POLITECNICO
DI MILANO





Masibty

Stefano Zanero, Claudio Criscione

Who's who

- Stefano Zanero
  - Assistant Professor @ Politecnico di Milano
- Claudio Criscione
  - Principal Consultant @ Secure Network
  - Hopefully soon-to-be PhD student @ Politecnico di Milano

What is our speech all about?

It's about letting people in charge of web applications security sleep at night*

* terms and conditions apply. We do not take care of your partner snoring

- Difficult, IRW to
  - **Detect attacks**
  - **Apply patches (without support from developers)**
  - **Have the time to follow all those 2458 unitasker web applications**
- In the meantime, you're likely going to get hacked by a pack of Monkeys (which can successfully hack web application, as scientifically demonstrated)

Web application IDSs and IPSs (so far)

- Web Application Firewalls – a must?
  - Patching is not always possible due to “obscure reasons”
  - *Application* and *infrastructure/security* are different departments
  - You just have to do “something” for web application security, and you have to do that **yesterday**
- Most WAF solutions suffer from the “Grep Dilemma”
  - Should I really use something which is little more than a complex **Grep**?

Why signatures are bad

- Inherent issues with signature based systems!
  - Application of blacklisting, and we all know blacklisting is intrinsically flawed
  - “Things that you do not hope for happen more frequently than things that you do hope for” (Plauto, “Mostellaria”)
  - You cannot enumerate all the possible attacks, and “generic signatures” yadda yadda simply do not work nearly well enough
- Applying whitelisting (i.e. only allowing through what is supposed to go through) would work, but it is a configuration nightmare
  - List **every** parameter of **every** form on **every** page of **every** application on **every** server
  - And then we can discuss “change management”, folks...
- This is why WAFs require careful configuration and constant updating
  - And **time** and **skills** are scarce resources, as usual

What are we trying to do?

- Recreate the "**Old Lady at the Window**" effect
  - You know, the old lady spotting "strange things happening" and dialing 9-1-1
- Which means...
  - Learning what's normal: Whitelisting : **Anomaly detection**
  - Block what's not: **Intrusion prevention**
  - Without administrator intervention : **Unsupervised learning**
  - With no (well, just a few) false positives
  - With attacks in the learning set – because that's what happens in the real world!

So, what is Masibty?

- A web application IPS
  - **Anomaly based**, and capable of doing unsupervised learning
  - Able to work in the “real-world”
  - Partly language-indipendant (Java **reverse proxy**) and partly language dependant (PHP PoC)
  - A flexible **architecture** where modules can be plugged into

Basic ideas

- What are we going to learn?
- How are we going to learn it?
- How are we going to use it?

What are we going to learn?

We have a name for that
Entry Point

- URI
- Parameters
- Session
- The ubiquitous external influence

Finding structure in entry points

- The first challenge: how do we identify Entry Points?
- **Online multimodel n-dimensional agglomerative approximate clustering algorithm**
  - Which we had to design
- Multiple models to identify **behaviors**
  - Parameters order, presence, type, names...
- We evaluate a **distance** between various queries on the same “URL”
- We end up with an “identifier of homogeneous input parameters”, which we assume is homogenous behaviour

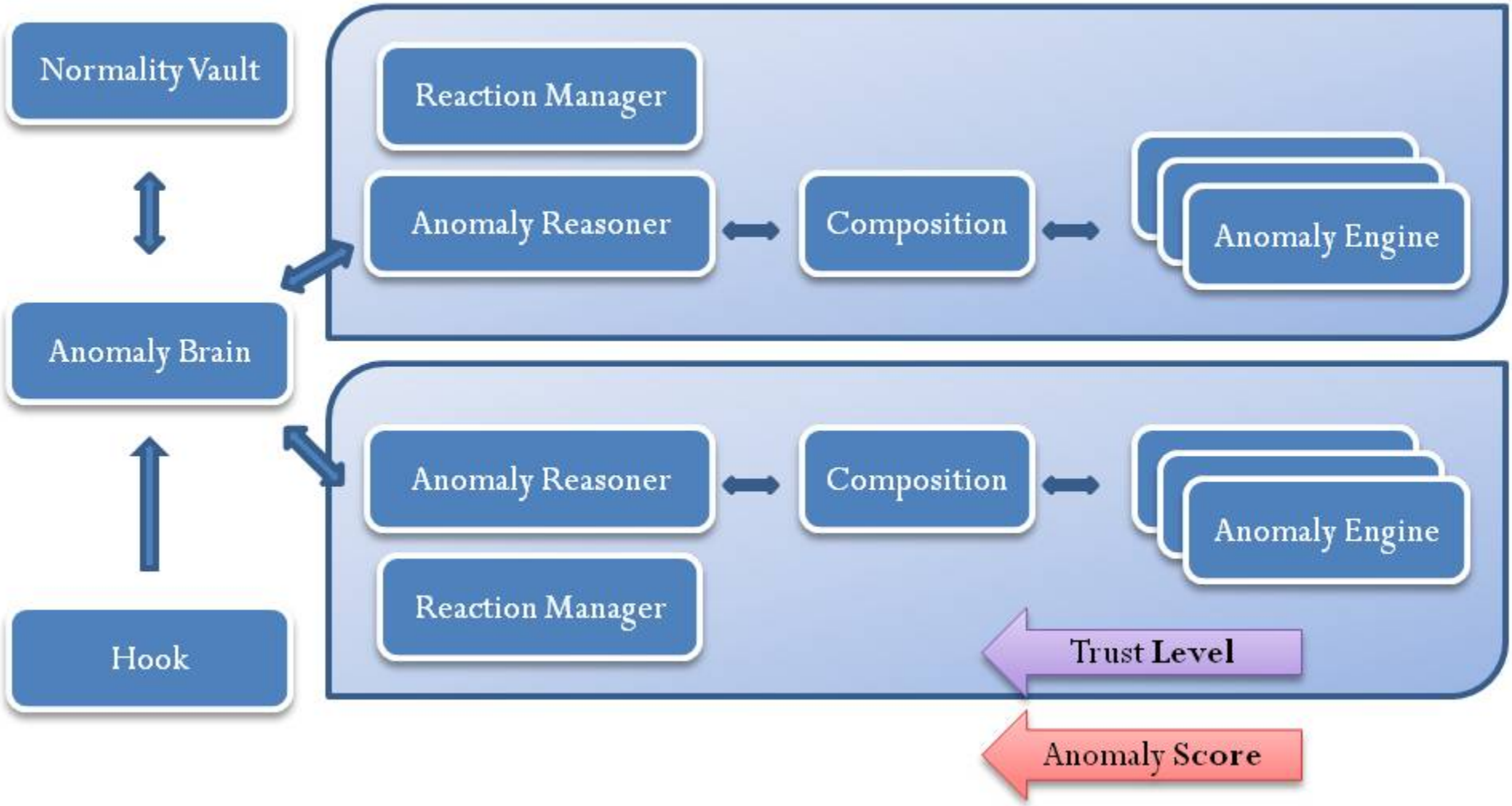
To clarify...

controller.php?
cmd=list_users&page=1

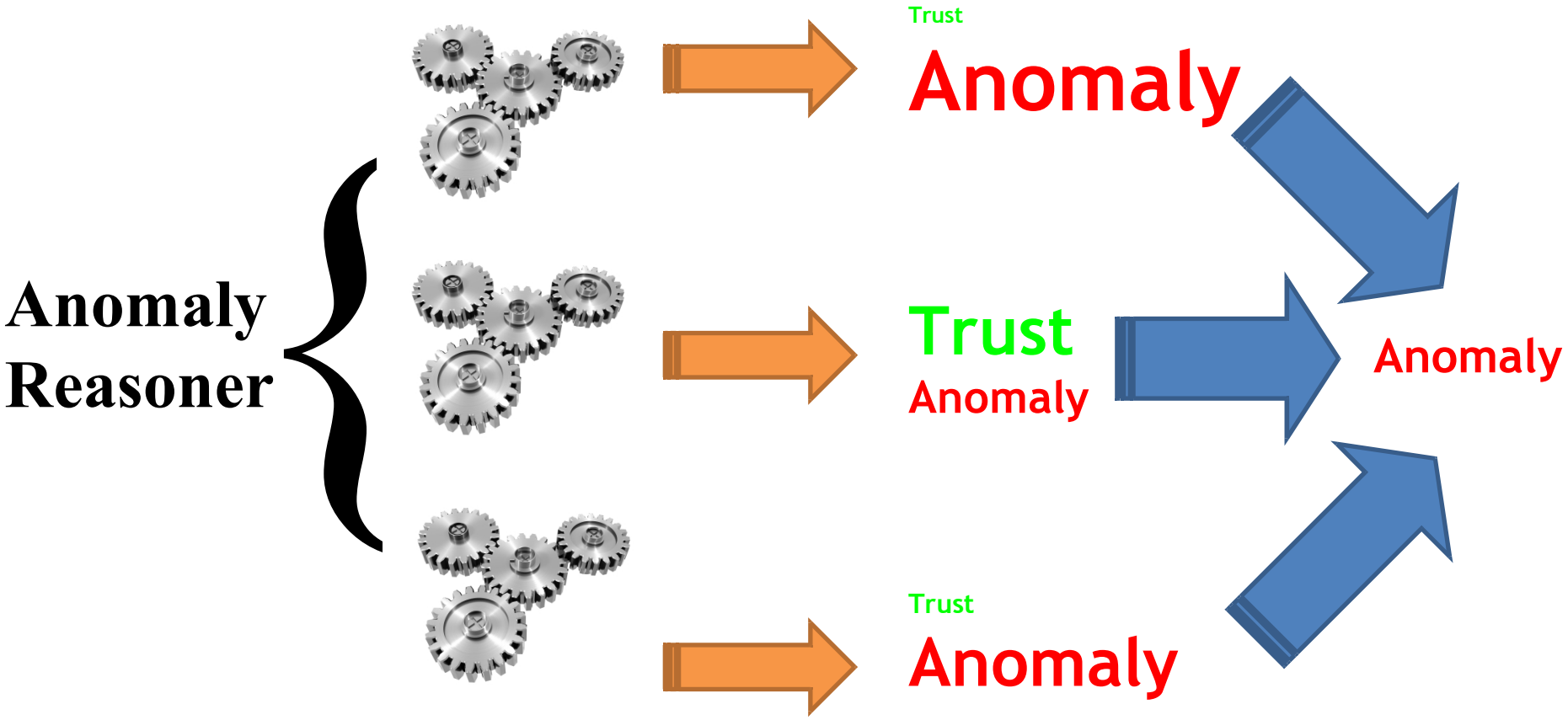
controller.php?
cmd=view_product&onWebsite=yes

controller.php?
cmd=view_product&pid=20&onWebsite=no&accessible_mode=on

How are we going to process the data?



Anomaly and Trust



Parameter Anomaly

- For each parameter, we build a profile using various **engines**
  - Order Engine
  - Presence Engine
  - Numbers Engine
  - Aliens Engine
  - Token Engine
  - Distribution Engine
  - Length Engine
- You can notice similarities with other models (like the ones proposed by Vigna and others)
  - We have improved some of their models or rebuilt them according to our new requirements

Content Engines

- Some of the engines take care of the “values” of the Parameters
  - **Number engine**: if we put a non-numerical value in an “almost always” numerical attribute, we get an anomaly
  - **Token Engine**: some parameters can only assume predefined values. They're *Tokens.*
  - **Length Engine**: parameters usually have a “similar” size
  - **Distribution Engine**: we should be able to identify notable peaks in the usage of a single character
  - **Alien Engine**: most parameters won't accept EVERY printable character

Structural Engines

- Web applications often are “regular”, parameters are usually in the same order
  - **Order Engine**
- ...and you usually have the same parameters on the same Entry Point
  - **Presence Engine**
- Most structural engines can be bypassed, but are very accurate against many automated attacks!

Client side attacks

- We now have a broad range of tools to identify attacks aimed at the server
- But yet, during the coding of Masibty, we wondered

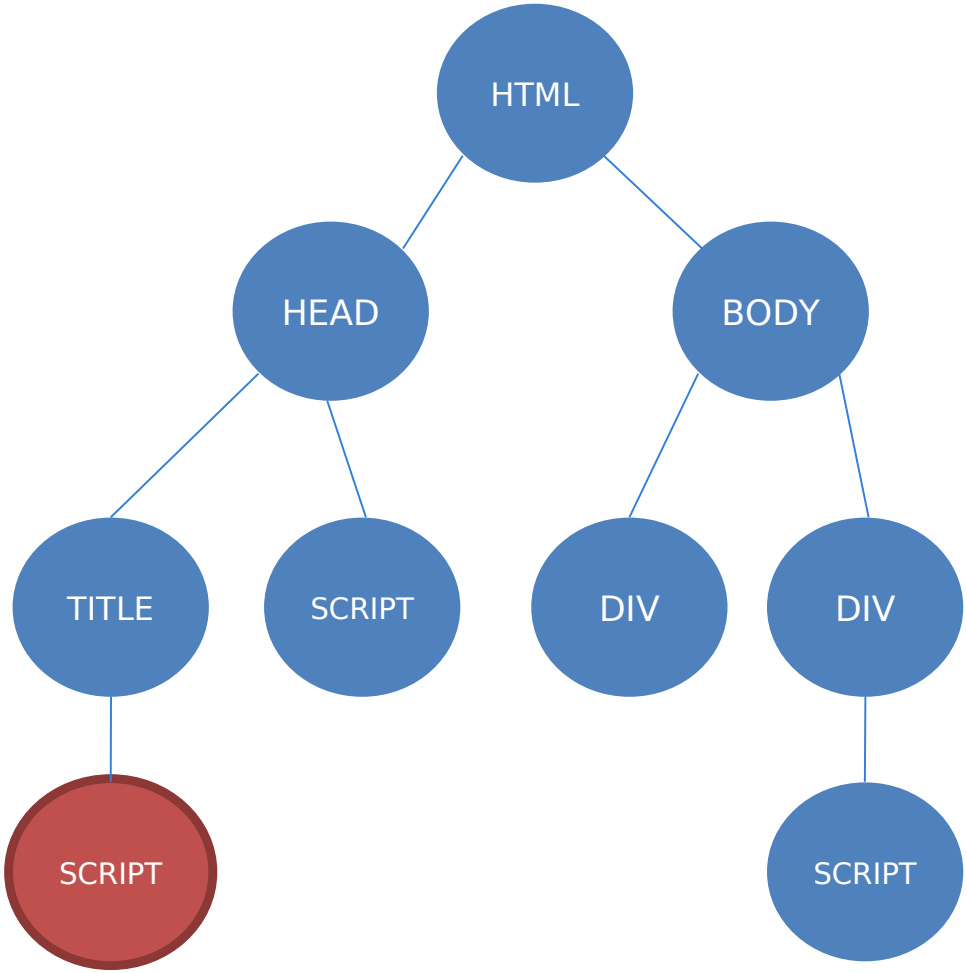
"Since we already see all of these server responses, why don't we analyze those as well?"

Anomaly Trees

- Build a representation of server responses
  - Plant a (DOM) tree, save the environment!
- Once we have generated the tree, we can “learn” it
- If we see at some point in the future an unexpected branch on the tree...

Anomaly Trees

```
<HTML>
 <HEAD>
  <TITLE>
  <script>attack</script>
 </TITLE>
  <SCRIPT>JS</SCRIPT>
 </HEAD>
 <BODY>
  <DIV> TEST 123 </DIV>
  <DIV>
   <SCRIPT>JS</SCRIPT>
  </DIV>
 </BODY>
</HTML>
```



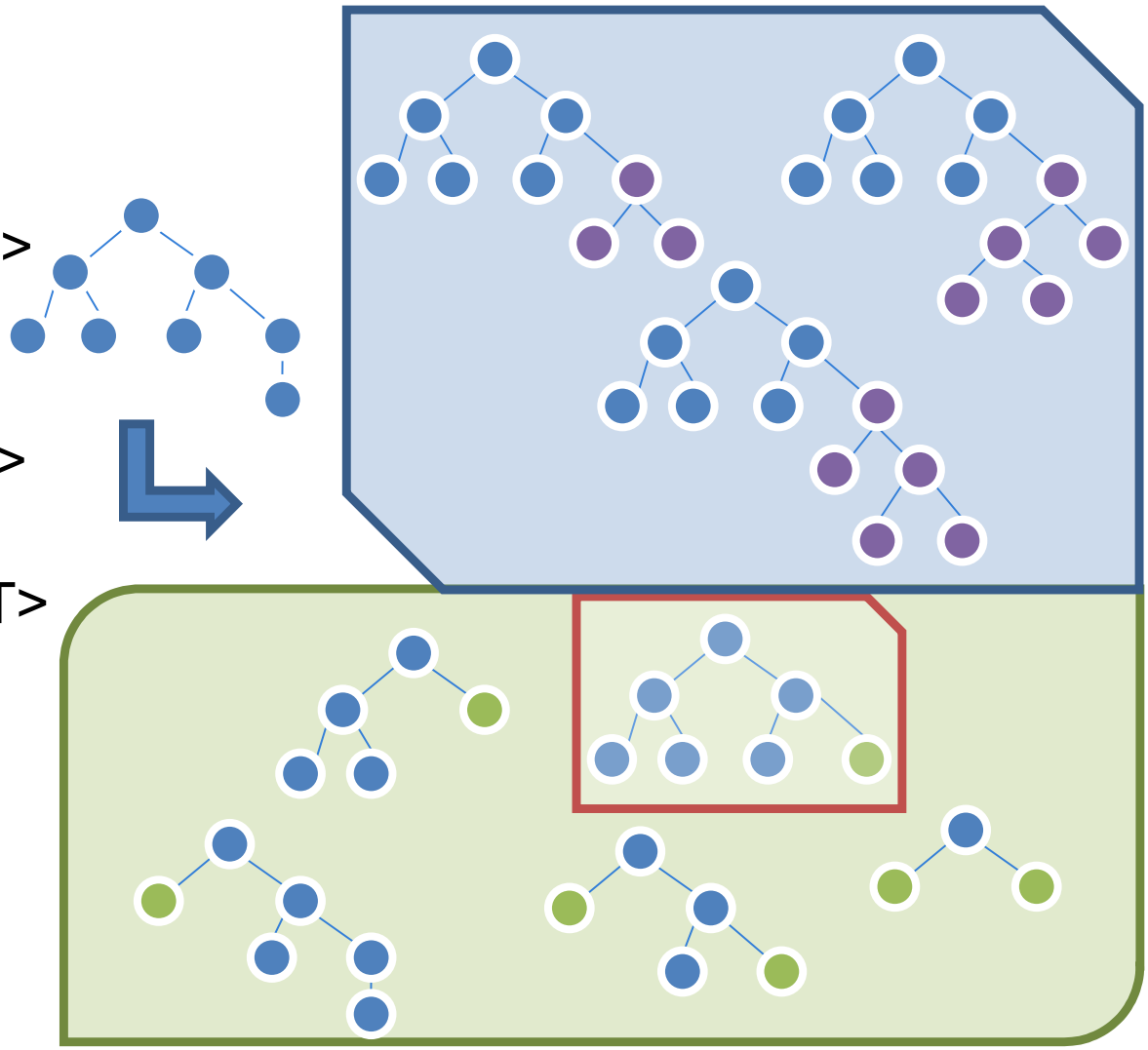
Growing trees in different shapes

- A trivial “difference” between trees would be very false-positive prone
  - And would cause a lot of issues on each update

- **Templates** : identify areas of the tree were new branches are more likely to happen.

Building templates

```
<HTML>
 <HEAD>
  <TITLE></TITLE>
  <SCRIPT>JS</SCRIPT>
 </HEAD>
 <BODY>
  <DIV> TEST 123 </DIV>
  <DIV>
   <SCRIPT>JS</SCRIPT>
  </DIV>
 </BODY>
</HTML>
```



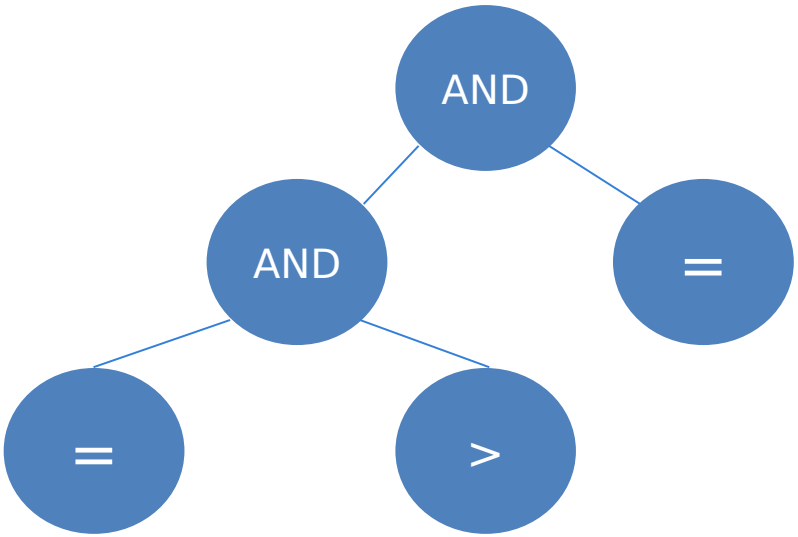
Parsing

- 2 issues
  - Are we looking at the **SAME** tree the user would see?
  - We only care about **JavaScript**
- **Gecko!**
- We build the DOM tree as the browser would do it
- We can ask Gecko where the javascripts lie
  - So we only have *meaningful* branches in the trees

Oh no, more trees! SQL Anomaly

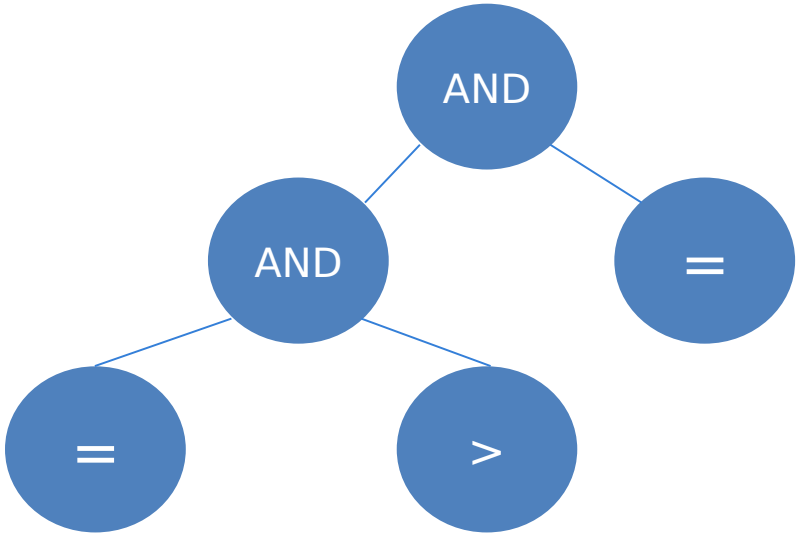
- Once we had Anomaly Tree algorithms working reliably on DOM documents, it was “easy” to port them on SQL
- Each SQL query can be represented as a tree
  - We can spot changes in the tree as we've done with the XSS Reasoner

```
SELECT * FROM USERS WHERE NAME = 'USER' AND
   (PASSWORD = 'PASS'  AND ROLE > 0)
```

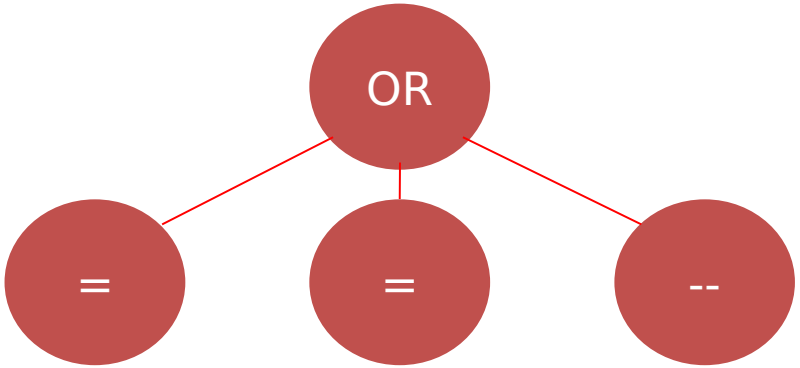


SQL Trees

```
SELECT * FROM USERS
    WHERE NAME = 'USER'
    AND ( PASSWORD = 'PASS'
    AND ROLE  > 0)
```




```
SELECT * FROM USERS
    WHERE NAME = 'USER'
    OR '1'='1' -- AND
    (PASSWORD = 'PASS' AND
    ROLE > 0')
```



Can we avoid the webocalipse?

- Evaluating the performance of an IDS isn't an easy task
- We tested 7 “real” applications
- A simple methodology
  - Install the application
  - Use the application “through Masibty” as normal users would do
  - Add some attacks during “learning”, either background noise like worms or real, successful attacks to the application
  - Switch to detection and repeat the tests
- Excellent (if not conclusive) results
  - 84% detection rate with a modest 0.14% false positive rate
  - Which gets to 93% DR if we take Badstore (yes, we've tested that one too) out of the pool
  - And gets to 100% DR, 0% FP if we remove the attacks from the training set...
    which is what everybody else does!

How slow is it?

- Codebase is not optimized
  - No really, it's just a PoC for now, blame Claudio :-)
- In our testing environment we got an average 4-50ms delta in response times during the training phase and 1-20 ms during the detection phase
- RAM and CPU usage were usually quite low – and it was running in Eclipse!
- More testing is on its way

How can I get it? and future works

- It is going to be released for testing
  - And hopefully we'll have a paper on that sooner or later
- We're building a *working* GUI
- Next steps include
  - Supervised learning addon
  - New dedicated reasoners (JSON, Flash, Headers...)
  - Some advanced agent based stuff

Thank you!

Questions!?!?

stefano.zanero@polimi.it
c.criscione@securenetwork.it