New Kid on the Web: A Study on the Prevalence of WebAssembly in the Wild

Marius Musch

TU Braunschweig

Together with Christian Wressnegger, Martin Johns, and Konrad Rieck

The native Web

Previous attempts at native performance

- Adobe’s Flash
- Microsoft’s ActiveX
- Google’s Native Client

asm.js

- Subset of JavaScript with special optimizations
- Type consistency and manual memory management
- Faster execution, but parsing still slow

WebAssembly (Wasm)

Introduced March 2017

- Supported by all major browsers, even on iOS and Android
- Faster transmission, parsing and execution than JS

Low-level bytecode language

- Standardized, platform-independent
- Executed in stack-based virtual machine

=> Compile any LLVM-supported language to the Web

Using Wasm modules

```
const obj = {
  imports: {
    imported_func: function (arg) { console.log(arg); }
  }
}


const wasm = await WebAssembly.
    instantiateStreaming(fetch('example.wasm'), obj);


let result = wasm.instance.exports.factorial(13);
```

WebAssembly in the Wild

Prevalence

Data collection

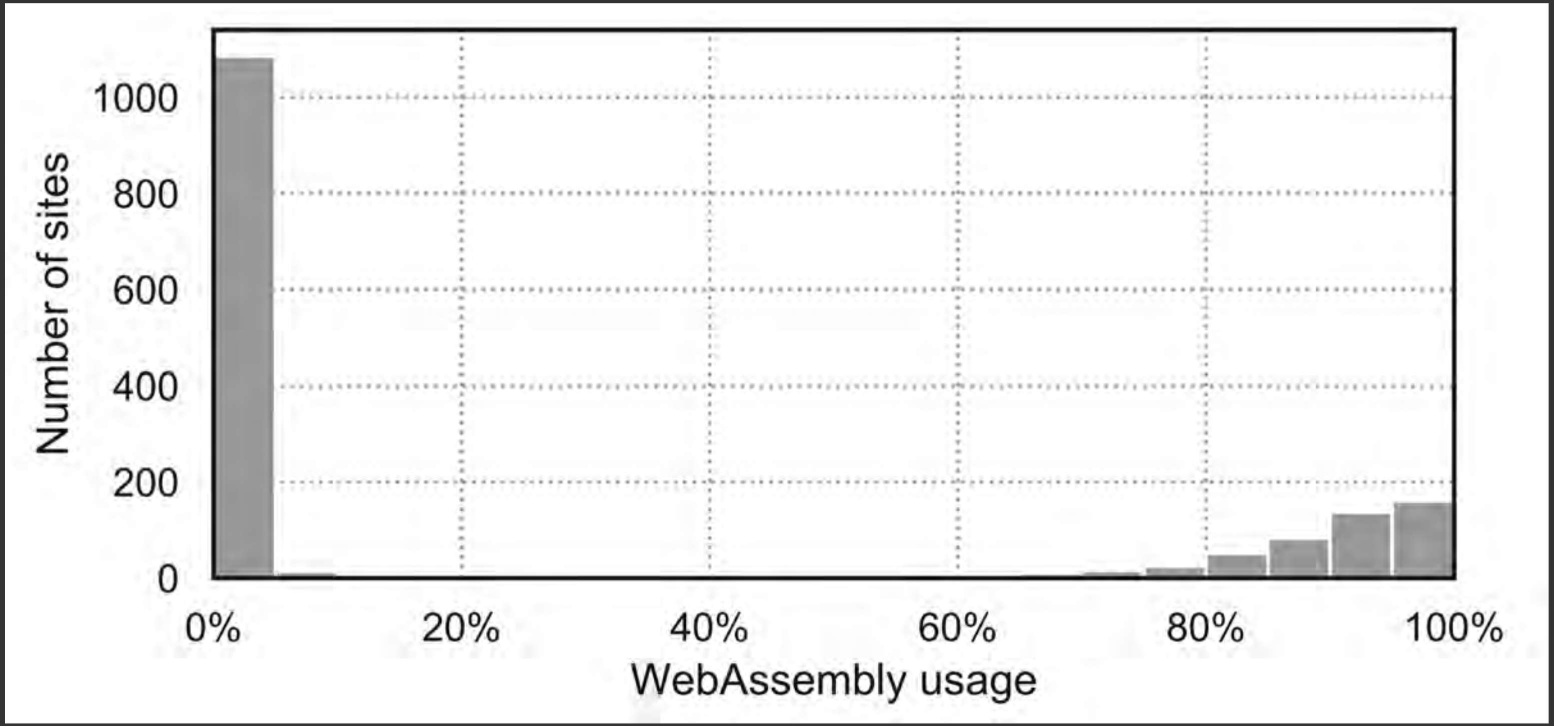
- Alexa Top 1 million sites + three random subpages
- In total about 3.5M pages

1950 Wasm modules on 1639 sites

- 150 unique samples
- Most popular module: On 346 sites
- Only seen once: 87 modules

Extent of usage

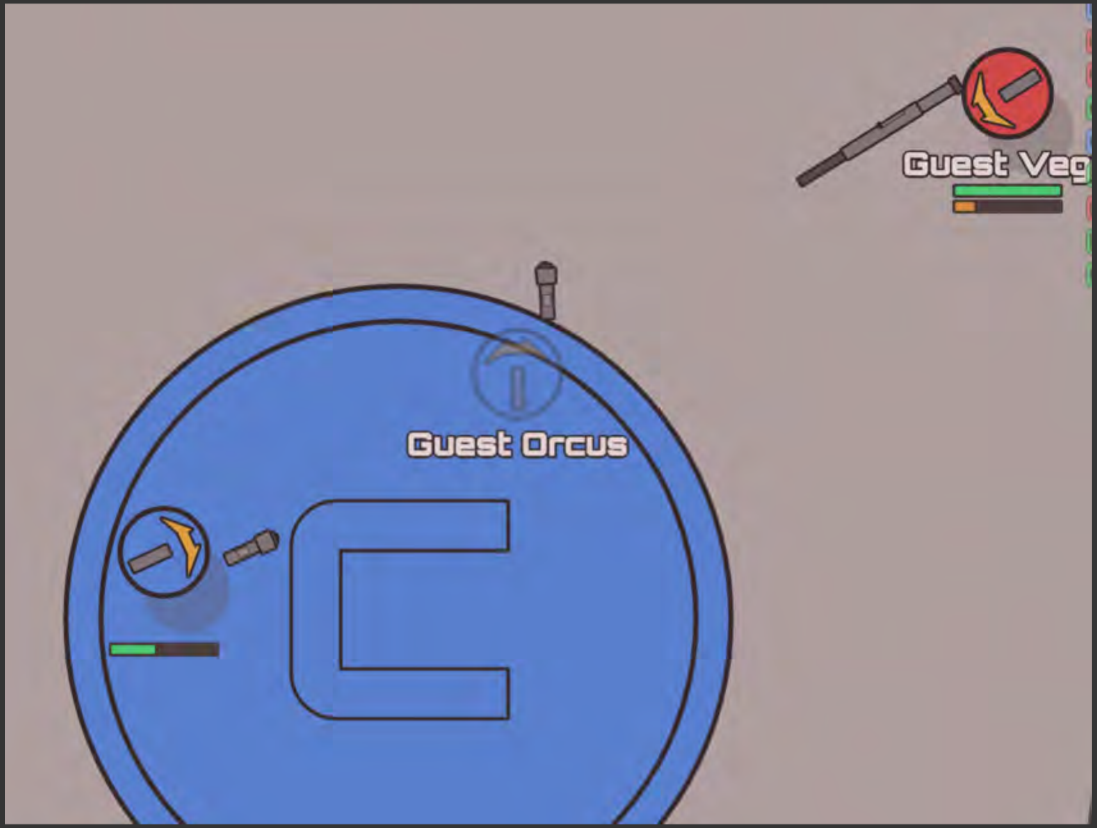
- 8 bytes – 25.3 MB module size
  - Wasm median 99.7 KB
  - JS median 2.79 MB

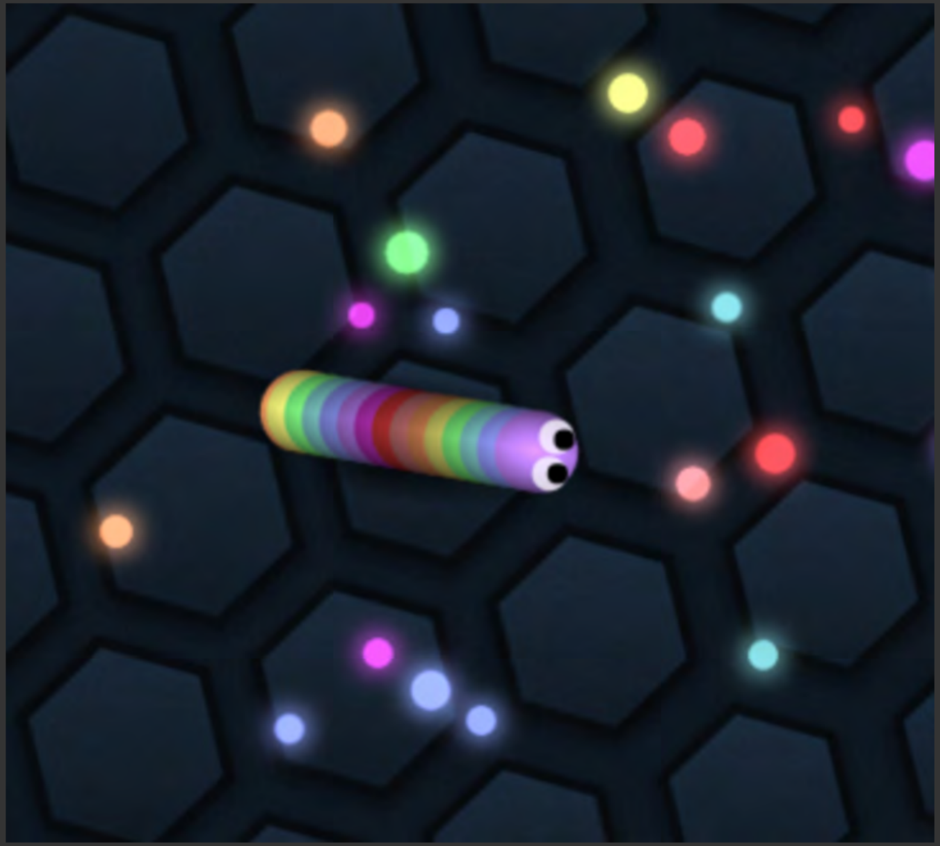


Applications of WebAssembly

Game

- 44 unique samples on 58 sites





Custom, Library and Test

Custom

- 17 unique samples on 14 sites
- Example programs, Background animations, ...

Library

- 25 unique samples on 636 sites
- Draco: Decompress 3D meshes

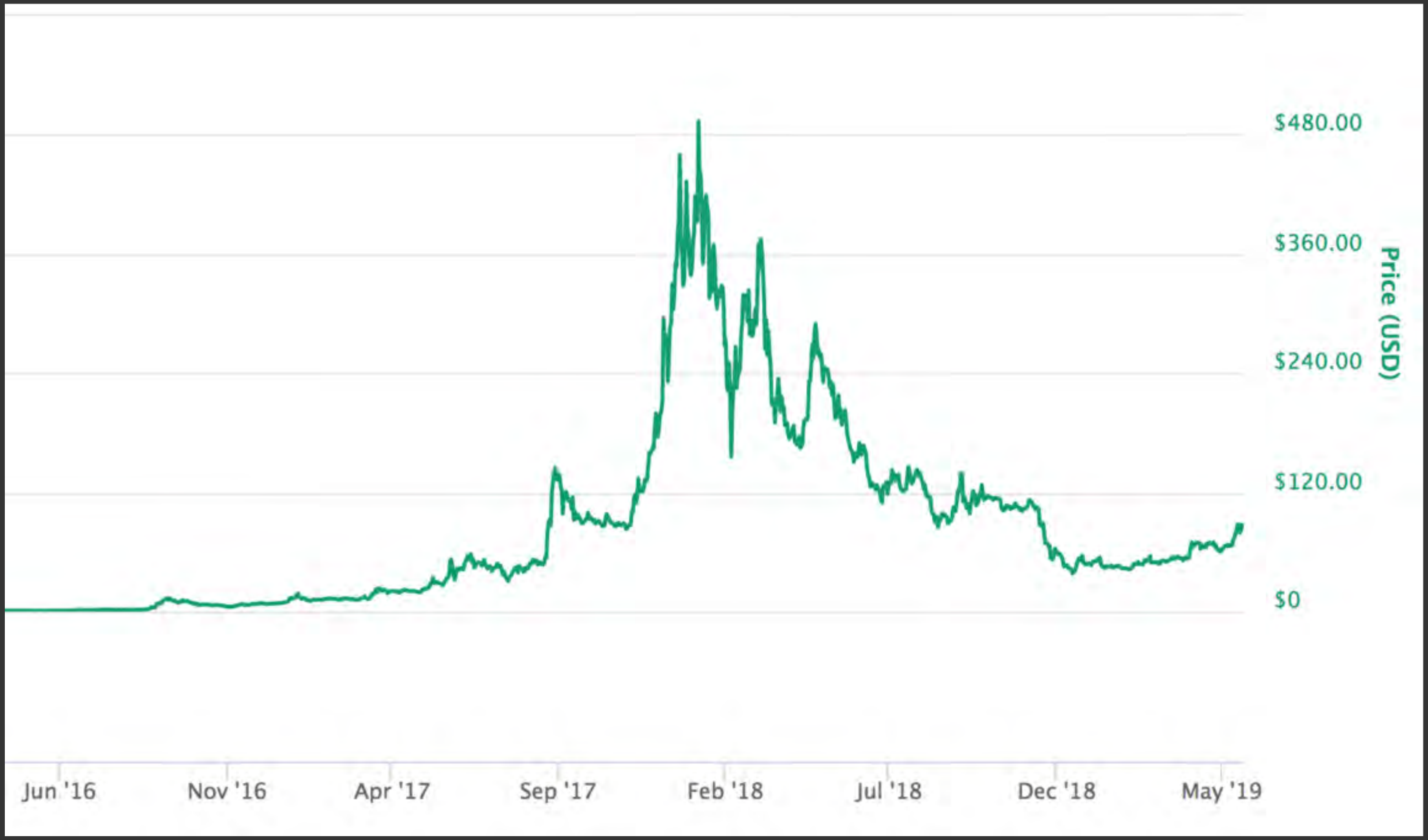
Test

- 2 unique samples on 244 sites

```
var a = new WebAssembly.Module(Uint8Array.of(0,97,115,109,1,0,0,0));
return new WebAssembly.Instance(a) instanceof WebAssembly.Instance;
```

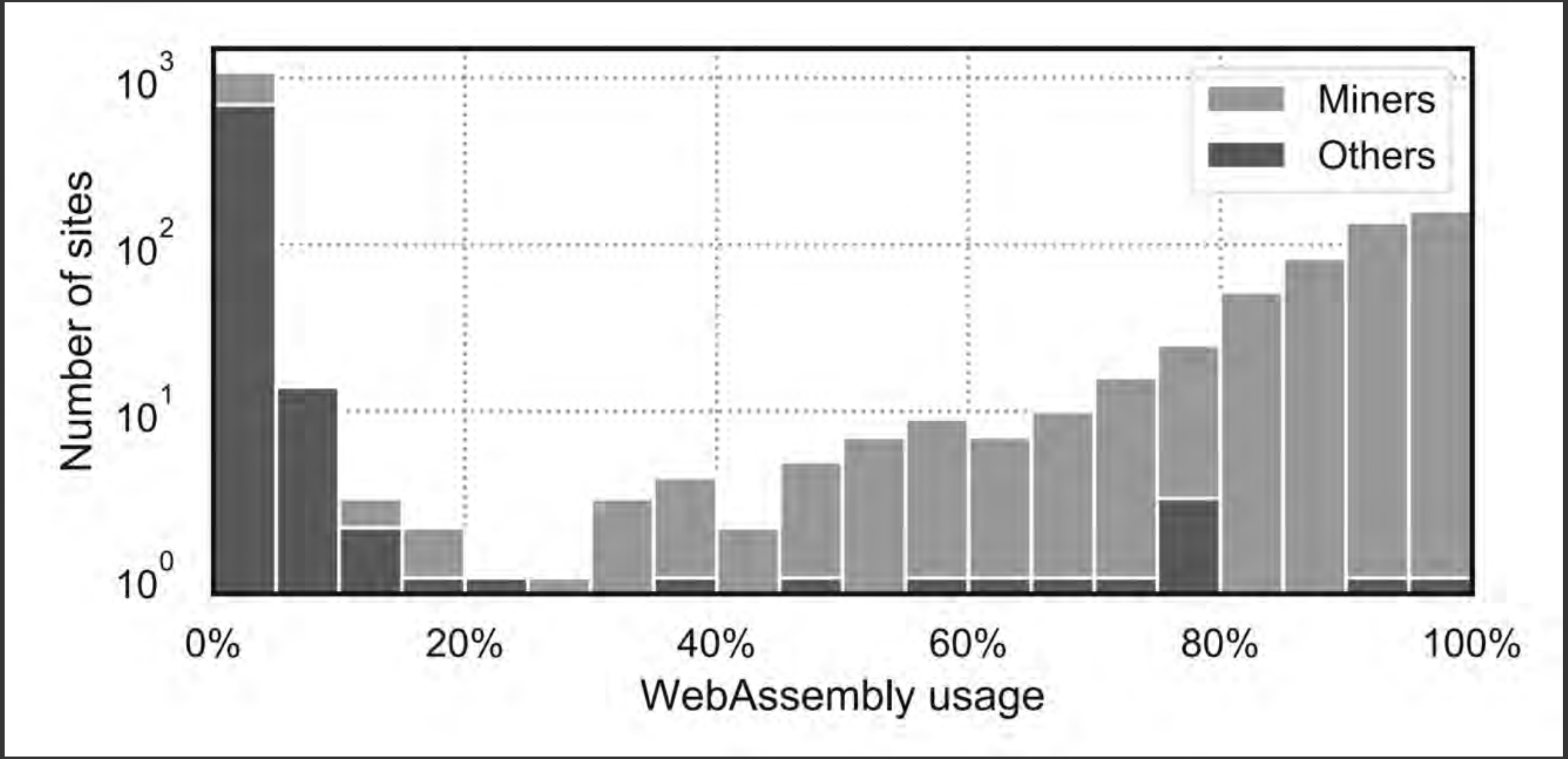
Mining

- 48 unique samples on 913 sites



Mining

- 48 unique samples on 913 sites

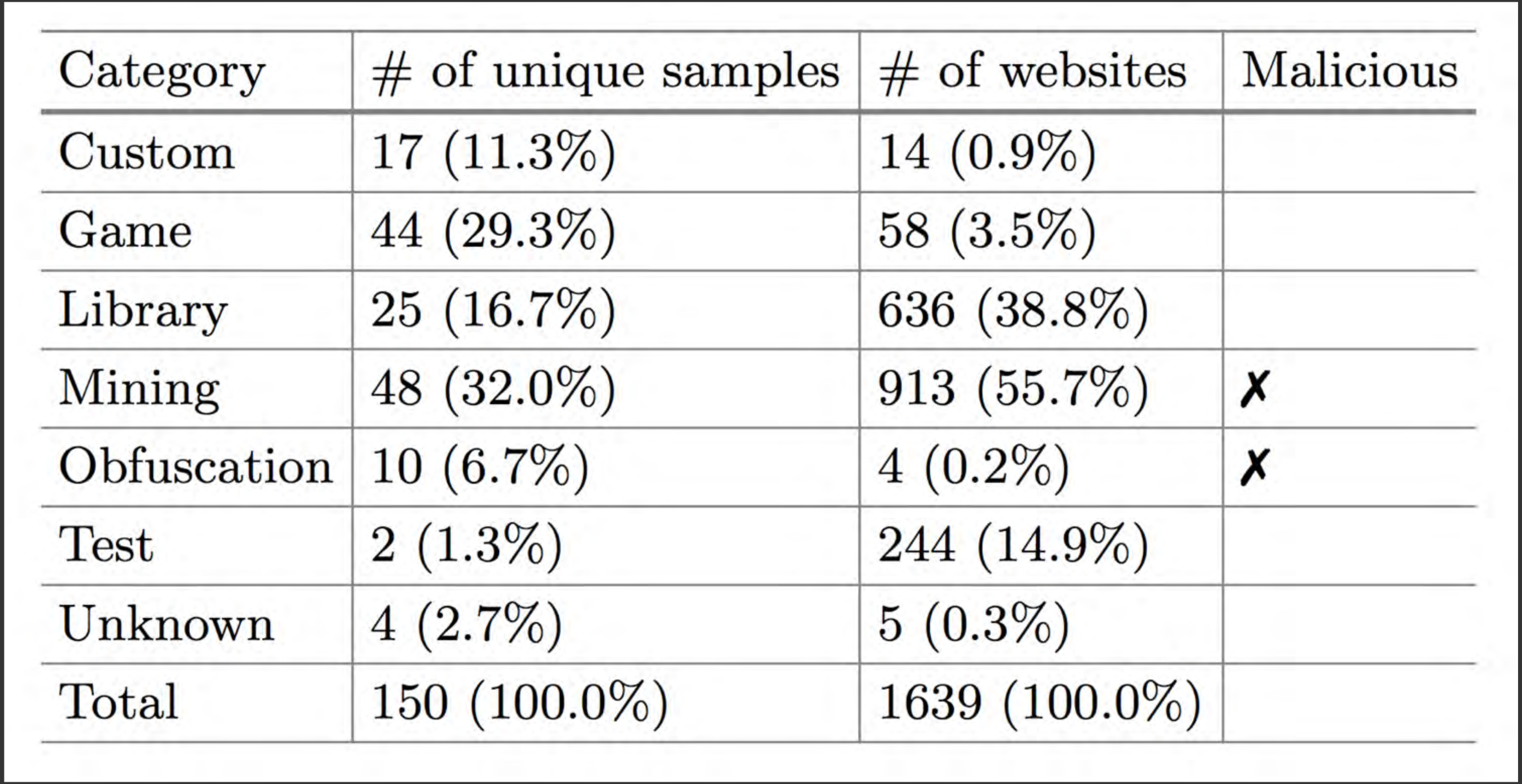


Obfuscation

- 10 unique samples on 4 sites
- Code embedded in the Wasm memory section

```
<script>
var popunder = {expire: 12,
url: ’//hook-ups-here.com/?u=8l3pd0x&o=4gwkpzn&t=all’};
</script>
<script src=’//hook-ups-
here.com/js/popunder.js’></script>
```

Overall



Category	# of unique samples	# of websites	Malicious
Custom	17 (11.3%)	14 (0.9%)	
Game	44 (29.3%)	58 (3.5%)	
Library	25 (16.7%)	636 (38.8%)	
Mining	48 (32.0%)	913 (55.7%)	✗
Obfuscation	10 (6.7%)	4 (0.2%)	✗
Test	2 (1.3%)	244 (14.9%)	
Unknown	4 (2.7%)	5 (0.3%)	
Total	150 (100.0%)	1639 (100.0%)	

The Future of Malicious Wasm

Possible progress

- Embedded HTML/JavaScript code
- Loader in Wasm
- Full implementation in Wasm
- Fully intertwined code

Conclusion

- Exciting new feature for the Web platform - but also for attackers
- Currently, over 50% of the sites misuse it for cryptojacking
- Enables novel obfuscation techniques
- Effective defense mechanisms will need to incorporate WebAssembly analysis

Thanks for your attention :)

Questions?

Contact

- Mail: m.musch@tu-bs.de
- Twitter: @m4riuz

IAS INSTITUTE FOR APPLICATION SECURITY

Technische Universität Braunschweig

Institute of System Security