Application Security Trends and Challenges

SecAppDev 2015

Wouter Joosen

iMinds- DistriNet, Department of Computer Science, KU Leuven

February 23, 2015

DistriNet iMinds KU LEUVEN

Trends & Context

Technology Trends

- [1/3] *Integration of software in the "physical world"*
  - *CPS – Cyber Physical Systems*
  - *IoT – Internet-of-Things*
  - *Computational capacity is omnipresent*
- *Example:*
  - *TRANSITION, From ad-hoc code development to code reuse through middleware for networked embedded control systems*

Technology Trends [2/3]

- Intelligence, relevant data all over the place….
  - Context-aware computing unfolding beyond location and profile…
  - *Strong dependencies* between sensing equipment, data processing entities and storage platforms
  - Computational capacity is omnipresent, so is analytics…
- *Example:*
  - *CAPRADS, A Context-Aware Platform for RApid Decision Support*

Technology Trends [3/3]

- Cloud Computing (*the trivial one*)
  - Ultimately determining the delivery model of software and services…
  - Flexible “software-defined” architectures to deal with rapid change, upgrading, reconfiguration, scaling etc.
  - (But how about the attack surfaces?)

- *Examples*
  - *DeCoMAdS: **De**ployment and **Co**nfiguration **M**iddleware for **Ad**aptive **S**oftware-as-a-Service.*
  - $(DMS)^2$: ***D**ecentralized **D**ata **M**anagement and **M**igration for **S**oftware-as-a-**S**ervice*

Application Development today

- Despite all technology trends…..

Application Security

- Many technologies are available
- Some still being developed, but on the horizon

- Yet other are subject to a strategic investment (Still R&D)

Available?

- Dynamic Application Security Testing (DAST)
- …Static Application Security Testing (SAST)

- SIEM (Security Incident and event management)
- … Context-aware security (e.g. credentials that are requested/presented can depend on location).
- etc.

Note: Both *development* support and run time *services/facilities* to be integrated…

Following soon

- Mobile application security testing
- Web application firewalls

- Professional Services?
- Application Security as-a-service?

Many sources confirm …

E.g. market analysts such as Gartner, Forrester etc.

Some of the heavy lifting

- DevOps & Security
- Protected Mobile Browsers
- (Runtime) Application Self-Protection

… and even further…

Illustration 1:
Isolating and Restricting Client-Side JavaScript

(*towards a secure browser*)

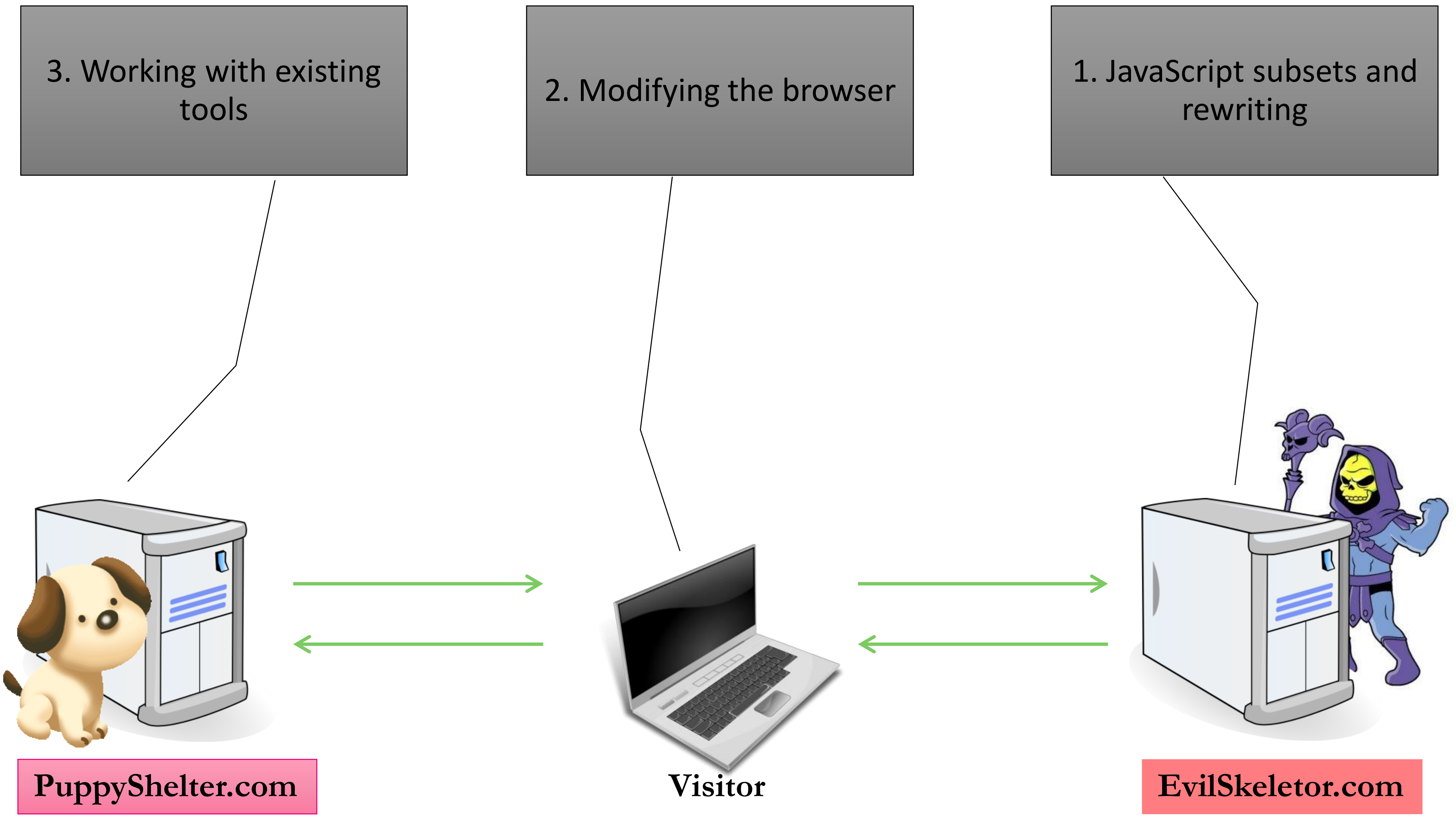
Featuring the PhD thesis of Dr. Steven Van Acker

January 6, 2015

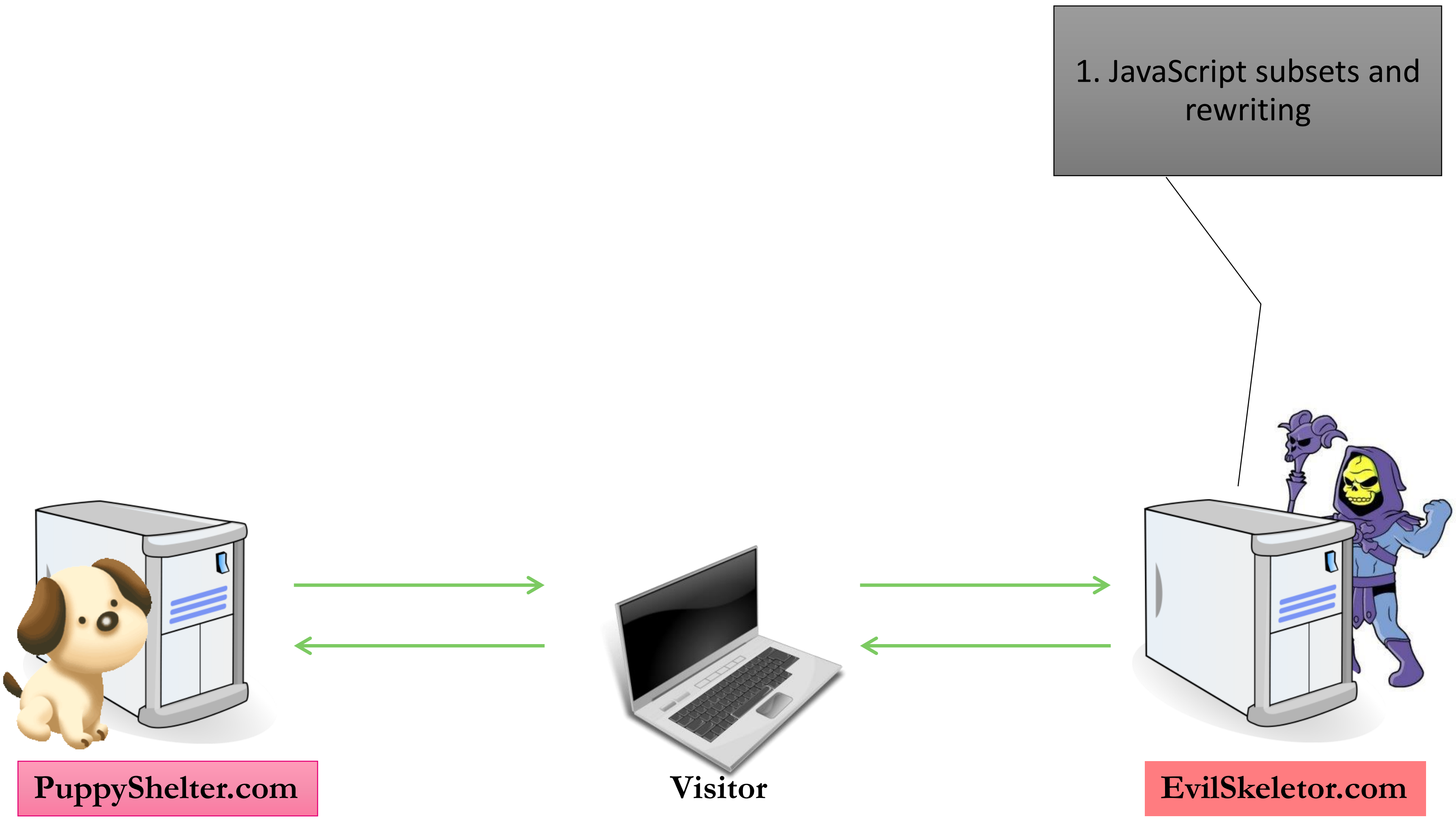
DistriNet

iMinds KU LEUVEN

Where to fix the problem?

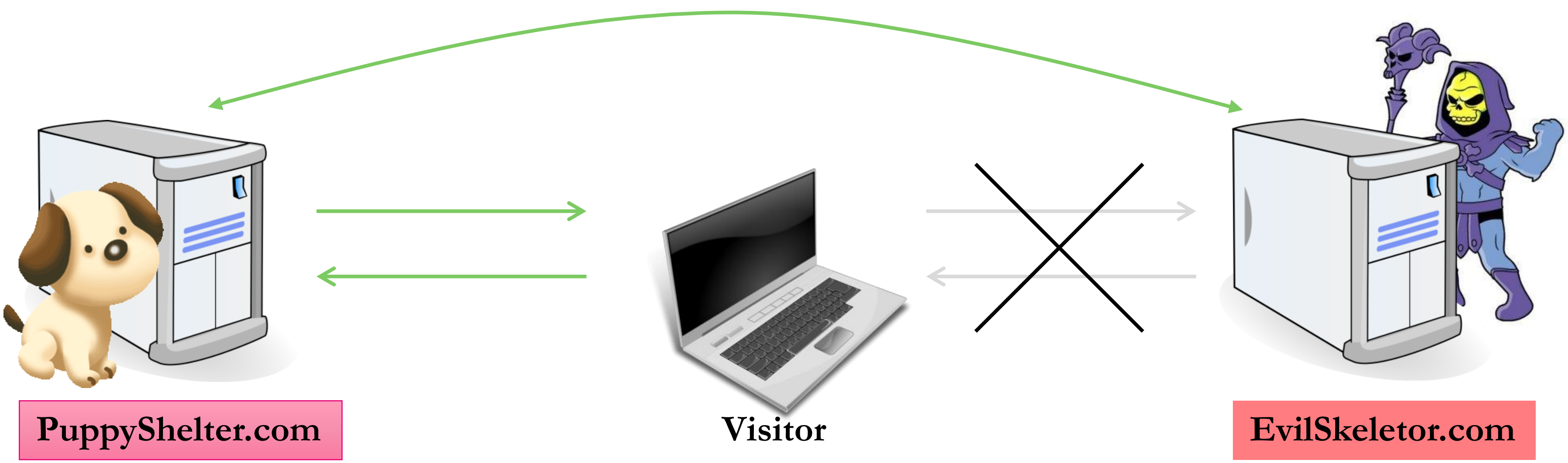


Where to fix the problem?

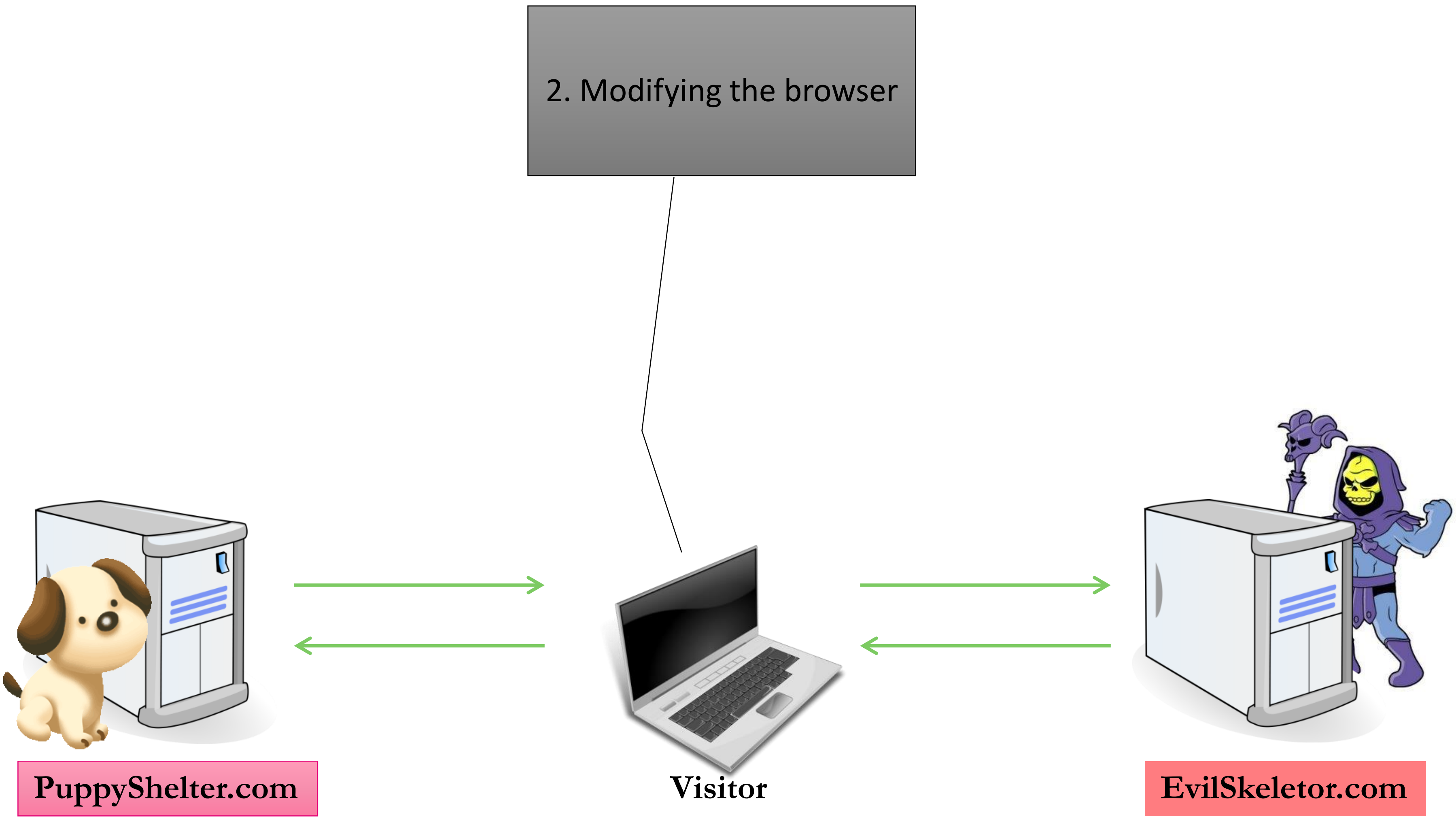


JavaScript subsets and rewriting

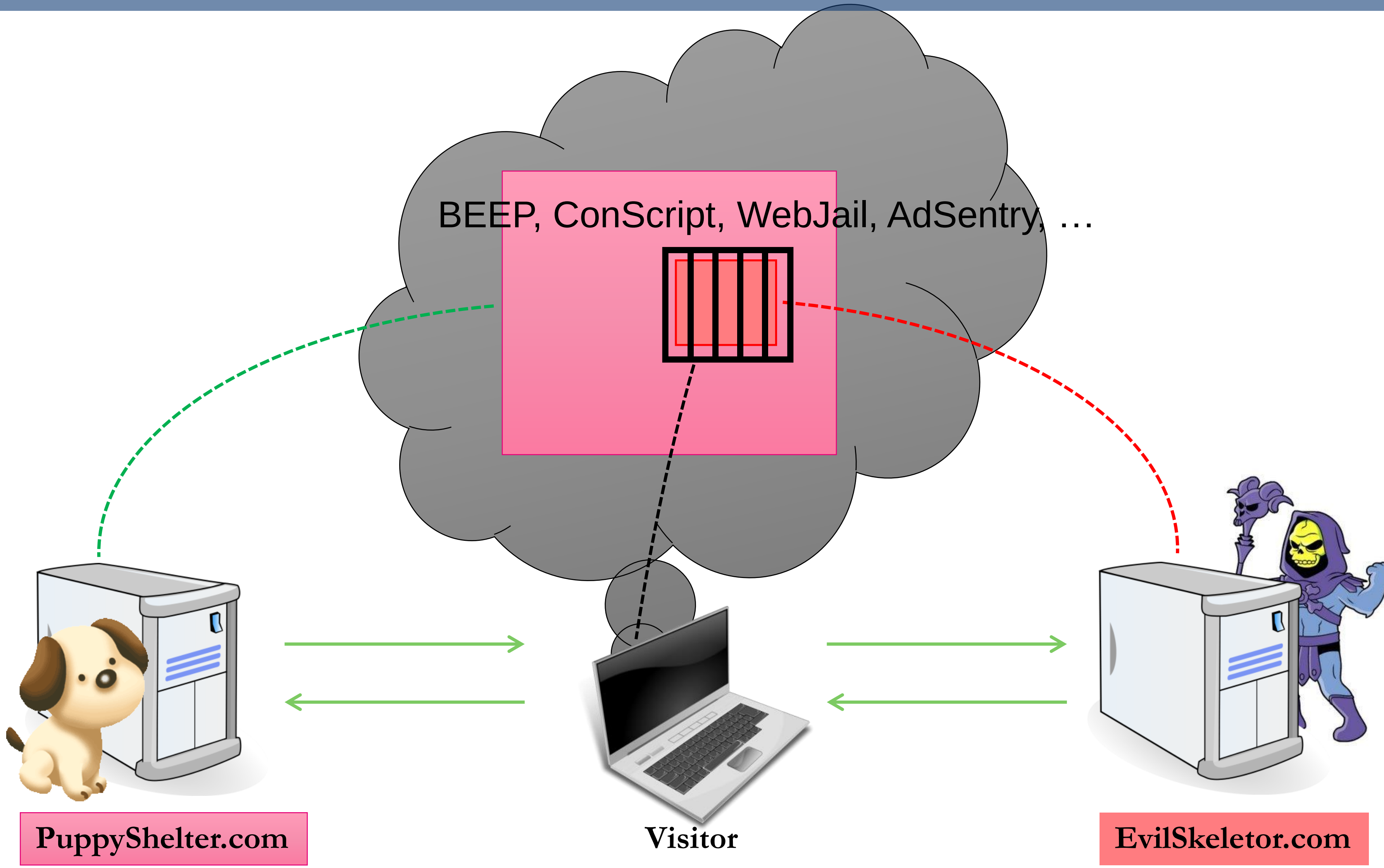
- Main idea: analyze JavaScript before executing it, rewriting if necessary
- Examples: Caja, FBJS, ADsafe, BrowserShield, …
- Unfortunately:
  - Analyzing JavaScript is difficult. Using a JavaScript subset makes it easier but requires effort from third-party
  - Rewriting JavaScript changes architecture of the Web



Where to fix the problem?



Modifying the browser

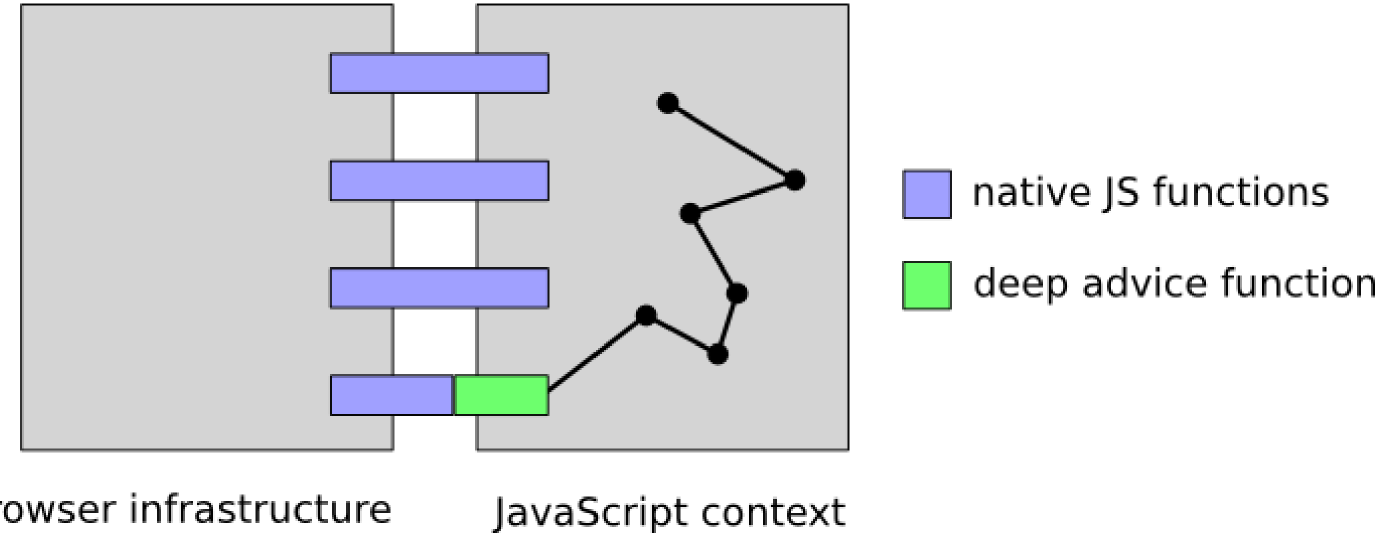


WebJail: Least-privilege Integration of Third-party Components in Web Mashups

Steven Van Acker, Philippe De Ryck, Lieven Desmet, Frank Piessens, Wouter Joosen
ACSAC 2011

WebJail: main idea

- Restrict sensitive JavaScript functionality in the DOM of an iframe
- An advice function intercepts calls to a DOM function and mediates access
- All access-paths go through the advice function
- Enforced in the browser, advice is locked away safely

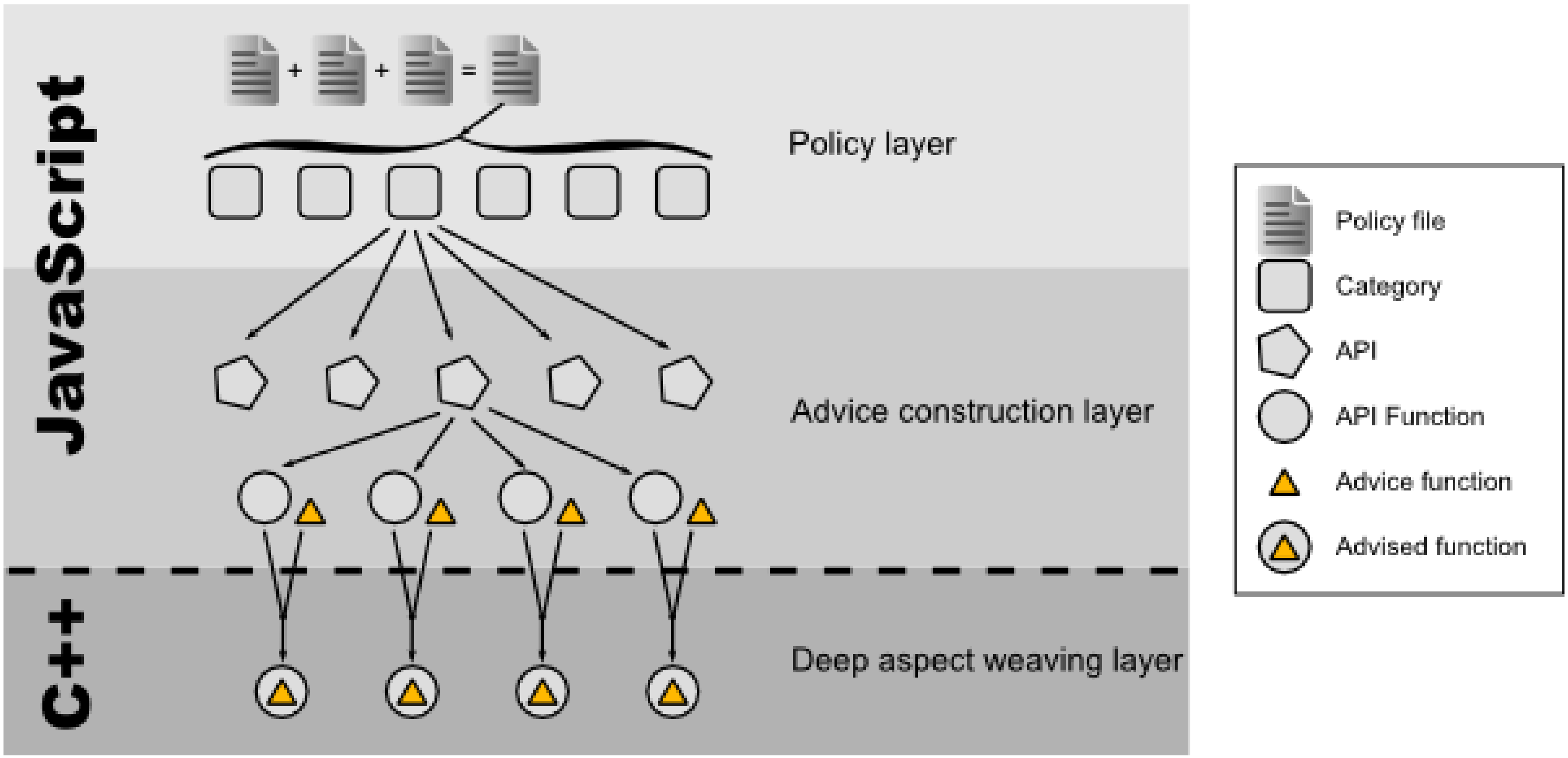


WebJail: policies

- Easy to use policy language
- All JavaScript functionality divided into 9 categories:
  - DOM Access
  - Cookies
  - External communication
  - Inter-frame communication
  - Client-side storage
  - UI and rendering
  - Media
  - Geolocation
  - Device access

```
{
   "framecomm" : "yes",
   "extcomm" : [ "google.com", "youtube.com" ], "device"
   : "no"
}
```

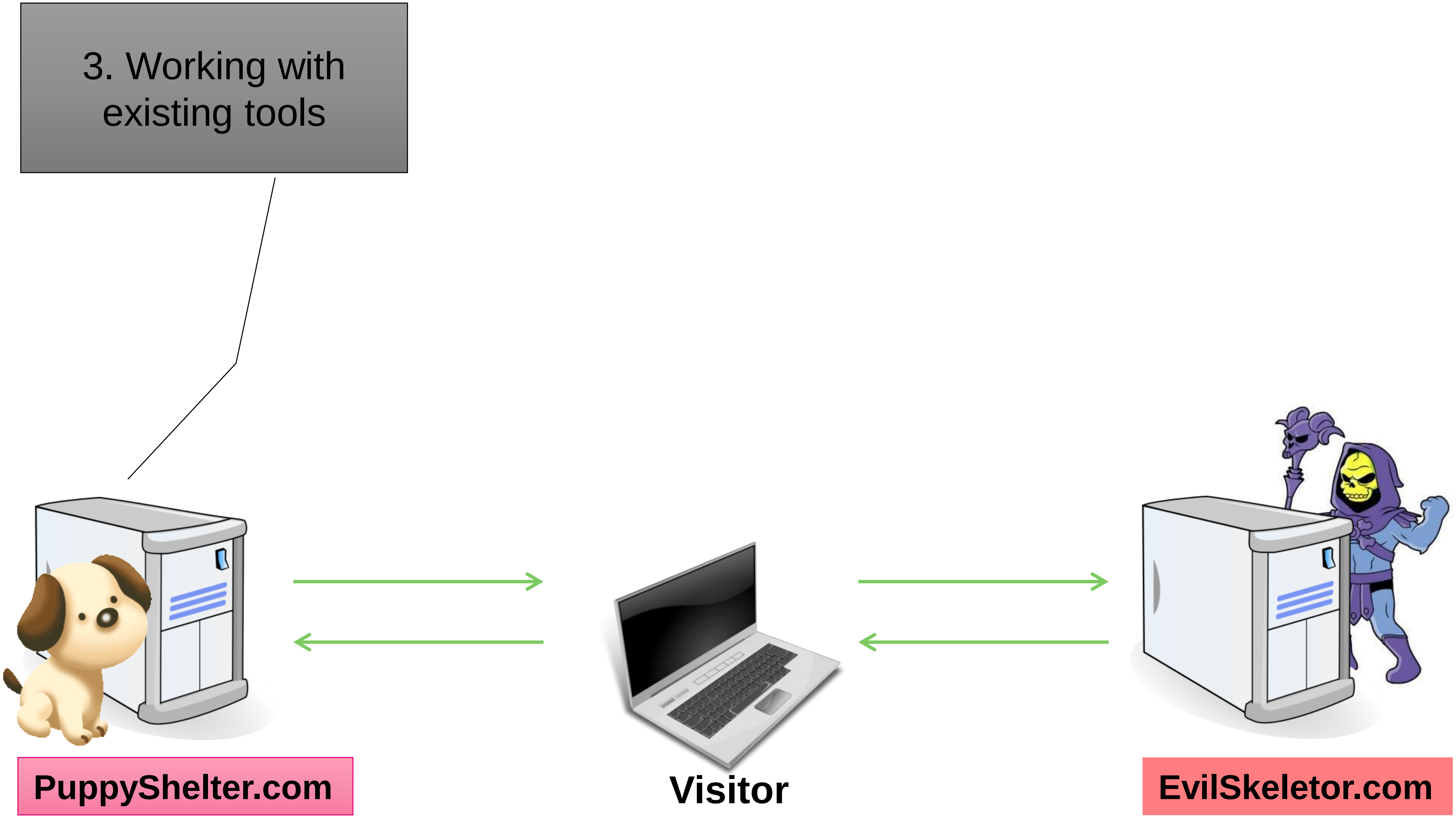
WebJail: architecture



WebJail: conclusion

- WebJail is a viable JavaScript sandbox
  - Full mediation
  - Fast
- Unfortunately:
  - Deploying a browser modification to all browsers on the Web is hard
  - “Just get the modification adopted by W3C so all browsers implement it” → not so easy…

Where to fix the problem?



JSand: Complete client-side sandboxing of third-party JavaScript without browser modifications.

Pieter Agten, Yoran Brondsema, **Steven Van Acker**, Phu Phung, Lieven Desmet, Frank Piessens

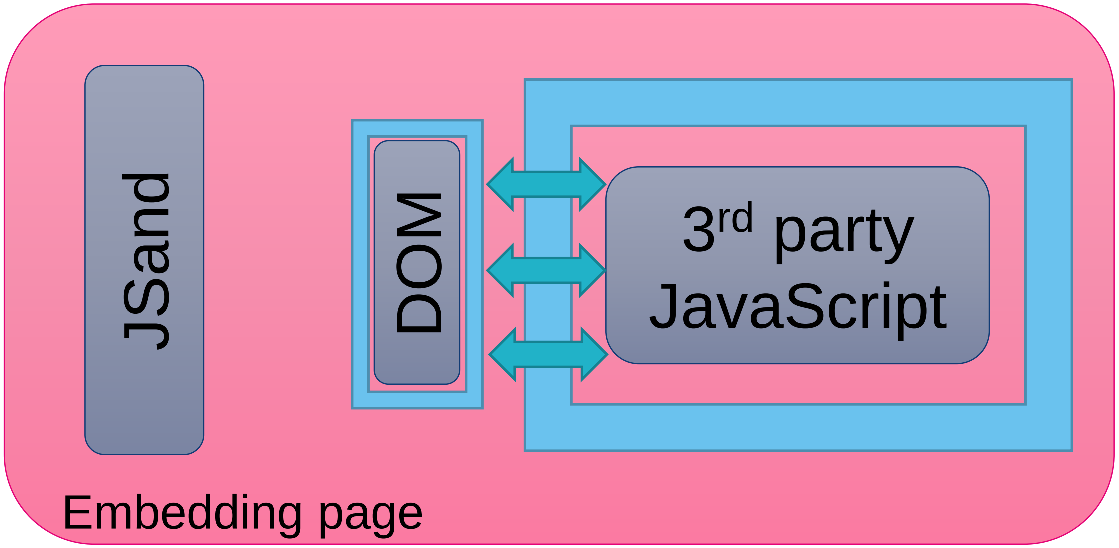
ACSAC 2012

JSand: object-capability env.

- Object capability environment:
  - All functionality is encapsulated in objects
  - References to those objects can not be forged
  - Without reference to a certain object, there is no access to its functionality
- E.g. window.alert()
  - alert is a property of the window object
  - Without access to the window object, alert() can not be used
- Secure ECMAScript is object-capability safe
  - Subset of JavaScript strict mode

JSand: Under the hood

- Download third-party script directly to browser
- Load script in **isolated** object-capability environment using Google's Secure ECMAScript
- Enable access to outside using *membrane* around DOM
  - Policy determines permitted operations



JSand: Conclusion

- JSand is also a viable sandboxing solution
  - Full mediation
  - Works out-of-the-box on modern browsers
- Unfortunately:
  - Reusing functionality that was not intended for sandboxing results in unwanted performance hit

Observations

- There is no silver bullet (yet)
- Reusing currently standardized functionality is not optimal
  - E.g. performance overhead
- Specialized JavaScript sandboxing functionality is required
  - Proof of concept as browser modification
  - But in long run, functionality must be standardized

Illustration 2:
Security Primitives for Protected Module Architectures

Featuring the PhD thesis of Dr. Raoul Strackx

December 17, 2014

DistriNet
iMinds KU LEUVEN

Emerging technology: PMA's

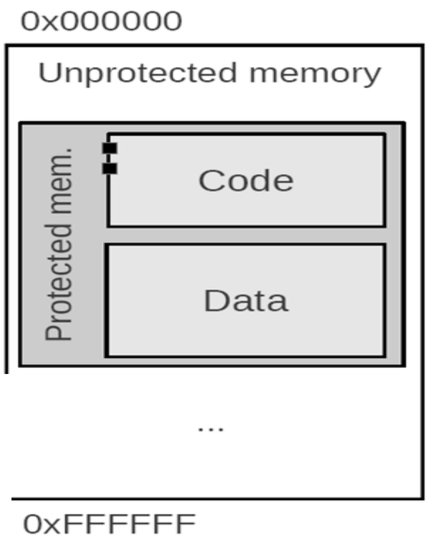
- Protected Module Architectures:
  - Low-level security architectures that implement an "inverse sandbox": protect a module from a buggy or malicious environment
    - E.g. run code securely even on top of a kernel infected with malware

Emerging technology: PMA’s

- Implementations
  - Pioneering work by Parno et. al. at CMU: the Flicker system
    - https://sparrow.ece.cmu.edu/group/flicker.html
    - Bryan Parno was awarded the ACM 2010 doctoral thesis award for this work
  - Follow-up implementations, including several from iMinds:
    - Fides (Strackx et al, CCS 2012), Sancus (Noorman et al., Usenix Sec 2013)
  - INTEL publicly announced their implementation quite a while ago (snclaves in SGX)
    - http://software.intel.com/en-us/intel-isa-extensions#pid-19539-1495

Protected module architecture (simplified)

- Modules consist of:
  - A code section, with designated entry points
  - A data section (also containing control data)
- The PMA:
  - Controls creation/deletion of modules
  - Enforces a PC-based access control model



from \ to	Protected			Unprotected
	Entry point	Code	Data	
Protected	r x	r x	r w	r w x
Unprotected	x			r w x

DistriNet iMinds KU LEUVEN

Some Achievements

- How can Protected Module Architectures efficiently, securely and reliably persist state?
- What is the minimal hardware support required to implement PMA’s:
  - That support remote attestation
  - That support state continuity
  - That do not need software in the TCB

Research challenges ahead

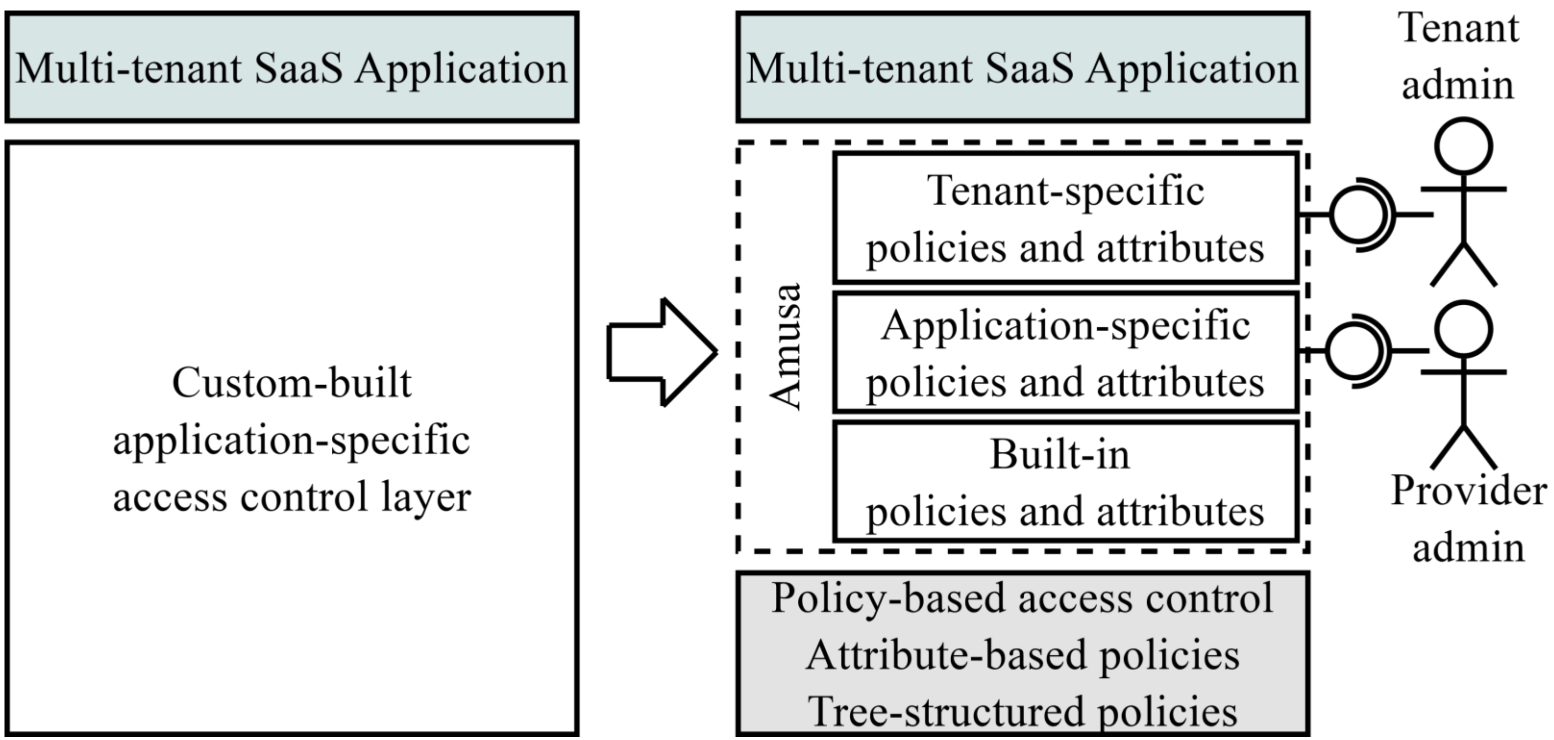
- How do we offer higher-level abstractions for these low-level security architectures?
  - Key idea: maintain the modularity properties of source code at machine code level by secure compilation.
- How do we provide assurance of the correctness of the protected module itself?
  - These modules might be small enough to be amenable to formal verification

This type of work may lift self-protection to the next level

Illustration 3: Amusa

Access control middleware for multi-tenant SaaS applications

Goal



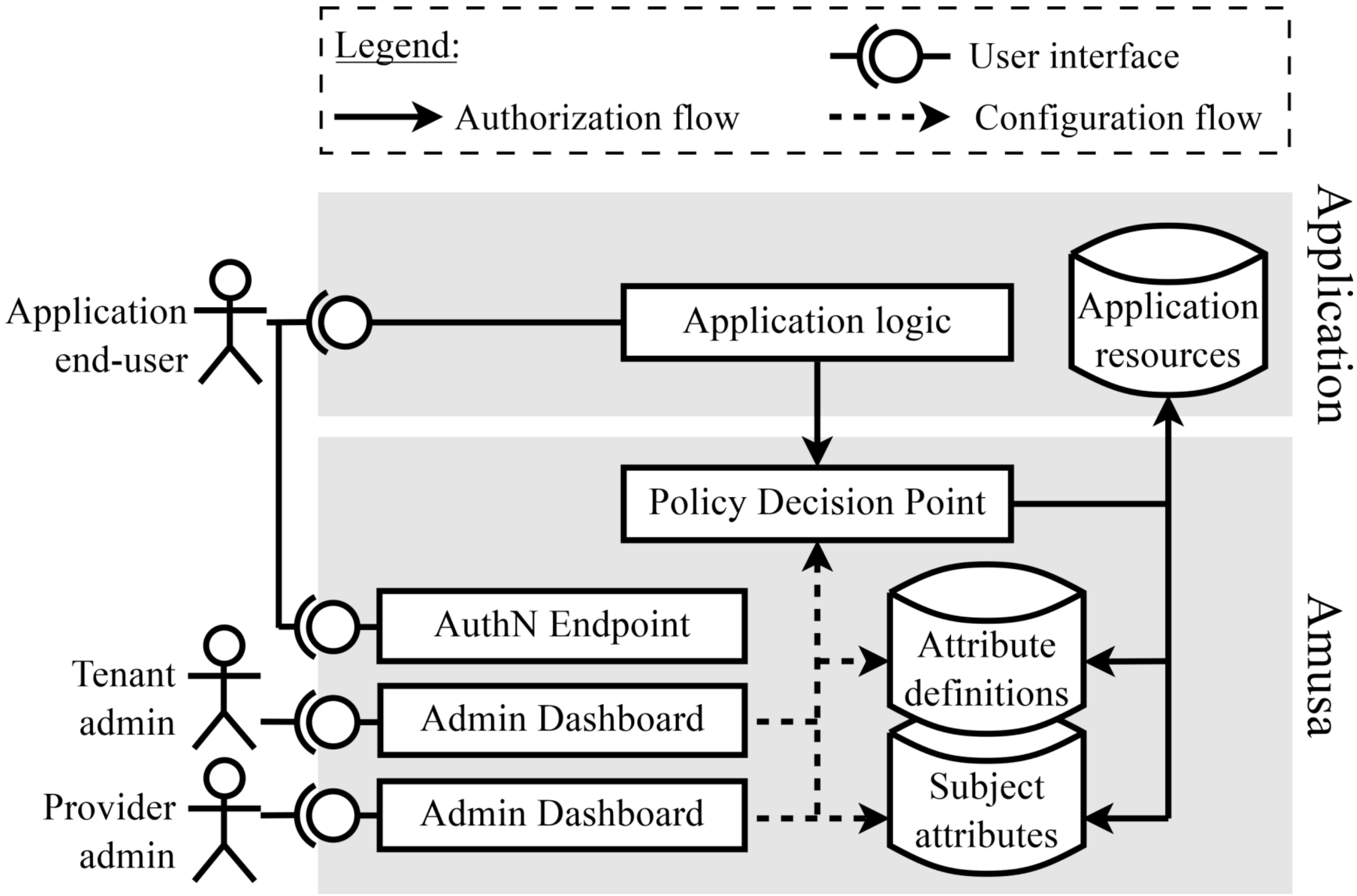
- Combine policies securely
- Enforce at run-time

Three-layered access control mgmt

	Large Bank	*Press Agency*
Tenants	subj.assigned_customers	subj.region
eDocs	subj.email, subj.tenant_credit, res.sender	
Amusa		

	Large Bank	*Press Agency*
Tenants	Deny if not res.owner in subj.assigned_customers Override isolation if subj.tenant == "PartnerA"	Deny if subj.region != "Europe"
eDocs	Deny if subj.tenant_credit < action.cost Override isolation if res.owner in subj.reseller_tenants	
Amusa	Default tenant isolation policy	

Logical architecture



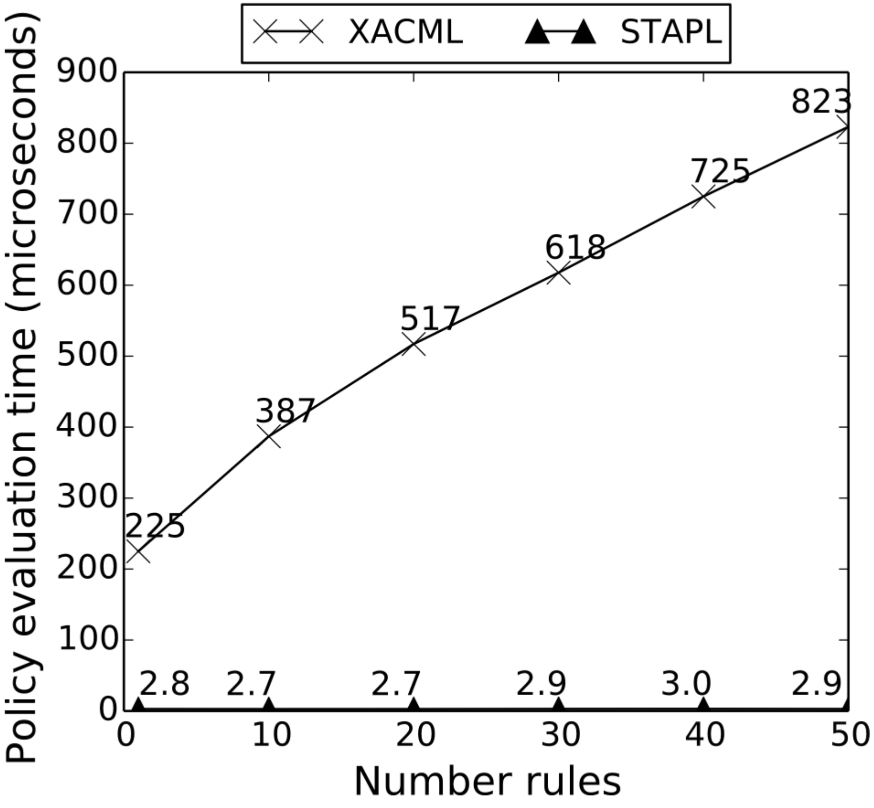
STAPL

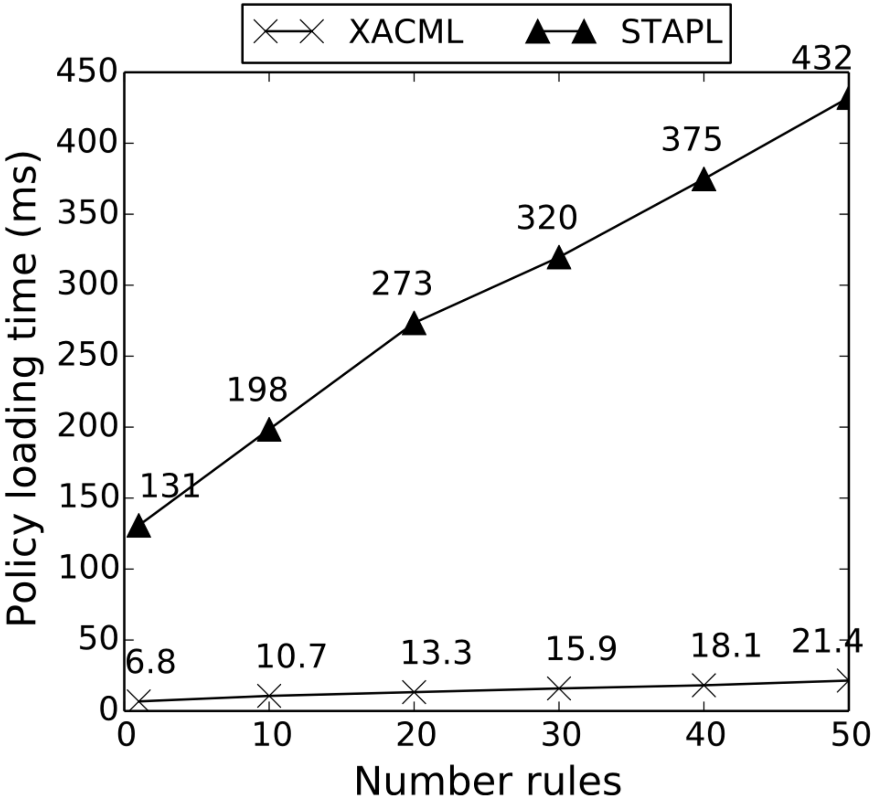
The Simple Tree-structure Attribute-based Policy Language

A note on the relative ease of specifying policies

		Attr. def.	Obl. def.	Pol. spec.	Total
E-health	**XACML**	-	-	706	706 (100%)
	ALFA	168	3	259	430 (60.9%)
	STAPL	27	4	84	115 (**16.3%**)
E-docs	**XACML**	-	-	1332	1332 (100%)
	ALFA	175	3	514	692 (52.0%)
	STAPL	31	4	196	231 (**17.3%**)

Performance evaluation



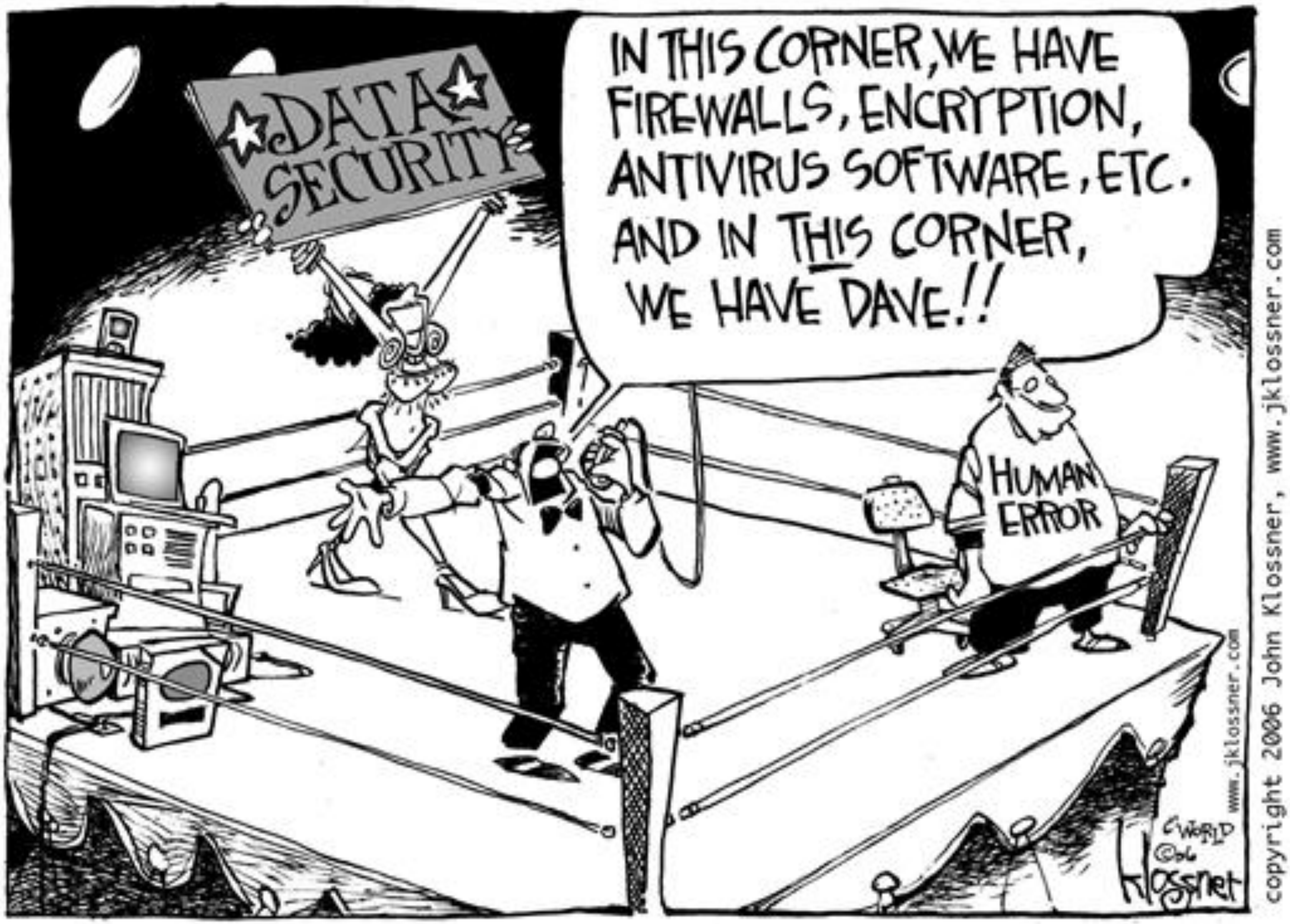


Summary

- Focus on multi-tenant IAM
- Main technology:
  - policy-based access control with attribute-based tree-structured policies
  - STAPL: policy language (DSL)
    - suited for extending with new technologies
- .. Clearly WIP.

Business Intermezzo

- Attitude of the market
  - Security Provider side: point solutions and network level technology taking a lot of spotlight.
  - Software Vendor (ISV side): managing performance indicators (e.g. #bugs found) may not truly support application security
  - Agility remains obviously crucial….
- So is there any room for Secure SDLC?☺☹


DATA SECURITY
IN THIS CORNER, WE HAVE FIREWALLS, ENCRYPTION, ANTIVIRUS SOFTWARE, ETC. AND IN THIS CORNER, WE HAVE DAVE!!
HUMAN ERROR
copyright 2006 John Klossner, www.jklossner.com
www.jklossner.com
Klossner

MUST consider:

4 angles

(1) Life Cycle Support

(not XP)

(2) Expressive Power

Remember policy languages…

(3) Composition & Transformation

Automation is crucial for cost purposes and robustness…

(4) Dev Ops…

Deployment (configurations etc.) must/will become an integrated part of software

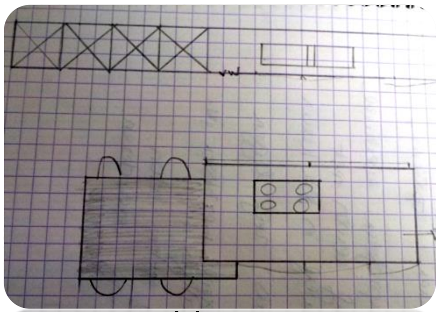
Illustration 1: RE

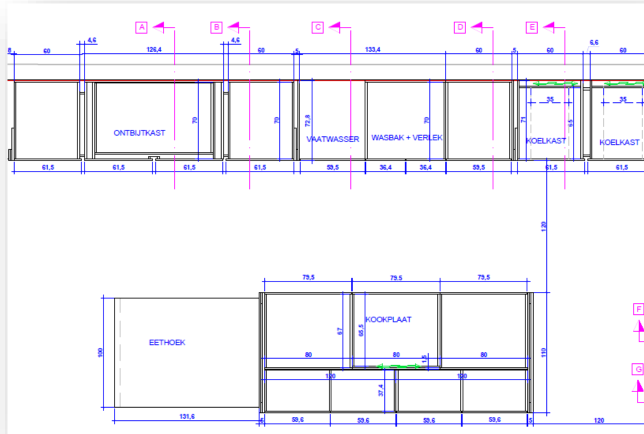
Requirements Engineering

Privacy threats in software architectures

Development lifecycle

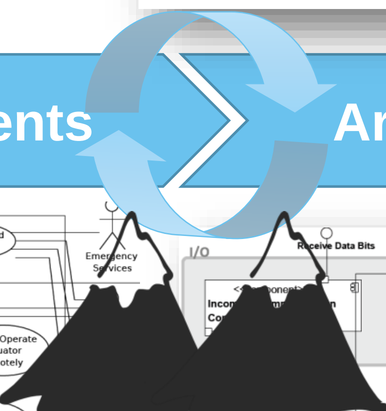
KITCHEN



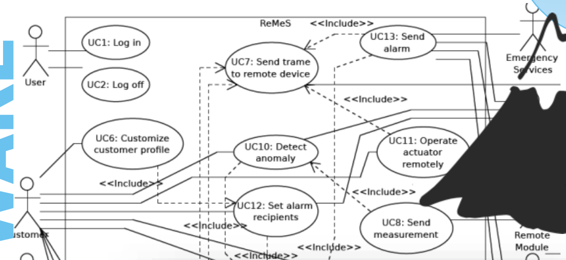


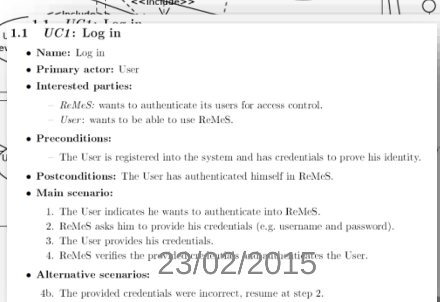
- white
- some storage space
- sleek design

Requirements → Architecture → Implementation



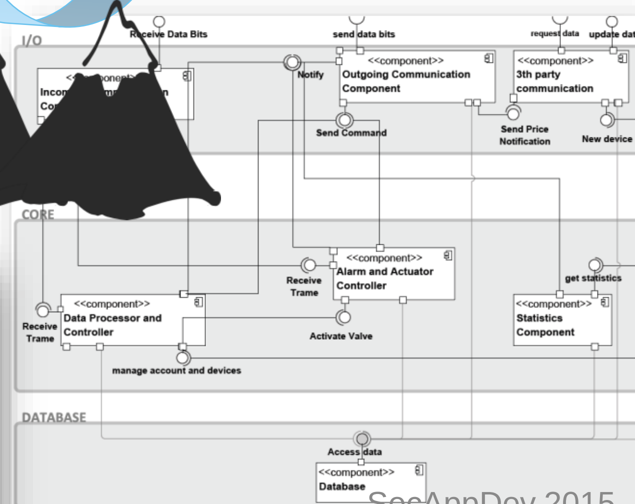
SOFTWARE





1.1 *UC1*: Log in

- **Name:** Log in
- **Primary actor:** User
- **Interested parties:**
  - *ReMeS:* wants to authenticate its users for access control.
  - *User:* wants to be able to use ReMeS.
- **Preconditions:**
  - The User is registered into the system and has credentials to prove his identity.
- **Postconditions:** The User has authenticated himself in ReMeS.
- **Main scenario:**
  1. The User indicates he wants to authenticate into ReMeS.
  2. ReMeS asks him to provide his credentials (e.g. username and password).
  3. The User provides his credentials.
  4. ReMeS verifies the provided credentials and authenticates the User.
- **Alternative scenarios:**
  4b. The provided credentials were incorrect, resume at step 2.

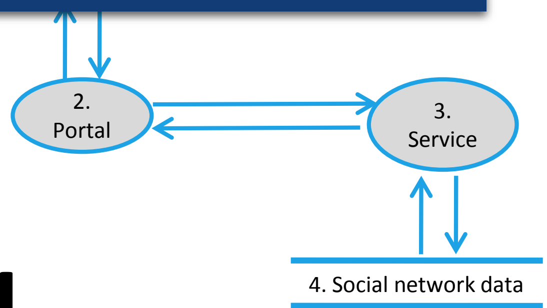


LINDDUN Threat modeling

- Eliciting threats
  - Related to **L**inkability, **I**dentifiability, **N**on-repudiation, **D**etectability, **D**isclosure of information, **U**nawareness, **N**on-compliance
- Model of the system highlighting the assets
  - Components (processing, data) and info flows
- Finding flaws that could lead to attacks
- "Not unlike" Microsoft's *STRIDE*

LINDDUN core steps

Model-based



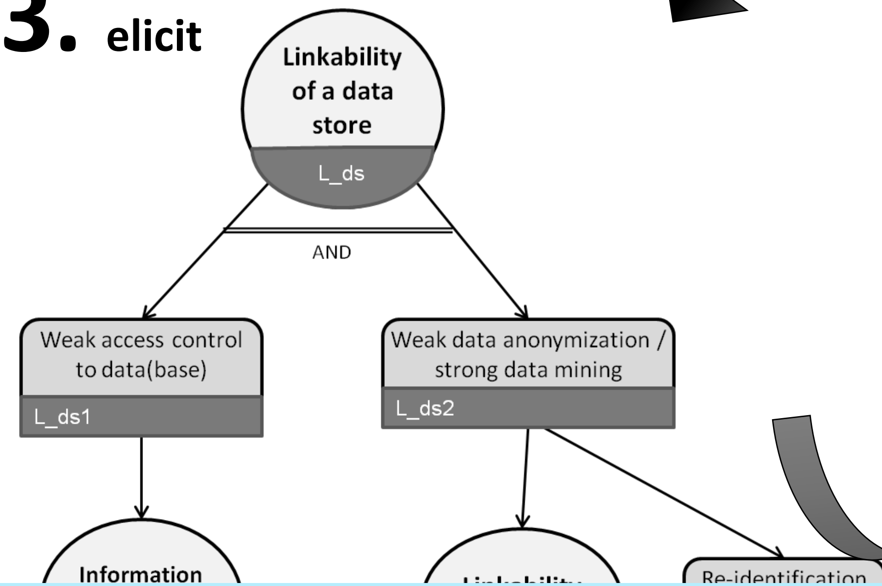
1. DFD

	Threat target	L	I	N	D	D	U	N
DS	Social network db	X	X	X	X			
DF	User data stream (user-portal)	X	X	X	X	X		
	Service data stream portal-service)	X	X	X	X	X		
	DB data tream (service – DB)	X	X	X	X	X		X
P	Portal	X	X	X	X	X		X
	Social etwork service	X	X	X	X	X		X
E	User	X	X				X	

	L	I	N	D	D	U	N
Data store	X	X	X	X	X		X
Data flow	X	X	X	X	X		X
Process	X	X	X	X	X		X
Entity	X	X				X	

2. map

3. elicit



Knowledge-based

MUC 04: linking data in DS

Summary: A researcher or other insider with malicious intent links PHR data (or user data)

Primary mis-actor: unskilled insider (authenticated user, e.g. researcher)

Basic path:

bf1. The misactor performs a set of targeted queries on the PHR data or user data store and retrieves very detailed results

bf2. The misactor links the results of the queries together (e.g. based on medication which is usually combined, medical conditions which occur together, or pseudo-identifiers like street and age)

Consequence: By combining the query results, the misactor has access to more information about the patient than anticipated

Reference to threat tree node(s): L_ds2, L_e2

Parent threat tree(s): L_ds, I_ds

DFD element(s): 5.1 PHR data, 5.2 user data

Remarks:

PUB

A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements

M. Deng, K. Wuyts, R. Scandariato, B. Preneel, W. Joosen, in Requirements Engineering 16 (1), 3-32, *2011*

LINDDUN in the wild

In privacy talks

RERUM

SEVENTH FRAMEWORK PROGRAMME

Applied in European projects

Use-cases definition and threat analysis

Editors:	Theodore Mouroutis, Athanasios Lioumpas (CYTA Hellas)
Deliverable nature:	Report (R)
Dissemination level: (Confidentiality)	Public (PU)
Contractual delivery date:	31 May 2014

BioMedBridges

Deliverable 5.3

Project Title:	Building data bridges between biological and medical infrastructures in Europe
Project Acronym:	BioMedBridges
Grant agreement no.:	284209
	Research Infrastructures, FP7 Capacities Specific Programme; [INFRA-2011-2.3.2.] "Implementation of common solutions for a cluster of ESFRI infrastructures in the field of "Life sciences"
Deliverable title:	Report describing the security architecture and framework
Actual delivery date:	30 June 2014

Contents

...ed by independent researchers

Ad Interim - Summary of evaluation

- Advantages
  - Acceptable **correctness** rate
  - Relatively **easy to learn** and apply
  - LINDDUN threat tree **catalog** is useful
  - Good **coverage** of privacy threats
- Room for improvement

Study 1 + 2
Reqs + archi

Study 3
experts

Illustration 2: Source Code analysis – vulnerability prediction

impact of software quality on security

- Specialists: verification technology
  - *Direct assessment (A)*
- For any developer
  - *Indirect assessment (B)*

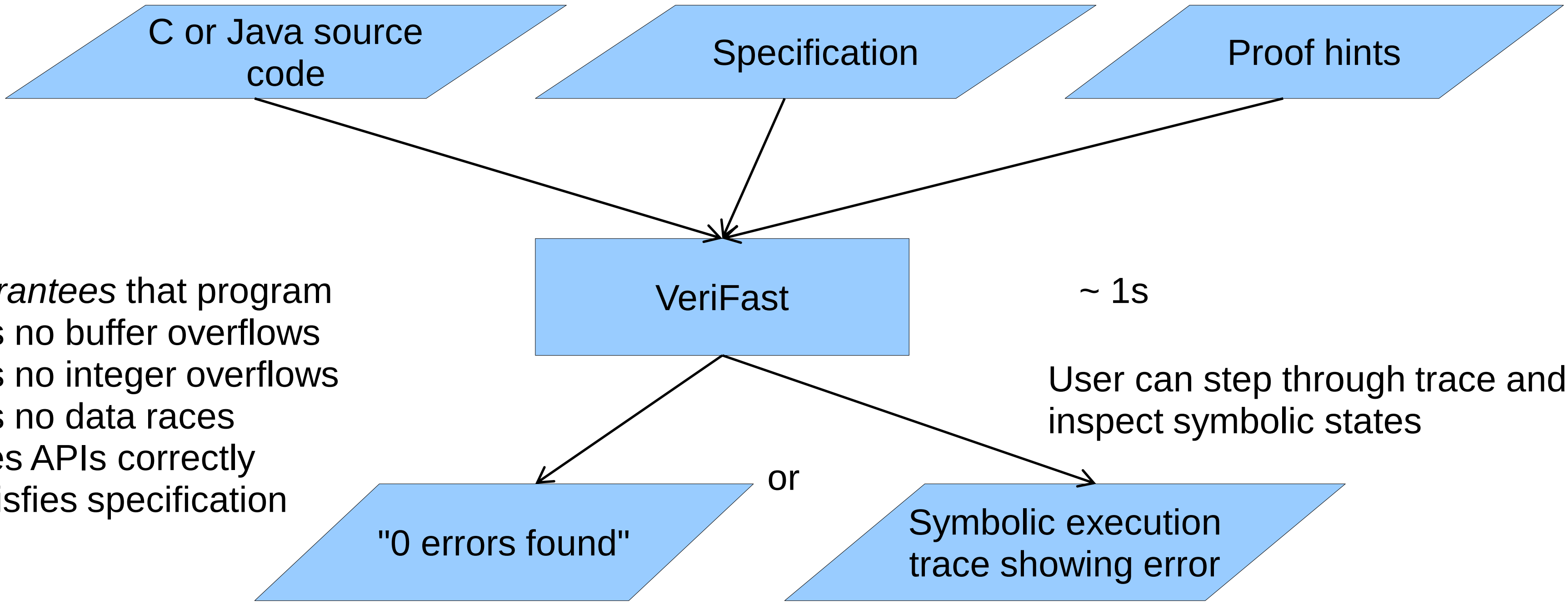
<A> VeriFast

Software Quality @ Development time

DistriNet

iMinds KU LEUVEN

VeriFast



Guarantees that program

- has no buffer overflows
- has no integer overflows
- has no data races
- uses APIs correctly
- satisfies specification

VeriFast: verified programs - cases

- Fine-grained concurrent data structures
  - Functional correctness
- JavaCard applets (incl. for Belgian eID card)
  - Crash-freedom, safe API usage
- Linux device drivers
  - Memory safety, data-race-freedom, safe API usage
- Embedded software (for Telefonica home gateway)
  - Memory safety, data-race-freedom, safe API usage
- Cryptographic protocol implementations (RPC, Needham-Schroeder-Lowe)

<B> Fault Prediction, based on Text Mining

Software Quality @ Development time

DistriNet

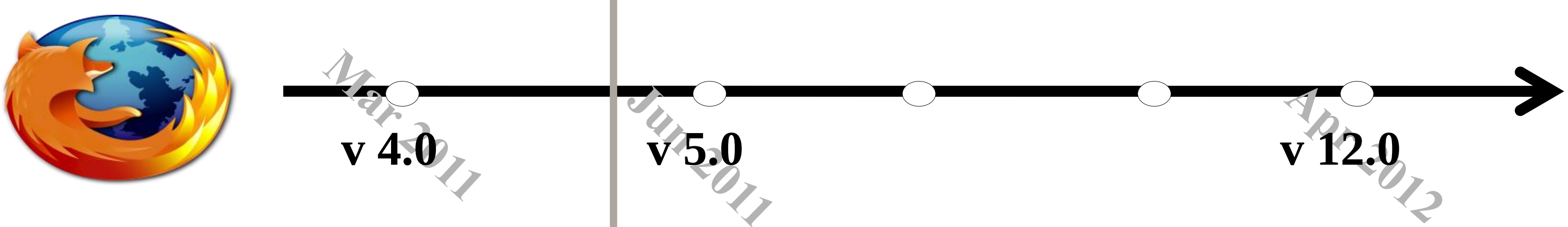
iMinds KU LEUVEN

Research question

Can we build a (good quality) classifier that predicts vulnerable components in C++ applications?

Idea: Analyze the tokens in each component's code (e.g., if, while, variable names) and use these as predictors

Prediction in the future



Build prediction model (using 1 version)

Test performance of prediction model (in each of the following 8 version)

Benchmark

- Find at least **80%** of the components containing vulnerabilities (*cost*) by inspecting at most **20%** of the application components (*benefit*)

Results

- We exceedingly meet the benchmark
  - For all the “future” versions

- Better than best results in the state-of-the-art (i.e., Shin et al., TSE 37(6), 2011)

And now...
Reaching out!

Which problems are perceived to be of the highest priority?

We start an anonymous survey of ISV's in Flanders and beyond (Q2 2015)

Challenges Summarized

Full life cycle support must become agile, but it remains *high priority*. (Part 2)

This cannot be achieved without managing the concept of *risk*

New techniques can and should contribute to *reducing the overall cost.*

This must be pursued while dealing with *all other trends* of these interesting times….(Part 1)

Thank *You!*

Thank *them* !

Jasper Bogaerts, Maarten Decat, Ming Deng, Philippe De Ryck, Lieven Desmet, Thomas Heyman, Aram Hovsepyan, Bart Jacobs, Bert Lagaisse, Fabio Massacci, Sam Michiels, Jasper Moeys, Frank Piessens, Bart Preneel, Davy Preuveneers, Riccardo Scandariato, Steven Van Acker, Dimitri Van Landuyt, James Walden, Kim Wuyts, Koen Yskout, …