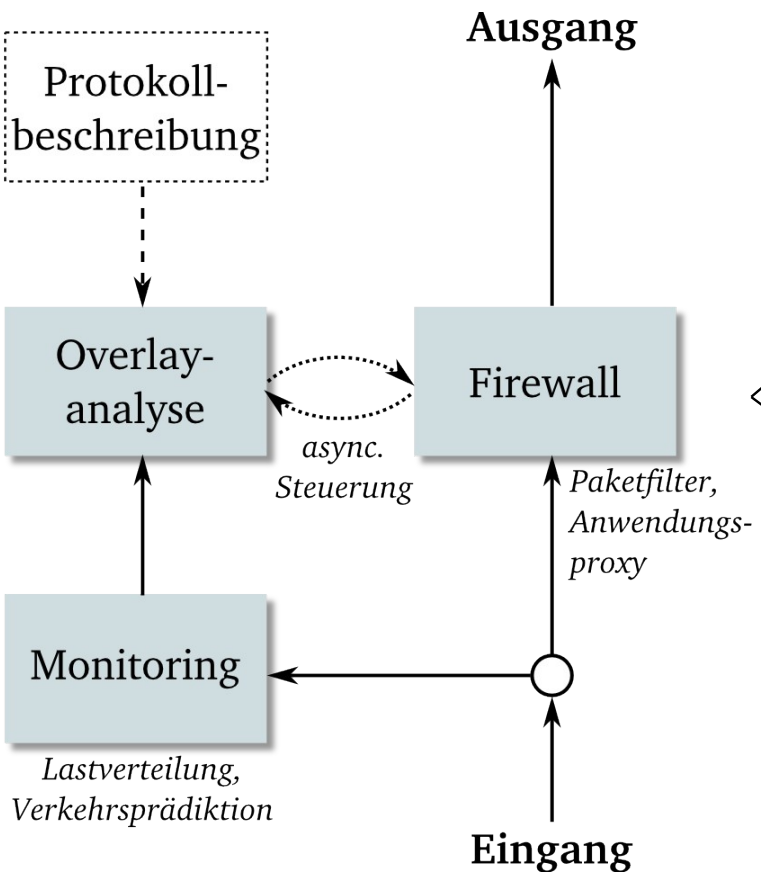
Brandenburg
University of Technology
Cottbus

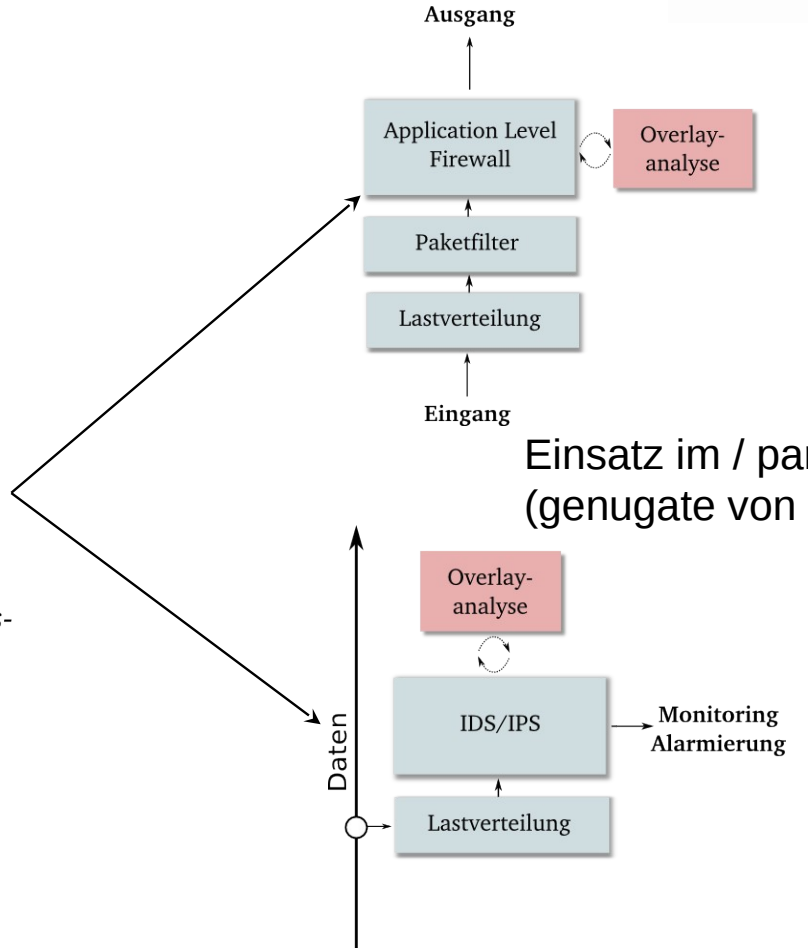
II.

Stand der Teilprojekte

PROJEKT-ÜBERGREIFEND

Architektonische Anpassungen





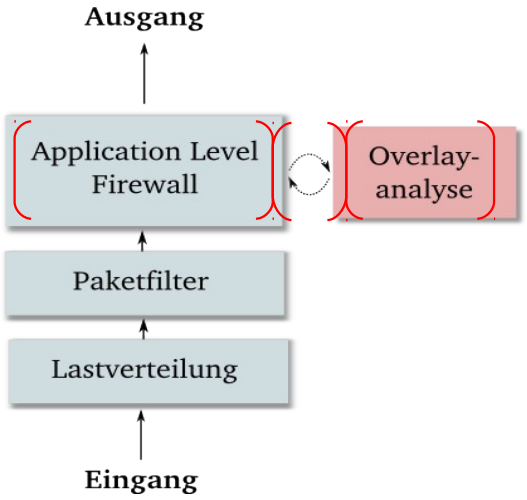
Einsatz im / parallel zu ALG (genugate von genua)

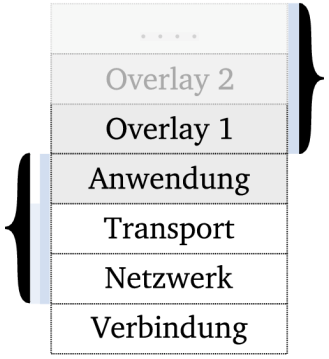
Einsatz im Bereich der Einbrucherkennung (Anpassung von Bro durch genua, Forschungsthematik an der BTU)

Brandenburg
University of Technology
Cottbus

A) genua

Strombasierte HTTP-Analyse / Schnittstelle Overlayanalyse





- **Ziele**

→ Analyse von Problemen auf der HTTP-Schicht (z.B. Response-Splitting, doppelte Header, invalide Inhaltslänge, …) + Bewertung des resultierenden Schadenspotenzials

→ Vorklassifizierung von Web-Anwendungen (Analyse Request/Response, Origin/Referer, Anfragemethoden)

→ Asynchrone Kopplung weiterer Analyse-komponenten (Overlay-Analyse), ohne dass die Latenz bei der Weiterleitung im Gateway spürbar steigt

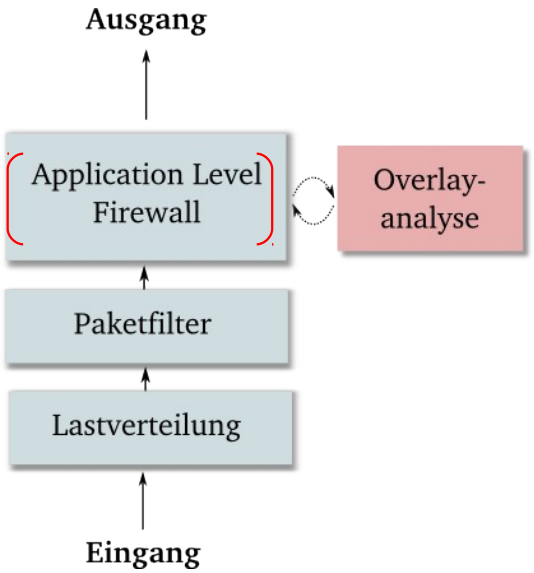
HTTP-Analyse

Brandenburg University of Technology Cottbus

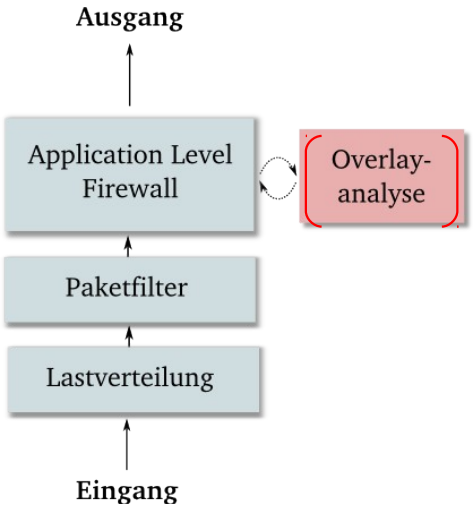
- Eingangsklassifizierung der Webanwendungen mittels HTTP-Analyse
  → Analyse Origin/Referer, Anfragemethode, Dokument-Typ, Response-Code

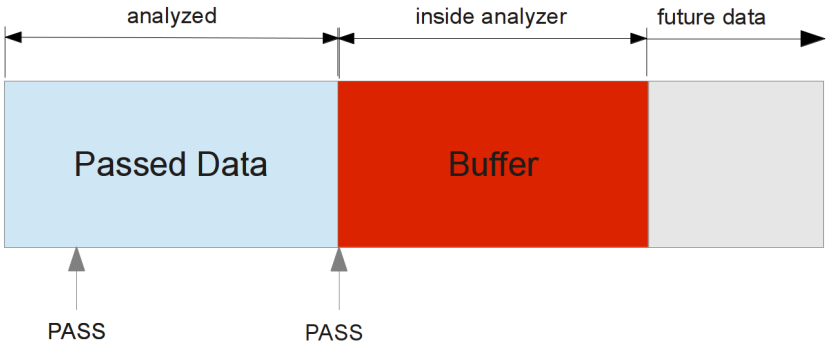
Beispiele:

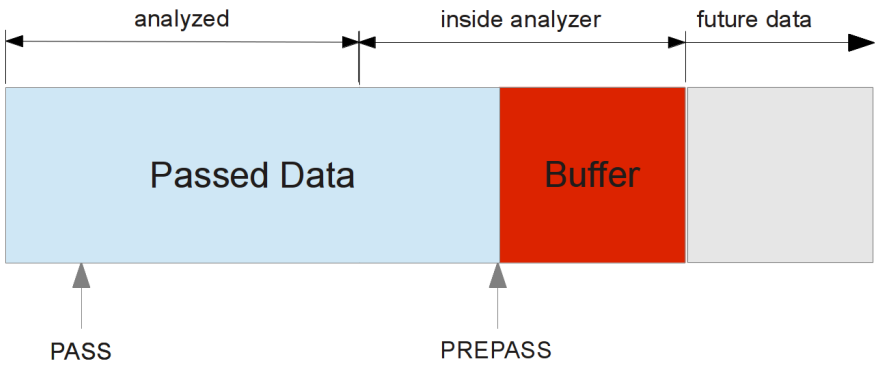
NOORIGIN weather.noaa.gov GET text/html
➔ Applets
NOORIGIN www.spiegel.de GET text/xml
NOORIGIN feeds.bbci.co.uk GET text/xml
NOORIGIN www.tagesschau.de GET application/xml
➔ XMLHTTPRequest/Bookmark/RSS-feed?
www.google.de www.google.de GET empty/empty
www.google.de news.google.de GET text/html
www.facebook.com www.facebook.com POST application/x-javascript
www.engadget.com www.facebook.com GET text/html

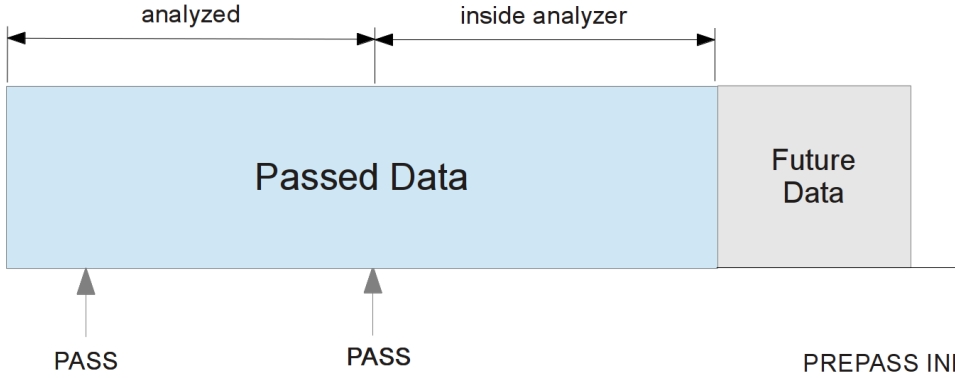


Asynchrone Ansteuerung der Overlay-Analyse









→ Tatsächliche Weiterleitung der Inhalte abhängig von Bewertung der Übereinstimmung mit Anwendungsprofil + potenzielle Gefährlichkeit von geänderten Inhalten

Übergang zum BTU-Teilprojekt

Brandenburg University of Technology Cottbus

- HTML-Analyse

→ Analyse der Einbettung von aktiven Inhalten:

➔ inline – direkte Inklusion

➔ isrc – Inkludierung vom selben Host, aus dem HTML-Dokument stammt

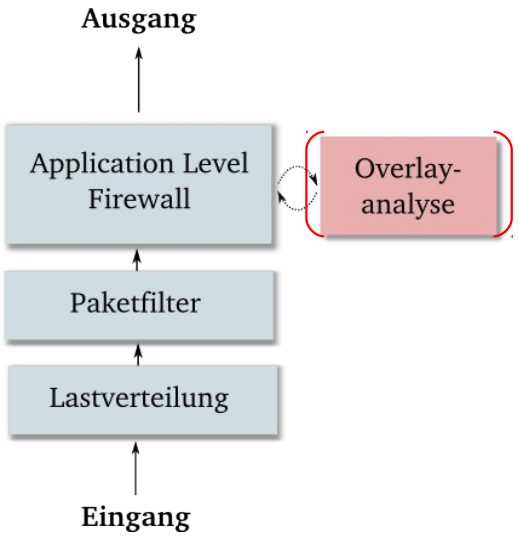
➔ ixsrc – Inkludierung von selber Domäne

➔ xsrc – Inkludierung von externer Domäne

➔ Normalisierung Script-Länge mittels: log(Länge)/log(5)

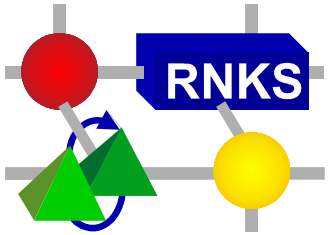
➔ Ergebnisse für Google vs. Facebook:

Google	Facebook
body.onload|3	a.onclick|2
center!div!script.inline	form.onsubmit|2
center!form!script.inline	html!head!script.xsrc
div.onload|1	html!head!script.inline
body!center!script.inline	html!body!script.inline
...	...



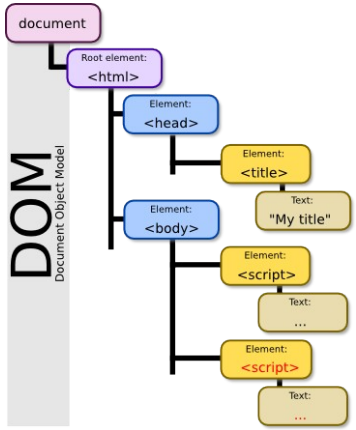
Brandenburg
University of Technology
Cottbus

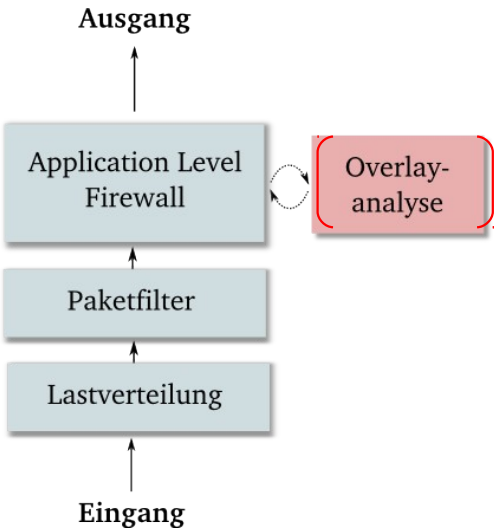
B) BTU Cottbus



Modellgenerierung für Web-Anwendungen

Brandenburg University of Technology Cottbus





- Ziele
  - ➔ Initiale Bewertung der Gefährlichkeit einer Web-Anwendung basierend auf den Inhalten (HTML/XML/SVG/CSS)
  - ➔ Generierung eines Modells der Web-Anwendung (im Fokus: JavaScript-Anwendungen)
  - ➔ Erzwingung bzw. Überprüfung dieses Modells zur Laufzeit, ggf. Bewertung von Änderungen
- Entwicklung bzw. Erweiterung der entsprechenden Parser
  - ➔ HTML/XML/SVG/CSS/JS-Parser + Analyse der Fehlerzustände
- WebApp-IDS
  - ➔ Analyse von Kodierverfahren (z.B. Änderung nach UTF-7, …) + problematischer HTML 5 Konstrukte (z.B. Data-URIs)

Signaturen für Web-Anwendungen (Probleme)

Brandenburg University of Technology Cottbus

- Interpretation von Dokumentinhalten:

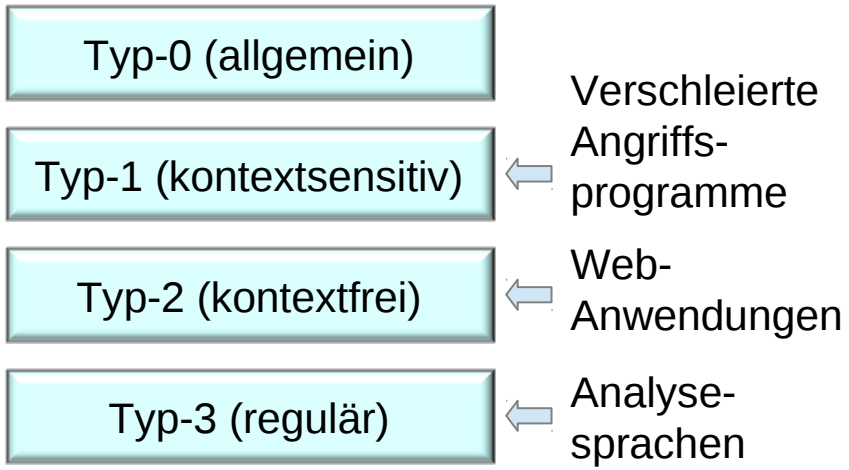
```
<html><head></head><body>
<script/>
alert(1);
</body></html>
```

- IE, Safari, Opera:

```
<html><head></head><body>
<script></script>
alert(1); <!-- Text!! -->
</body></html>
```

- Firefox:

```
<html><head></head><body>
<script>
alert(1); <!-- Script!! -->
</script></body></html>
```



Chomsky-Hierarchie im IDS-Kontext

Signaturen für Web-Anwendungen (Probleme)

Brandenburg University of Technology Cottbus

- Caching:

<html><head></head><body>

<script src="image.gif"></script>

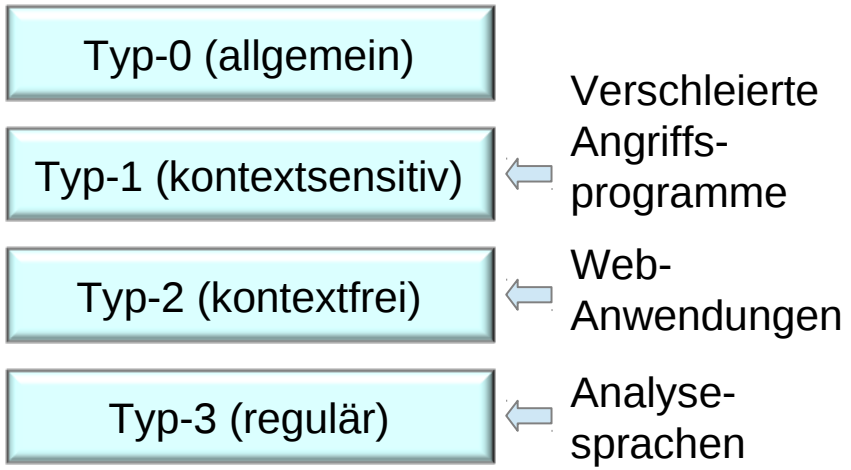
</body></html>

- image.gif (IE):

GIF89a=1;alert(1);

- IE „überprüft" Dateityp im Zweifelsfall auf Basis weniger Informationen aus Dateikopf („GIF89a")

➔gültiges Java-Script!!



Chomsky-Hierarchie im IDS-Kontext

Signaturen für Web-Anwendungen (Probleme)

Brandenburg University of Technology Cottbus

- Kodierung von Dokumentinhalten:
- Klassiker (IE):

¼script¾alert(1);¼/script¾

➔IE streicht achtes Bit bei US-ASCII

- Chrome, Safari:

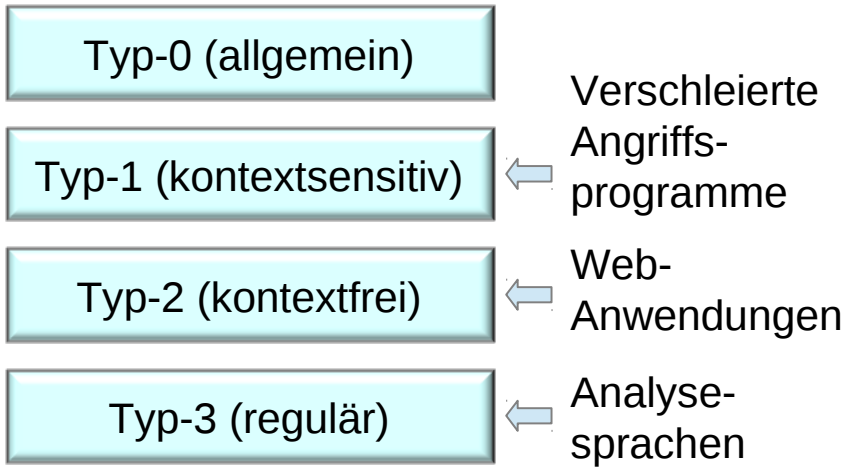
XXX

➔Unicode-Decoder „verschlucken“ oft ungültige Zeichen

- Firefox (HTML5):

<embed src="data:text/html; base64, PHNjcmlwdD5hbGVydCgxKTwvc2NyaXB0Pg==">

➔<script>alert(1)</script>

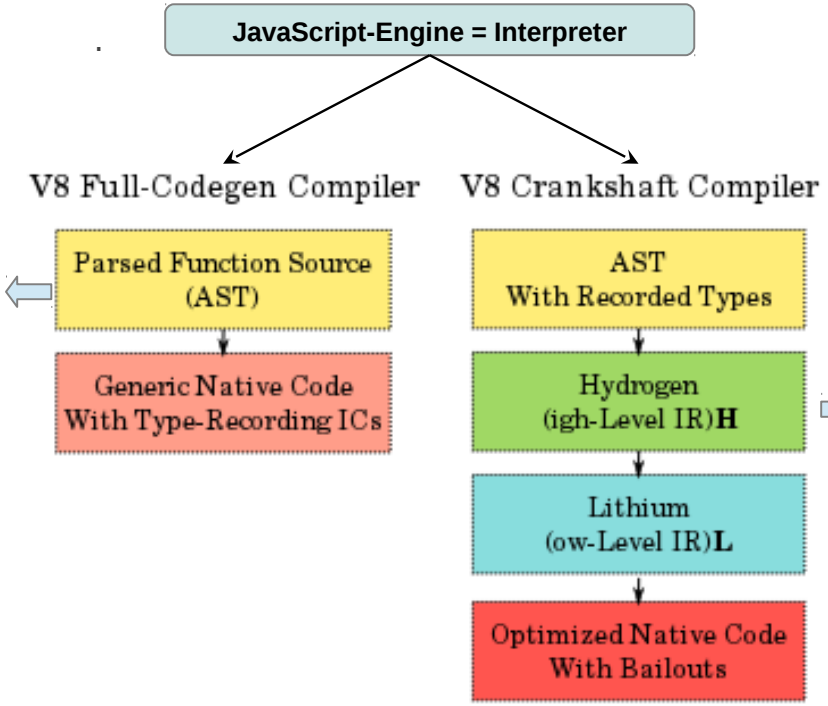


Chomsky-Hierarchie im IDS-Kontext

Modelle/Signaturen für JavaScript-Anwendungen (Probleme)

Brandenburg University of Technology Cottbus

Grundidee:
Extraktion von Kontrollflussgraphen aus abstrakten Syntaxbaum +Basisblockanalyse

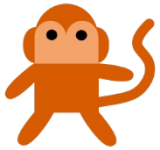


Vorhandensein von AST ist Glückssache (z.T. nur für einzelne Funktionen, falls benötigt)

Ausführung nur, wenn Code-anteil besonders oft durchlaufen wird

ab dieser Stufe sind analyserelevante Basisblockinformationen vorhanden

AST in Mozilla-Engine (Spidermonkey, Jaegermonkey, Ionmonkey) auch längerfristig verfügbar
→ daher im Projekt für Analysezwecke eingesetzt



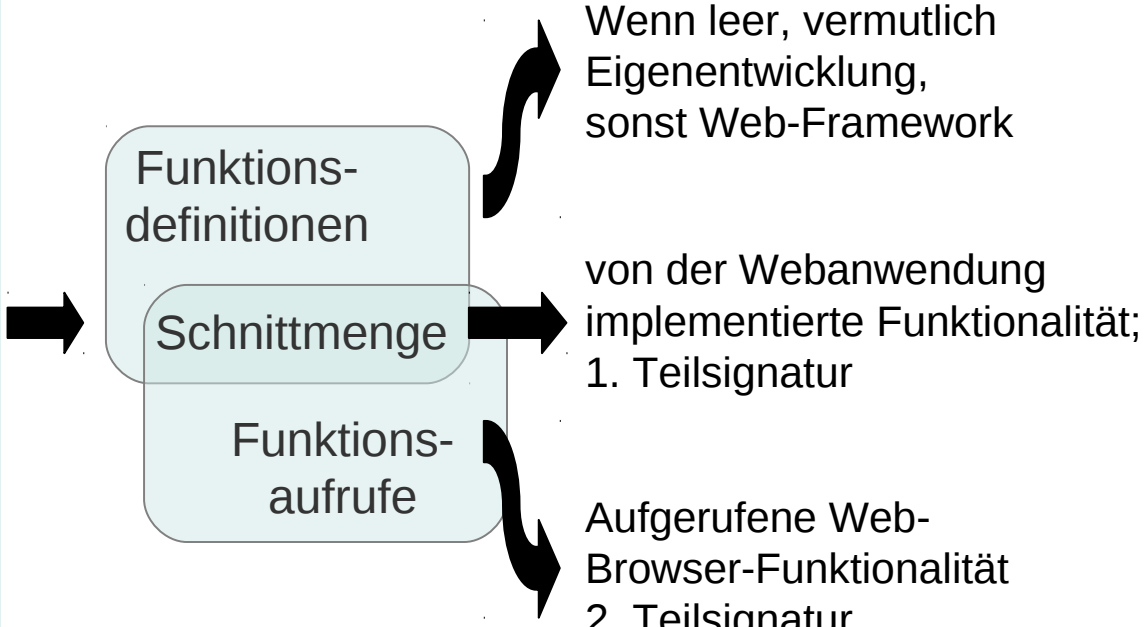
Quelle: http://wingolog.org/archives/2011/07/05/v8-a-tale-of-two-compilers

Signaturen für JavaScript-Anwendungen (Ansatz)

Brandenburg University of Technology Cottbus

- aktuell im Projekt eingesetztes Verfahren:
  - ➔ JavaScript-Syntaxbaumanalyse zur Erkennung von Funktionsdefinitionen und Funktionsaufrufen

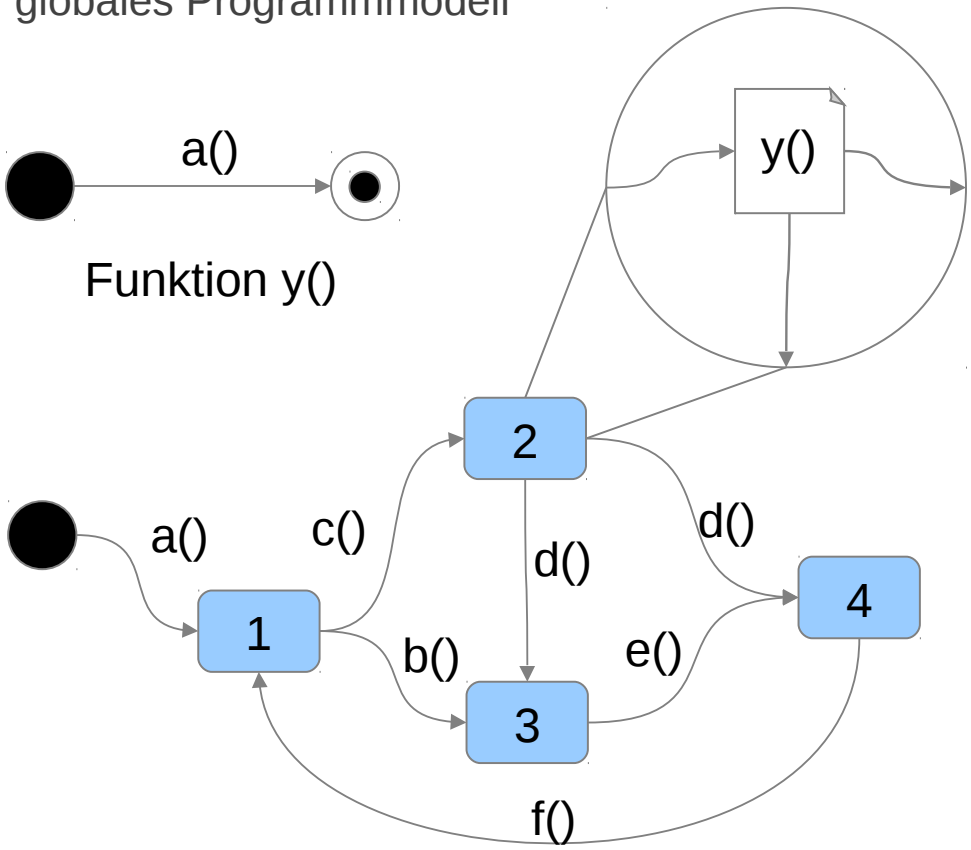
```
    js::PNK_VAR
        js::PNK_NAME
⇒ r
        js::PNK_FUNCTION
         js::PNK_UPVARS
          js::PNK_STATEMENTLIST
           js::PNK_SEMI
            js::PNK_LP
             js::PNK_DOT
⇒ push
              js::PNK_DOT
⇒ first
               js::PNK_DOT
⇒ y
                js::PNK_NAME
⇒ google

          ...
```



Weiterführende Verfahren für FAU-Teilprojekt

Brandenburg University of Technology Cottbus

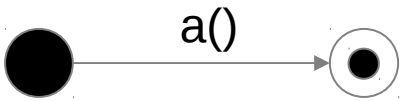
1. Generierung von Kontrollflussgraphen aus JavaScript-AST
2. Reduktion auf globales Programmmodell



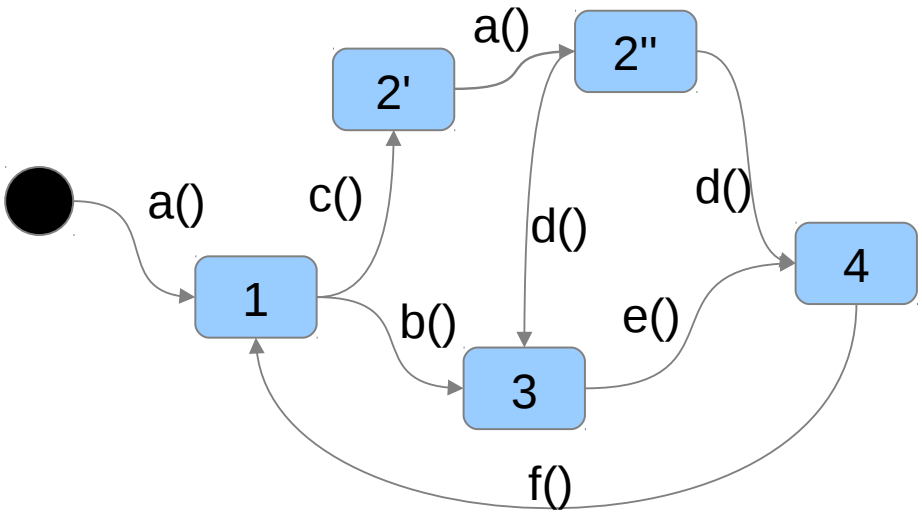
Weiterführende Verfahren für FAU-Teilprojekt

Brandenburg University of Technology Cottbus

1. Generierung von Kontrollflussgraphen aus JavaScript-AST
2. Reduktion auf globales Programmmodell
3. Generierung von N-Grammen für maschinelles Lernen



Funktion y()



Funktion x()

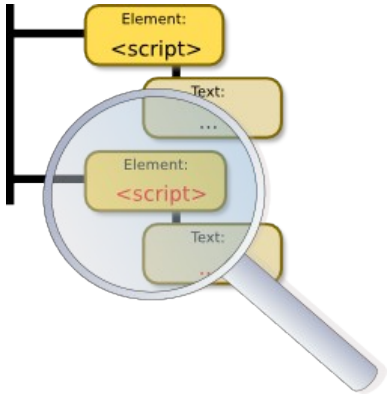
→ aca
→ acb
→ cad
→ bef

Brandenburg
University of Technology
Cottbus

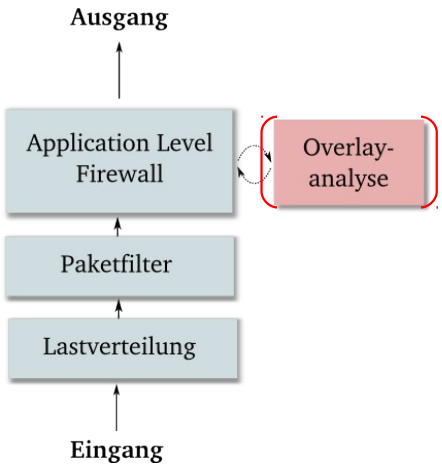
C) FAU Erlangen

Einschätzung der Code-Bösartigkeit

Brandenburg University of Technology Cottbus

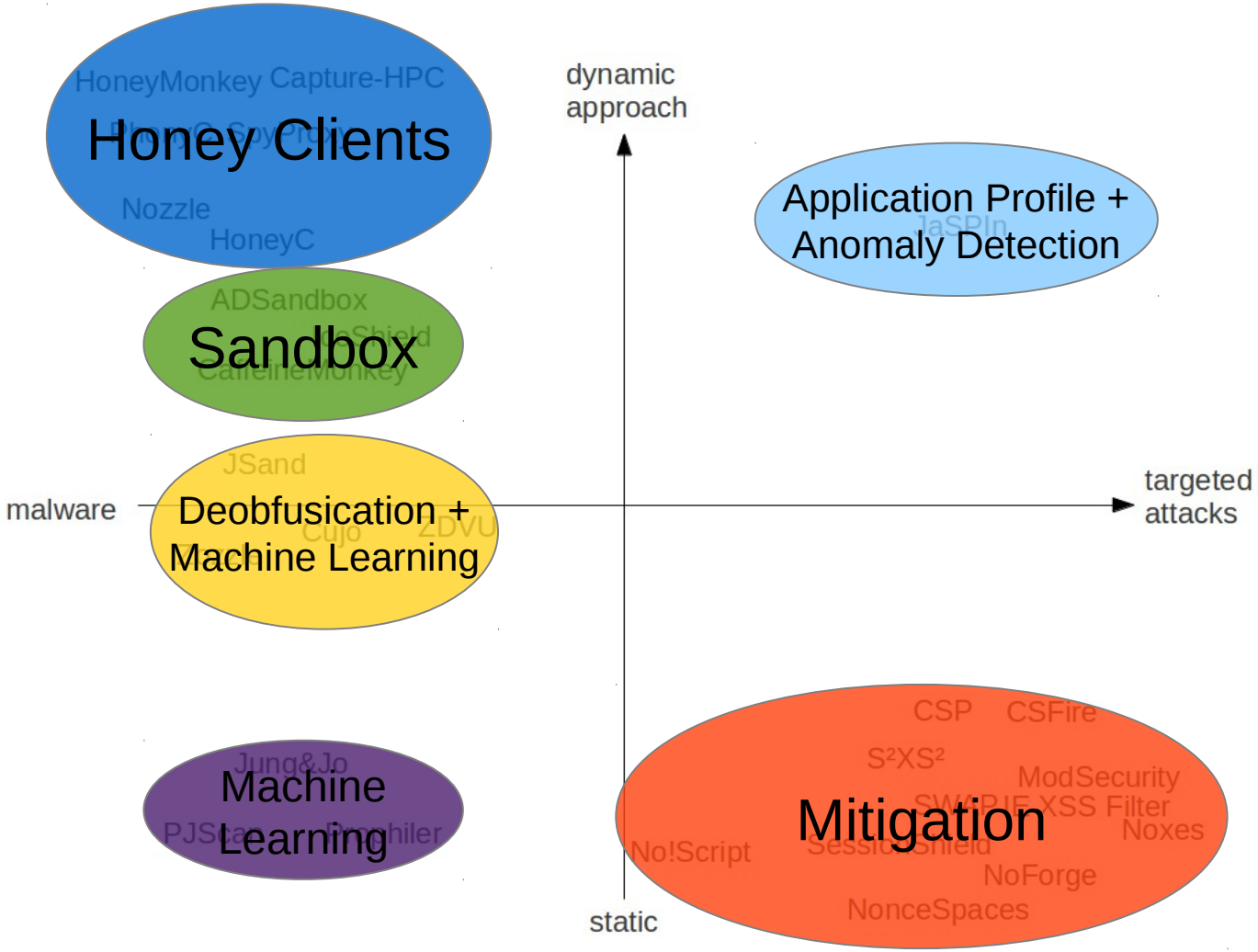


- Ziele
  - → Erkennung typischer Angriffsmuster von Malware (client-side/drive-by-download exploits, Heap Spraying, etc.)
  - → Erkennung zielgerichteter Angriffe (Session Hijacking, Phishing, CSRF, etc.)
  - → Basierend auf einzelnen JavaScript-Funktionen ohne weiteren Kontext
- Mittels Machine-Learning
  - → Anpassung eines existierenden Ansatzes (Prophiler, PJScan)
  - → Training je Domain (weil Coding-Style oder verwendete Funktionen abweichen)
- Zusätzlich Implementierung eines JavaScript-IDS
  - → Pattern-Matching, Heuristiken (Blocken/Entfernen potentiell gefährlicher Funktionen)



Erkennung von Schadsoftware / gezielter Angriffe

Brandenburg University of Technology Cottbus



Brandenburg
University of Technology
Cottbus

D) Ausblick

Aktuelle bzw. noch offene Entwicklungen

Brandenburg University of Technology Cottbus

- genua
  - → Kopplung eigener Prototypen mit *libpadiofire*
  - → DB-Backend für Speicherung von Anwendungsmodellen, Angriffssignaturen je Domäne
- BTU Cottbus
  - → Signaturen für vollständige Web-Anwendung basierend auf HTML/XML/SVG/CSS/JS
  - → Socks 4/5 Proxy mit *libpadiofire* für Dauertest der eigenen Entwicklungen
- FAU Erlangen
  - → Analyse von Veränderungen am AST und Bewertung potenziell sicherheitskritischer Änderungen

Brandenburg
University of Technology
Cottbus

Fragen?

Projektstand - Zusammenfassung

Brandenburg University of Technology Cottbus

- GeNUA:
  - Schnittstelle zur Anbindung der GeNUA Firewalls an die Overlayanalyse
  - Verhaltensbasierte HTTP- (Origin/Referer-Beziehungen) und HTML-Analyse (Vorgehensweise bei Einbettung von Script-Code)
- BTU Cottbus:
  - Abstrakte Syntaxbaumanalyse (AST) für JavaScript-Code
  - Extraktion von Signaturen für externes Verhalten aus JavaScript-AST (Interaktion der Web-Anwendung mit dem Browser)
- FAU Erlangen:
  - Untersuchungen zur Erkennung von Schadsoftware und gezielten Angriffen
  - Festlegung von Methoden zur Anomalieerkennung und Detektion von potenziell gefährlichen Funktionsaufrufen
    ☞ Ziel: JavaScript-IDS zur Erkennung von XSS

Assoziierte Partner:

- Universität Innsbruck: Masterarbeit zur Erkennung von Webanwendungen mittels Vermont
- IXIA: Monitoring-Toolkit Vermont mittels Lastgenerierung (Web-Verkehr) evaluiert