Saint Petersburg
State University of Aerospace Instrumentation

DETECTION AND NOTIFICATION OF ADDITIONAL ACTIONS OF SIGNED JAVA APPLETS

Roman Zharinov
Ulia Trifonova

08/11/2011

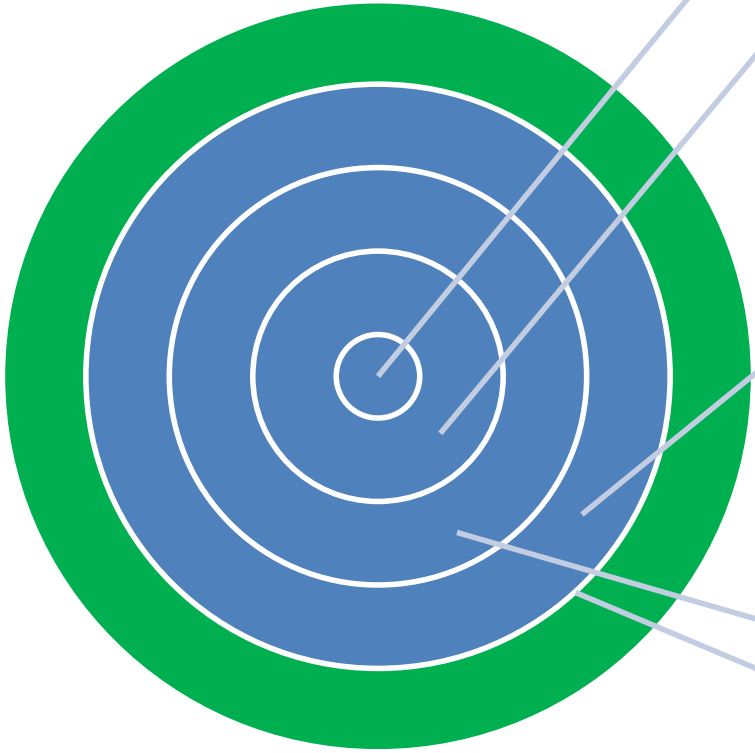
Contents

- Main goal
- Security architecture of java-applet
- Potential attack
- Analogues
- Review detection system
  - Tasks, algorithm, etc.
- Conclusion

Main goal

- Some information
  - Nowadays a lot of modern web-applications are used JavaScript, where needs additional functionality usually used java-applets;
  - Both technologies are used in applications, where need security aspects: billing systems, web-banking, etc.;
  - At this moment there is no applications that can detect threats from this class of web applications in real-time.
- Goal
  - Identify and classify potential sources of threats;
  - Create an extension for Google Chrome browser that can detect malicious source code in the real time.

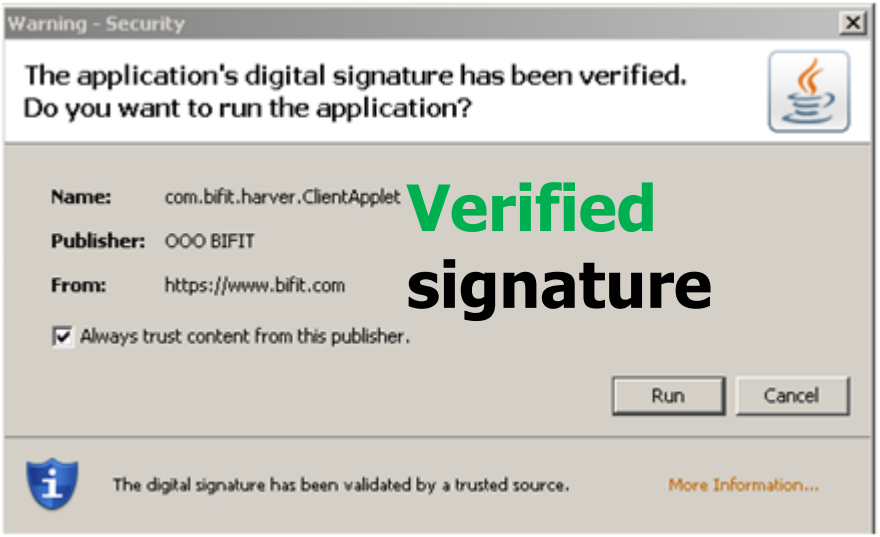
Security architecture of java-applet

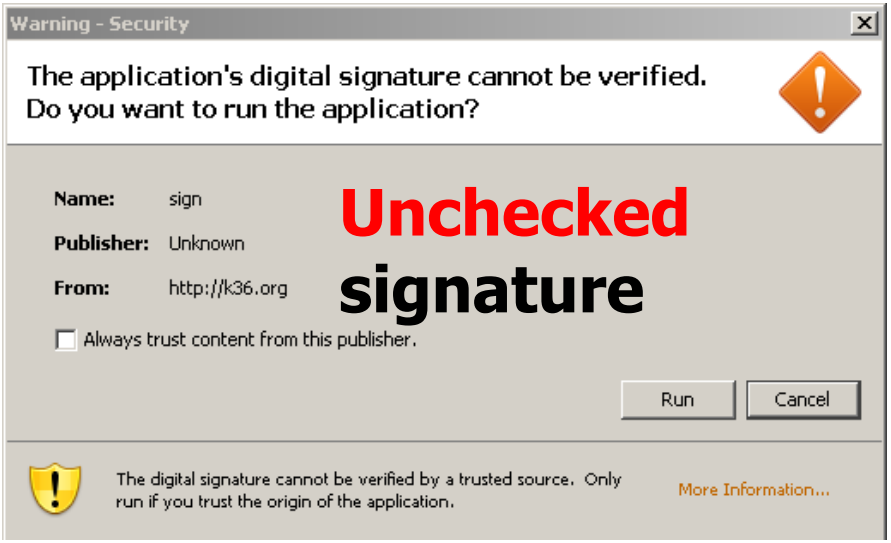


- Internal
  - Use the sandbox (*necessarily* - JRE functionality)
  - Digital signature (*optional* - set up by the developer)
    - **Checked –** trusted certificates
    - **Unchecked –** self-signed certificates
  - Confirmation window than applet retrieve additional functionality
- Extended
  - Security connection (ssl/tls)
  - Obfuscate source code
  - Detection system

Applet security: Digital signature

- Signed applet
  - Access to user filesystem;
  - Create network connections;
  - Execution of operations outside the sandbox;
- Verified signature – signature which is located in a trusted CA list;
- Unsigned applet
  - Execution of "safe" code;
  - interaction with JavaScript;
  - Almost is not used.





Potential attack

Potential attack	Execute by JavaScript	Execute by Java-applet
DoS – generation of windows	+ while (true) {alert('hello')}	+ while(true) {JOptionPane.showMessageDialog(this, "hello");}
DoS – CPU and memory load	+ while (true) {some_calc()}	+ while (true) {some_calc()}
Information collect available to the browser: email, cookie ...	+ document.getElementById(…)	+ *Call js code from class*
Data transfer without reloading the page (AJAX)	+ new XMLHttpRequest()…	+ new URLConnection() …
Access to user filesystem – operations with files	–	+ *Signed applet*
Execution of programs	–	+ *Signed applet*

Protection of client-side Web-page

Software, which ensures the safety of the client from malicious Web-applications

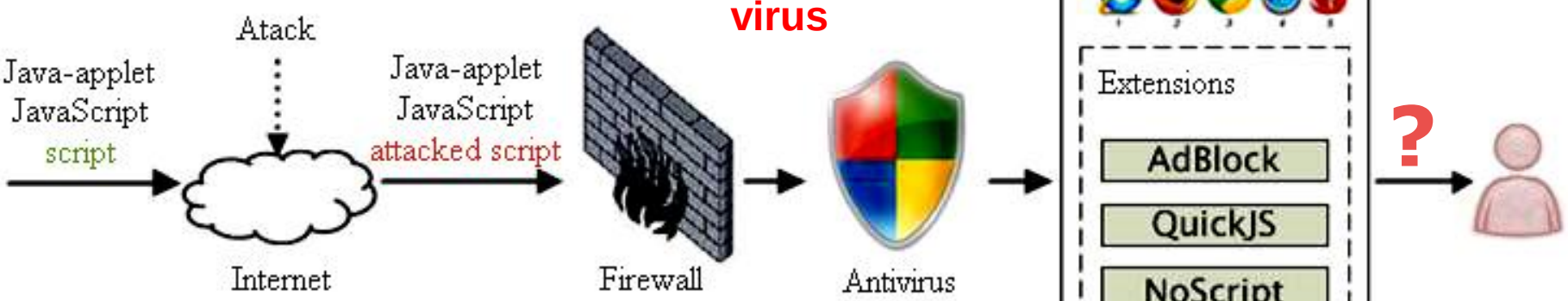
Product	Purpose	+	–
Antiviruses (Eset, AVP etc.)	Scanning for viruses and malware.	**Detection of viruses**	• **Do not focus** on client web application • **Require** professional settings
Firewalls (Outpost Firewall etc.)	Monitoring network traffic, search advertising, etc.	•Removal of advertising •The ability to block scripts, web sites •Blocking the transfer of personal data	
Browsers tools	«AntiDos js»	• **Blocking** multiple js windows • **Ability to control the browser** (to access pages)	• **Limited functionality**

Protection of client-side Web-page

Built-in tools of browser, aimed at ensuring the safety of Web client-side

Browser extention	Purpose	+	–
AdBlock	Ads, banners	blocking ads	No analysis of malicious code
QuickJS	Disabling JavaScript on the page	• Turn off by key combination • Attacks -> 0	**Turns off all of the code** as a useful and malicious
NoScript	Run only active content on "trusted" sites	Running the executable content to user-selected sites	For each site, you need to think before they give permission

Illustration of security systems



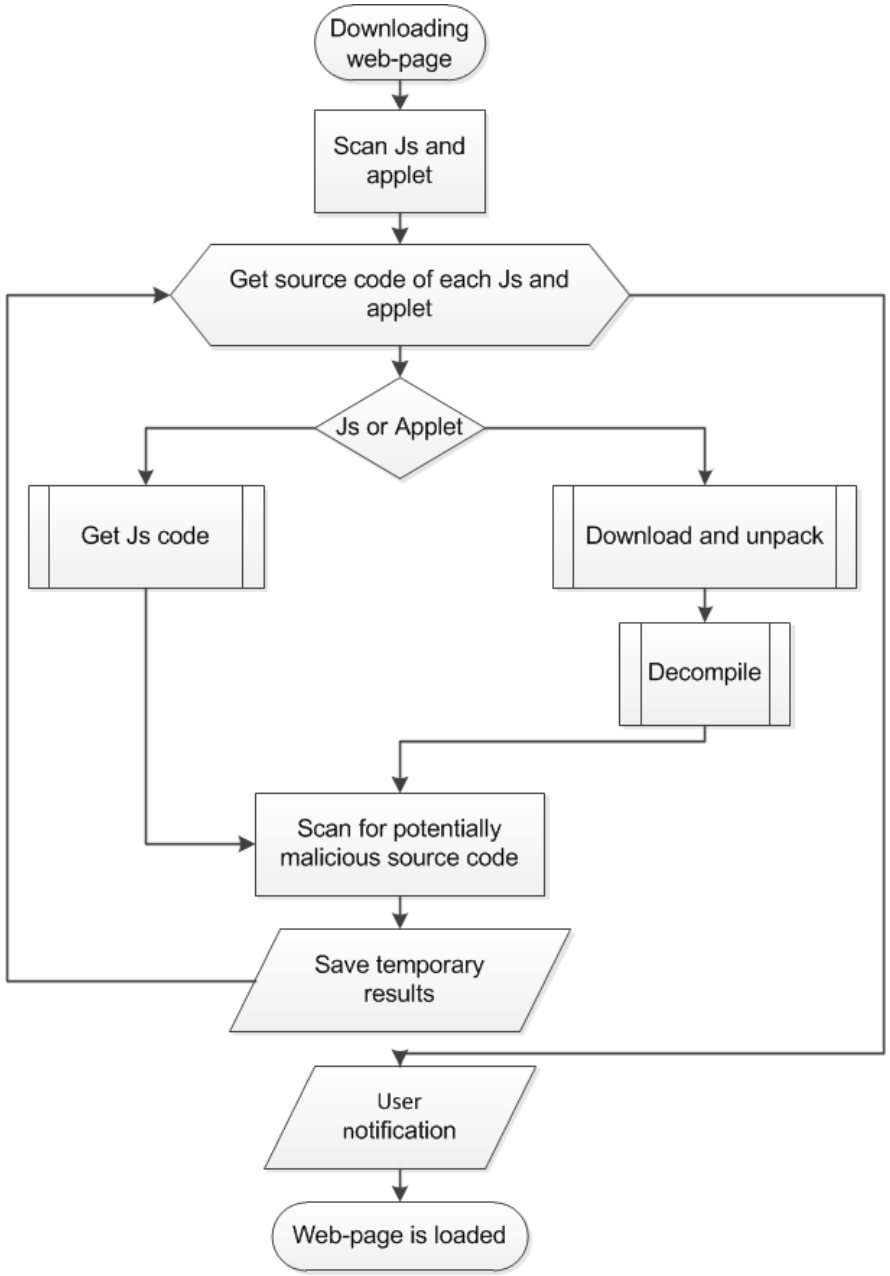
- Lack of security templates
- Applet, which is virus
- Connection outside the firewall
- Deactivated JavaScript
- Uncomfortable work in a network without JavaScript

Tasks of detection system

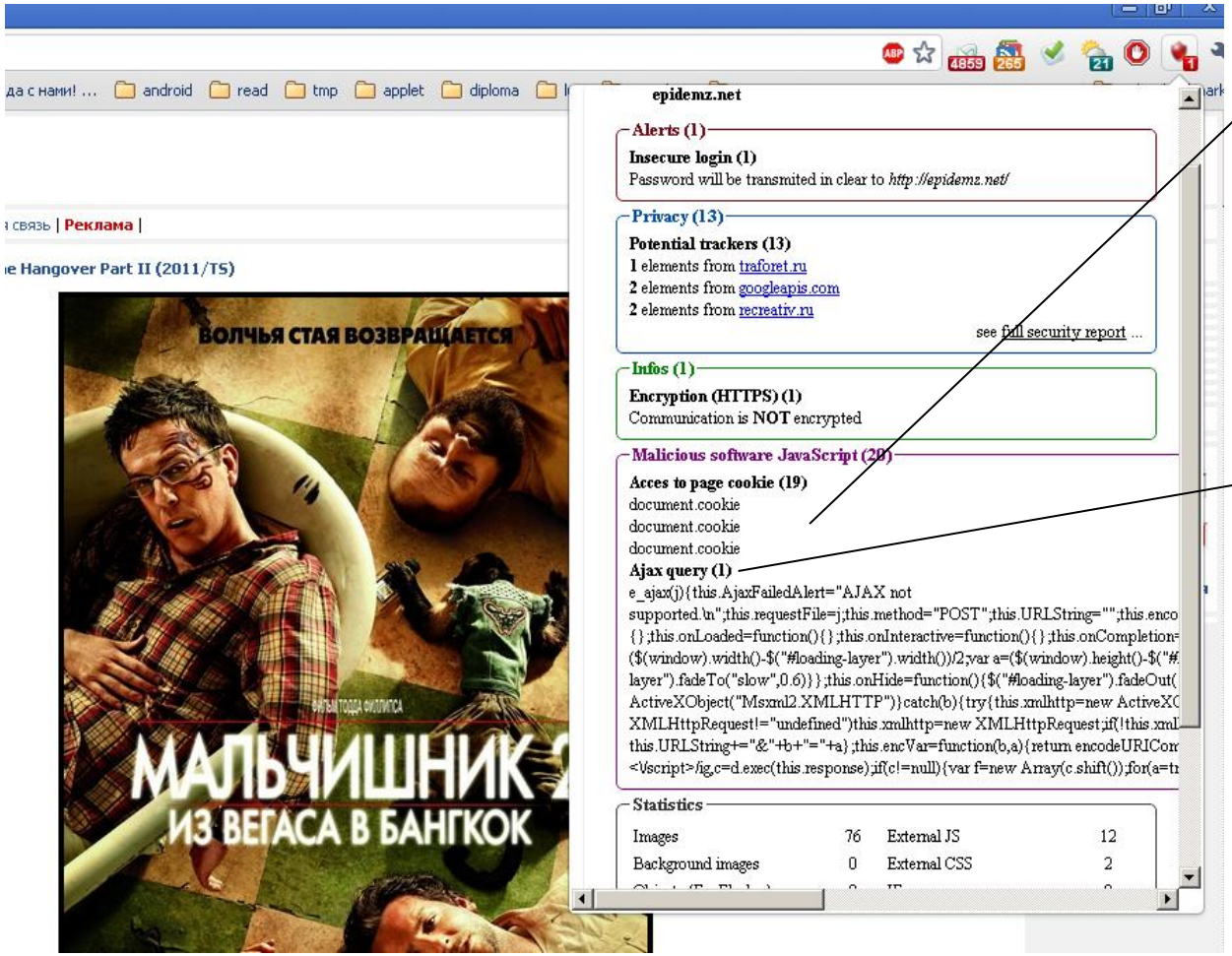
- Warning when sending a password over insecure channel;
- Redirect the user to a secure version of the site, if supported;
- Analysis of all client-side scripts for the presence malicious code.

Algorithm

- Scan all client-side code;
- Scan all Java-applet
  - Unpack jar-archive;
  - Decompile class-files;
- Search of malicious code, using patterns;
- Create a full report to the user.



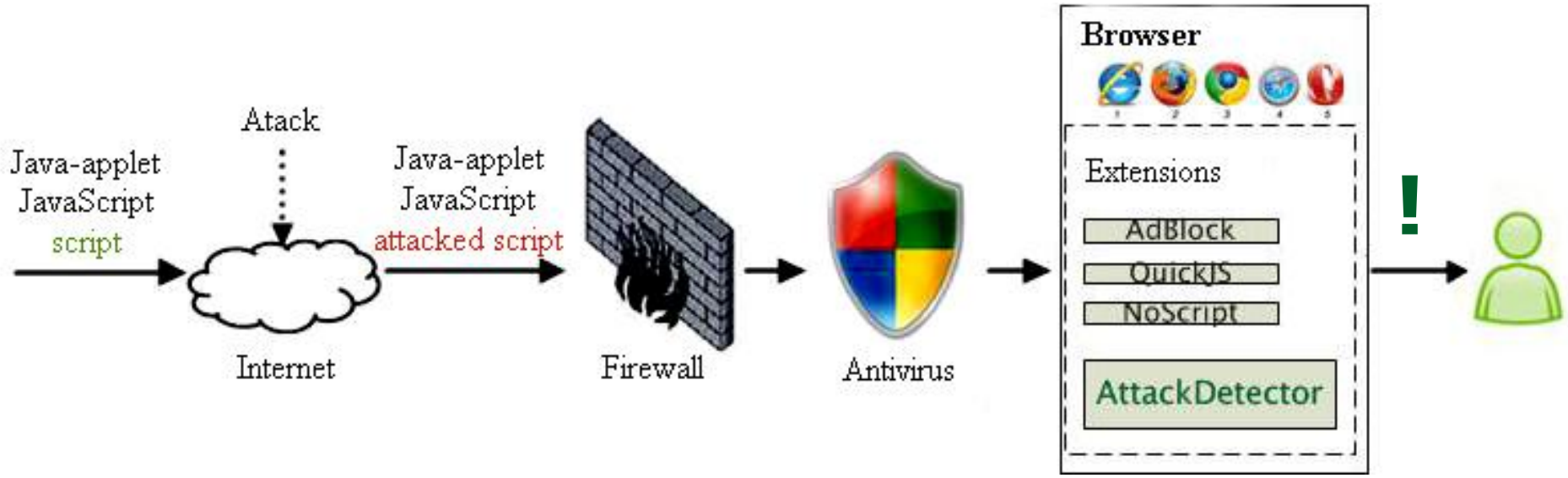
Example of the detection system



Access detectionto cookie

Possibility use AJAX technology

Illustration of the system



Detection of malicious source code (JavaScript and Java-applet)

Conclusion

- Results of our work
  - Review security architecture technology java-applet;
  - Identified direction of potential attacks on the client-side Web-pages;
  - Create an extension for Google Chrome browser that can detect malicious source code in the real time.

Thanks for attention!

Questions?