Scriptless Attacks

Stealing the Pie without touching the Sill

Mario Heiderich, Felix Schuster, Marcus Niemietz,
Jörg Schwenk, Thorsten Holz
ACM CCS 2012

HGI / Chair for Network and Data Security
Ruhr-University Bochum
mario.heiderich@rub.de || @0x6D6172696F

hgi
Horst Görtz Institut
für IT-Sicherheit

RUB

CURE53

Our Dear Speaker

- **Dr.-Ing. Mario Heiderich**
  - Researcher and Post-Doc, **R**uhr-**U**ni **B**ochum
    - PhD Thesis on Client Side Security and Defense
  - Founder of Cure53
    - Penetration Testing Firm
    - Consulting
    - Simply the Best Company of the World
  - Published author and international speaker
    - Specialized in HTML5 and SVG Security
    - JavaScript, XSS and Client Side Attacks
  - HTML5 Security Cheatsheet
    - @0x6D6172696F
    - mario@cure53.de

hgi Horst Görtz Institut für IT-Sicherheit RUB CURE53

Background

hgi
Horst Görtz Institut
für IT-Sicherheit

RUB

CURE53

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, Tripwires, Browser XSS Filters
    - mod_security, PHPIDS, some (often nonsense) WAF products
- **But why use scripting at all?**

Topics Today

- **Scriptless Attacks in your Browser**
  - Attacks bypassing NoScript
  - Attacks bypassing XSS Filters
  - Attacks bypassing **C**ontent **S**ecurity **P**olicy
- **Thought Experiment**
  - What if we defeat XSS?
  - What attack surface will remain?
  - Will it make a difference?

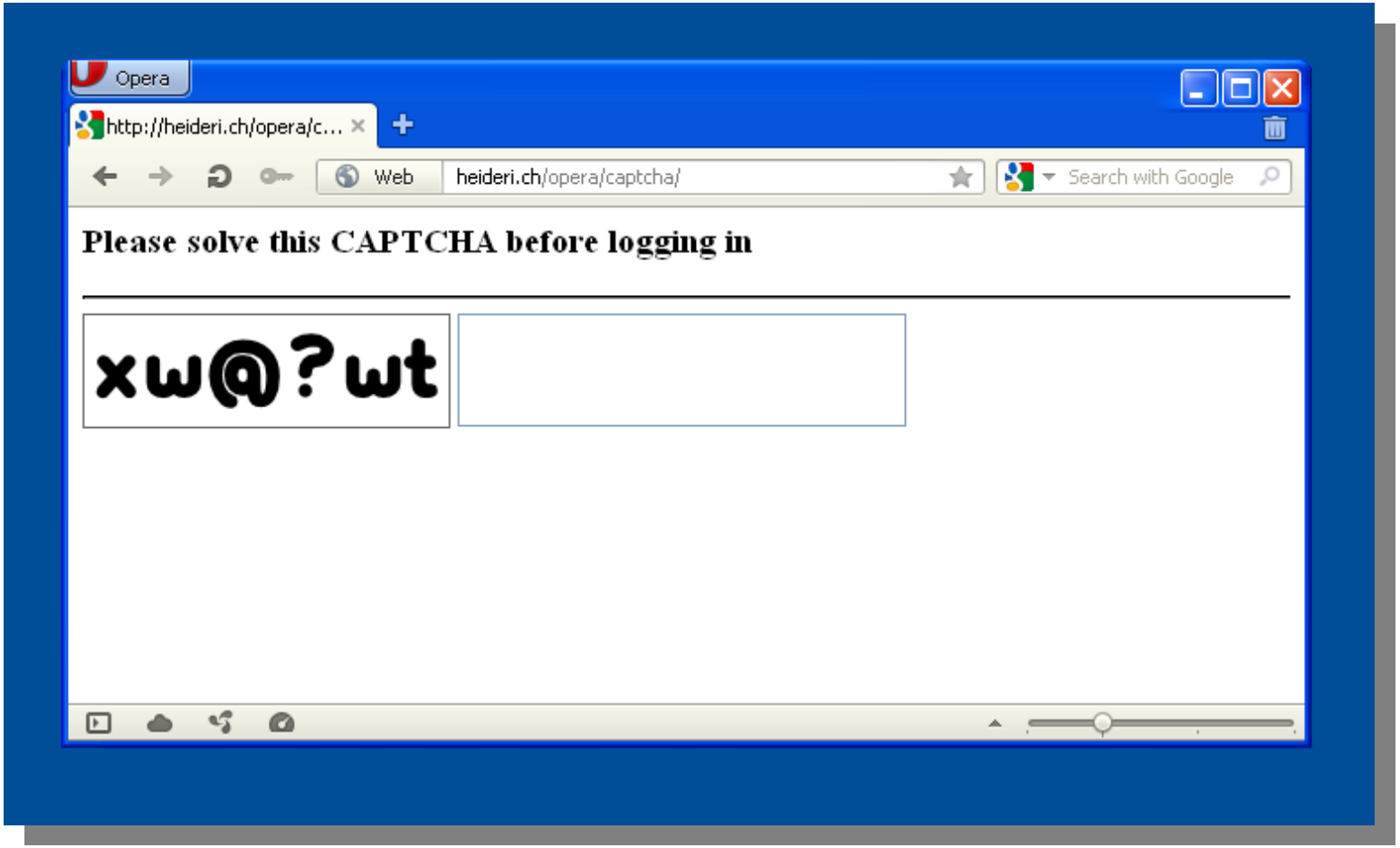
Happy Injections

Exploits

- Three Chapters to be presented
  - Chapter 1: **These simple tricks**
  - Chapter 2: **Advanced Class**
  - Chapter 3: **For Science!**

Chapter one

CAPCTHA Of Doom



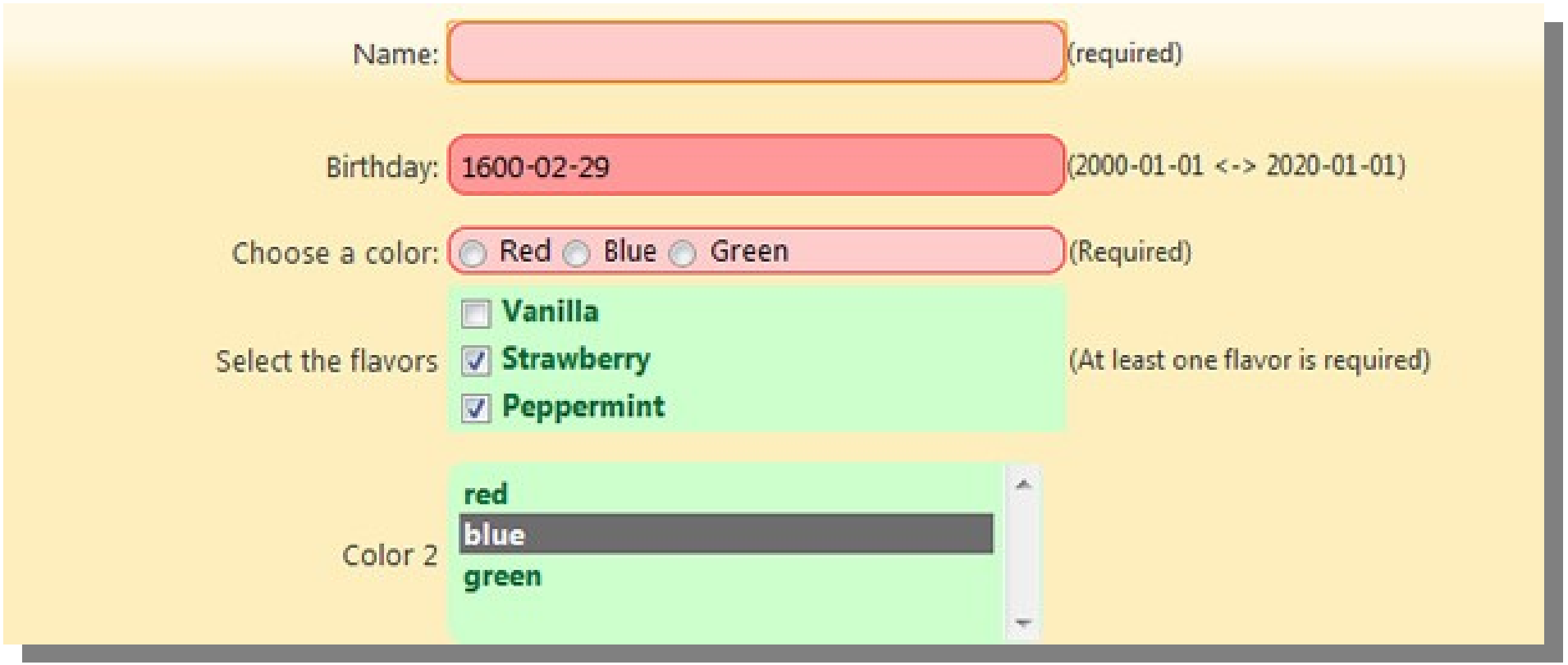
- Seems legit?
- See it live: **http://heideri.ch/opera/captcha/**

hgi Horst Görtz Institut für IT-Sicherheit
RUB
CURE53

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**

Validation



CSS + 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**

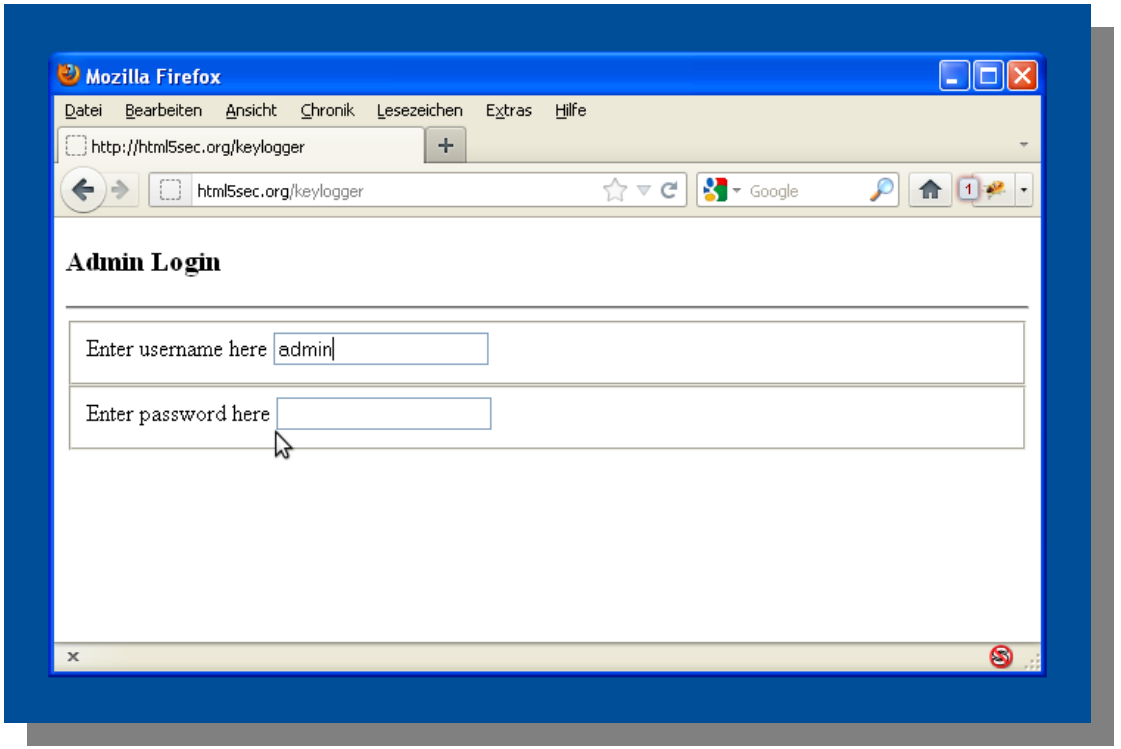
- **Good thing it works on all browsers**
  - Limited by smart password managers though

hgi Horst Görtz Institut für IT-Sicherheit RUB CURE53

Chapter TWO

SVG Keylogger

- Just a harmless login page



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

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**

CSS + URL + Regex = ?

- More info we can steal
- CSS3 and `@document`
- Allows to cast a Regex on the loaded URL
- **Then deploy custom CSS**

- We can steal stuff now
- But we can do even more ;)
  - **http://html5sec.org/xssfilter/**
  - **Is that all?**
  - **Maybe not**

More Madness

- HTML5's `dirname` attribute
- The most useless attribute ever
- Worse than `formaction`... which one should know or look up :)

- Meant to tell the server about...

  drumroll
- **Text-Flow Direction!**
- Also does cruel things to HTTP requests
  - **DEMO**

Chapter Three

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

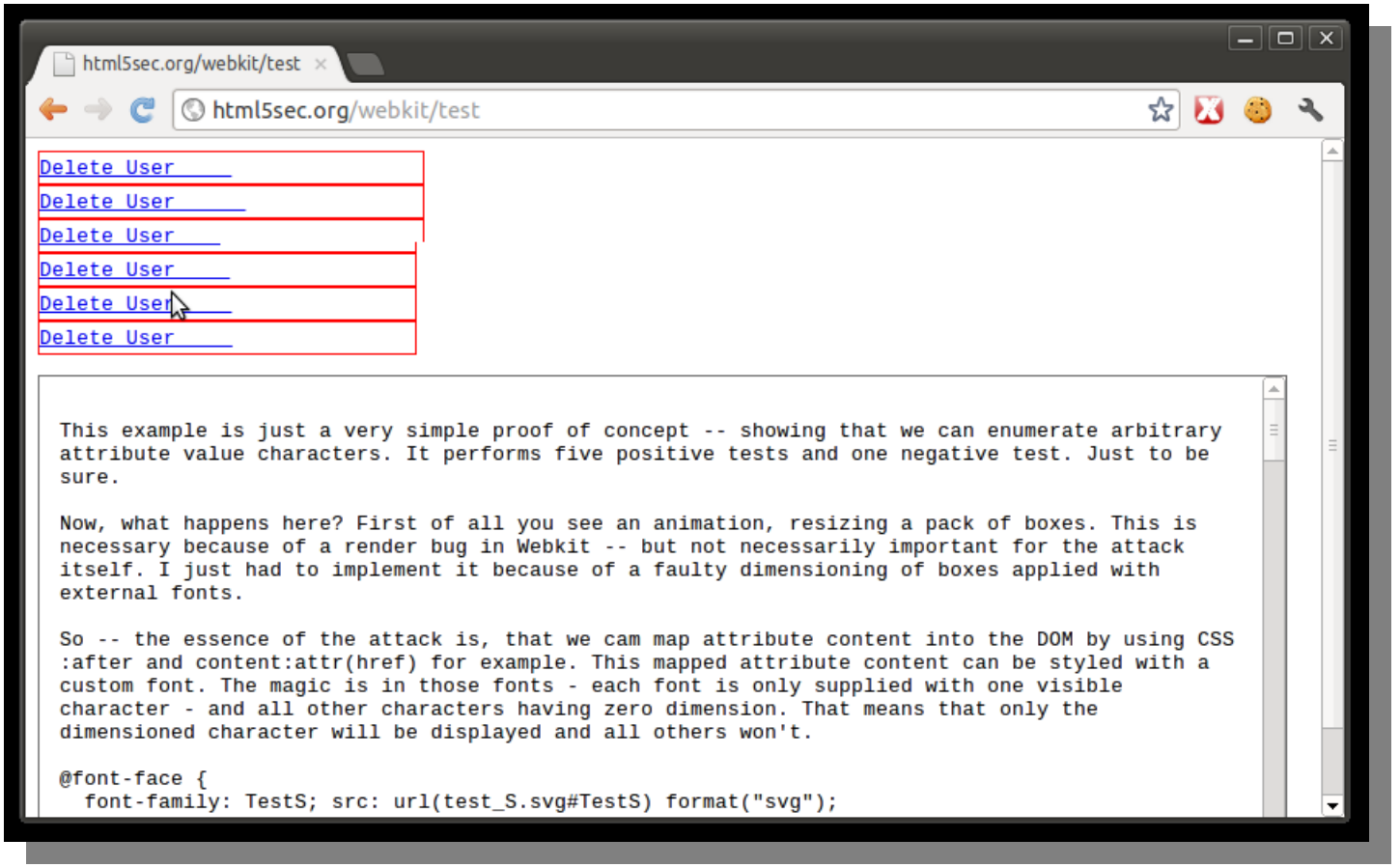
- SDC, Gaz and thornmaker already did it
- Check out http://p42.us/css/
- They used CSS
  - Basically a brute-force via attribute selectors
  - `input[value^=a]{background:url(?a)}`
  - If the server catches GET /?a...
  - The first character is an **a**
- But then what?
- There's no „second or Nth character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

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



The Magic Part

- 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**

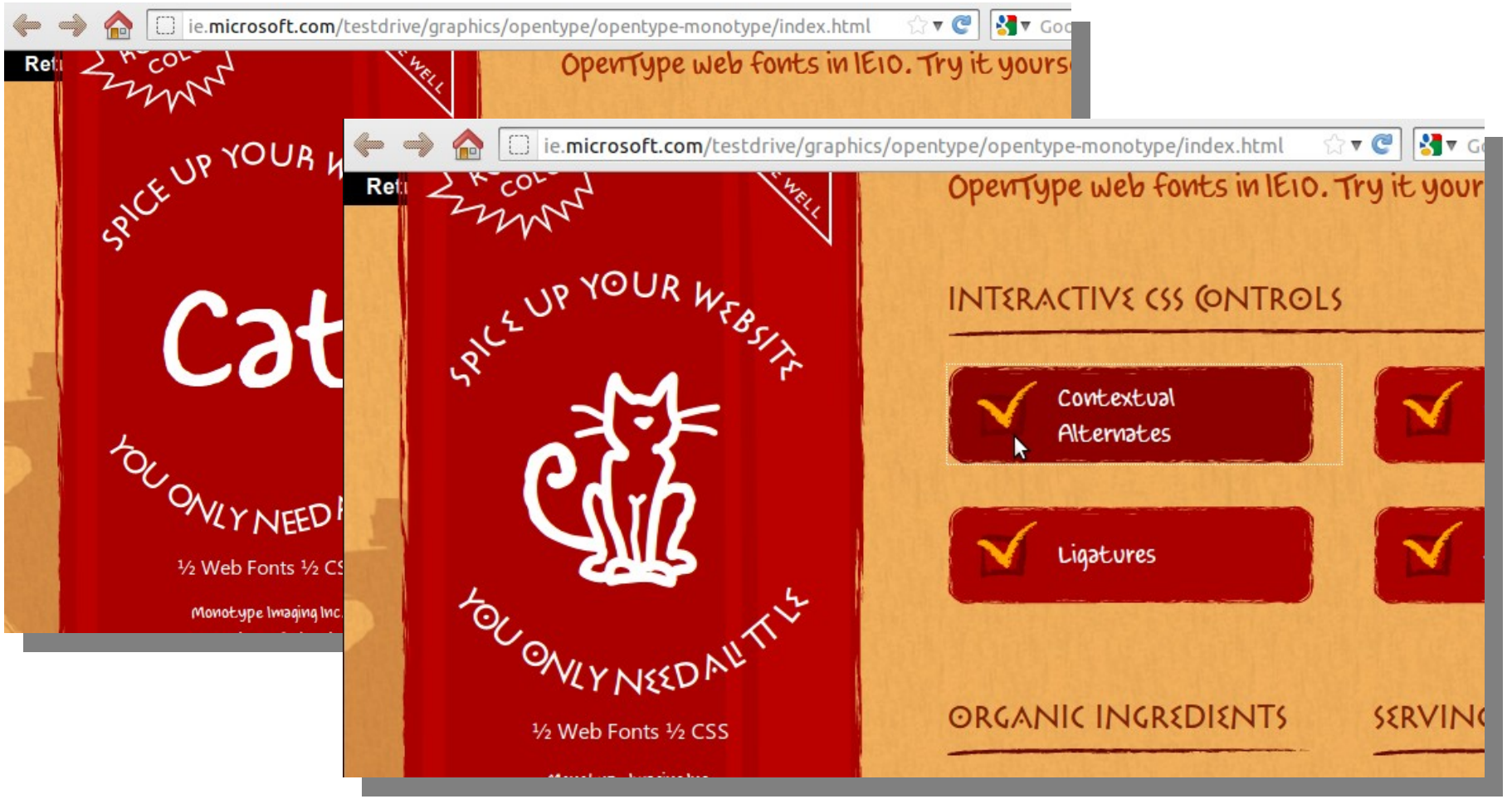
```
div.s::-webkit-scrollbar-track-piece
:vertical:increment {background:red url(/s)}
```

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*

Ligatures



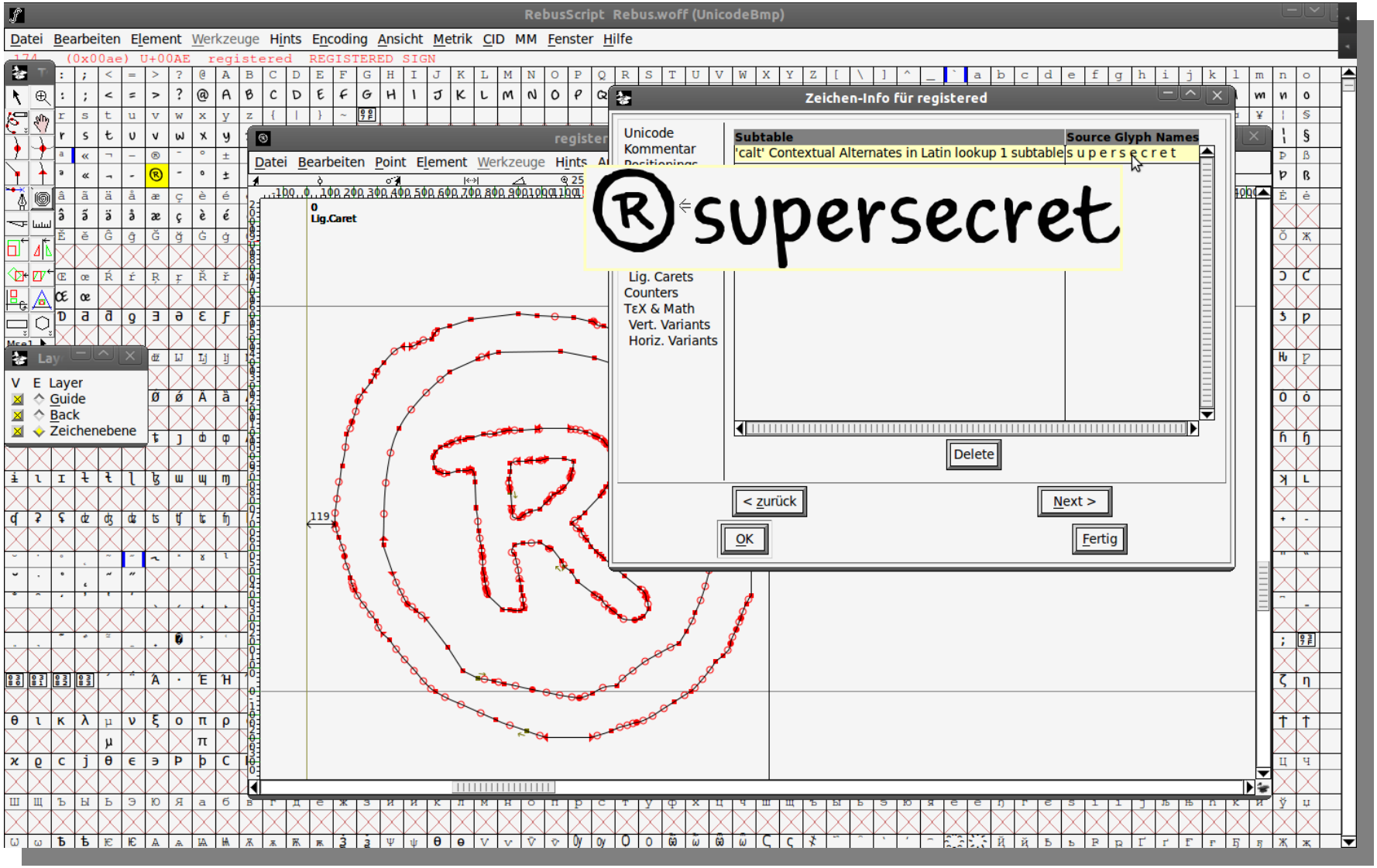
- http://ie.microsoft.com/testdrive/graphics/opentype/opentype-monotype/index.html

hgi Horst Görtz Institut für IT-Sicherheit

RUB

CURE53

Fontforge



Horst Görtz Institut für IT-Sicherheit

RUB

CURE53

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

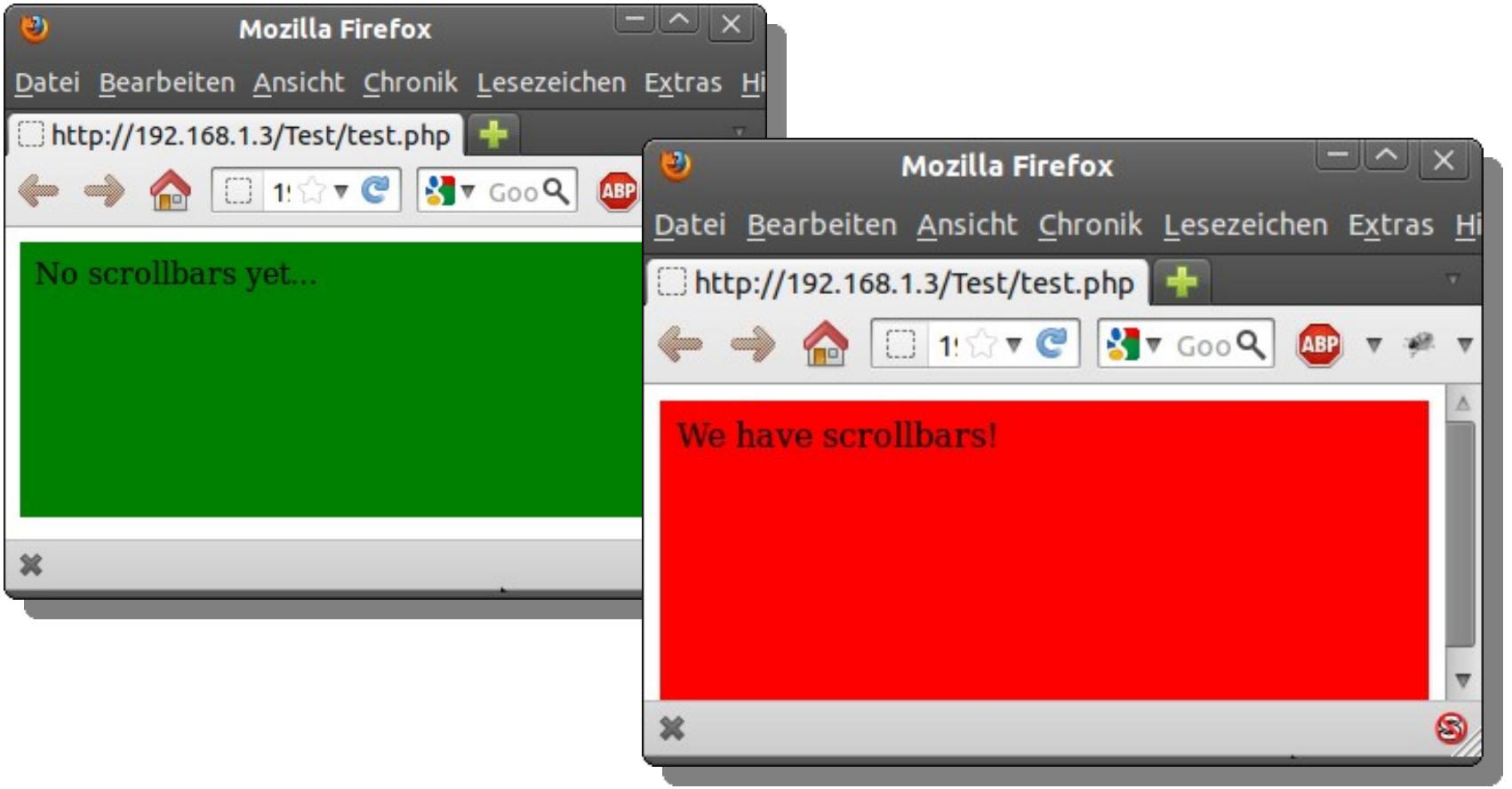
- **How can we find out if nothing – or just one character is visible?**

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**

Demo



DEMO

hgi Horst Görtz Institut für IT-Sicherheit

RUB

CURE53

Conclusion

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

Defense

- How to protect against features?
- How to protect against side-channels
  - Reduce data leakage?
  - Build better sandboxes?
  - Extend SOP to images and other side channels,
  - CSP maybe? One day?
- XFO and Frame-Busters
- Better CSS filter tools are needed!
- **Know your spec, contribute!**

Fin

- Questions?
- Discussion?
- Please read our Paper and...
- **Thanks for your time!**

hgi Horst Görtz Institut für IT-Sicherheit

RUB

CURE53