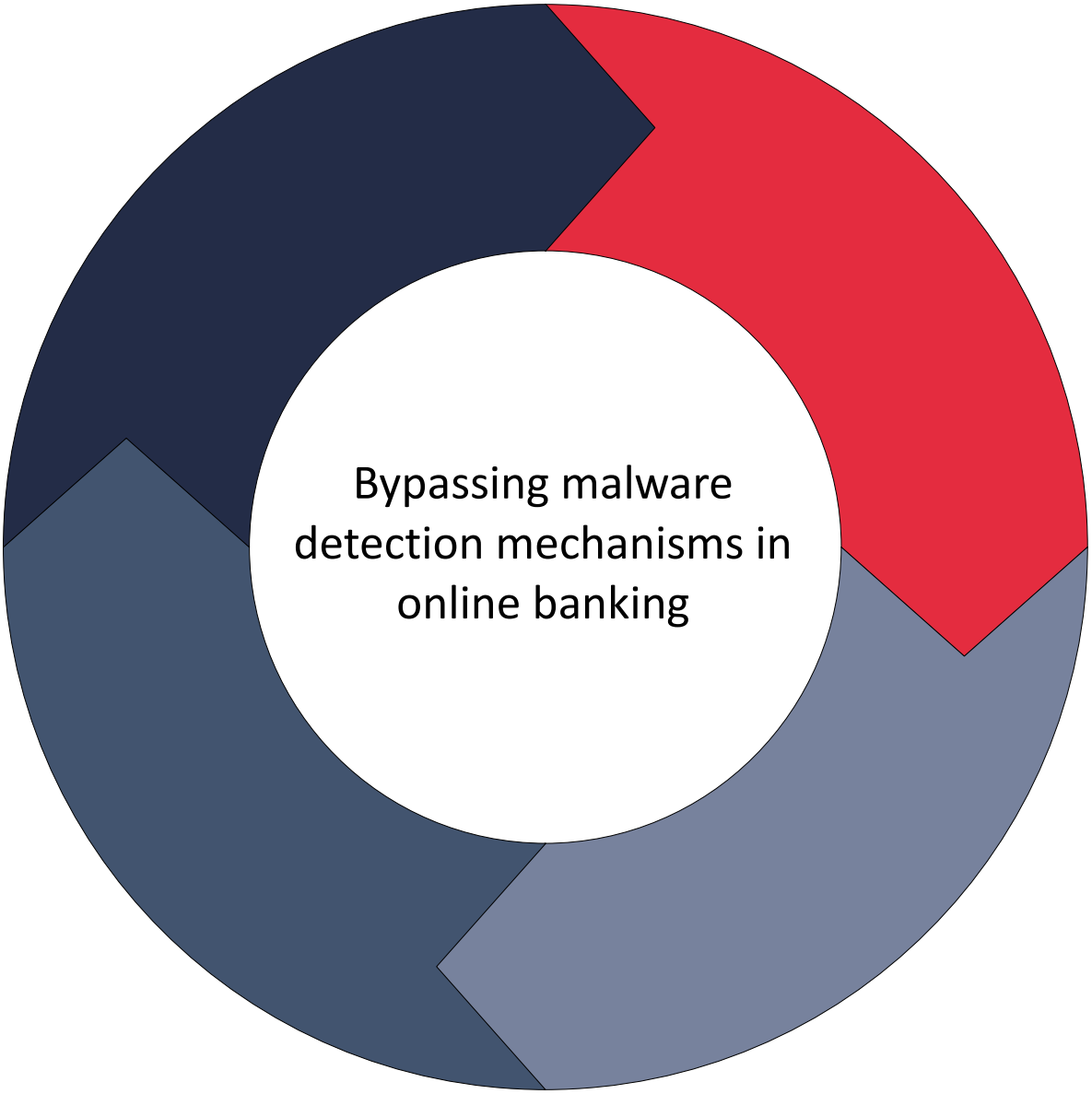


Bypassing malware detection mechanisms in online banking

Authors:
Jakub Kałużny (jkaluzny@securing.pl)
Mateusz Olejarka (molejarka@securing.pl)

SecuRing
e-mail: info@securing.pl

1. Abstract

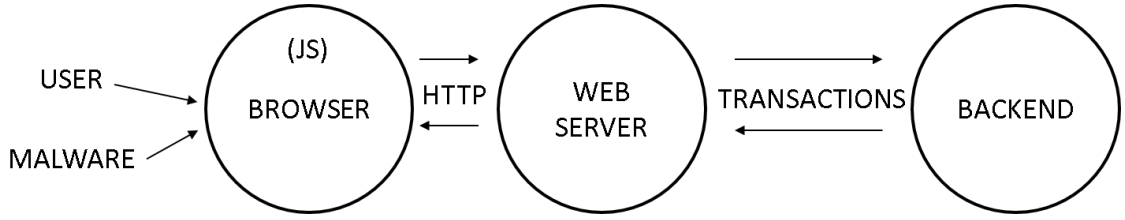
Online banking applications are particularly exposed to malware attacks. To minimize losses, banks have invested in malware detection mechanisms which are not running as programs on client machine but either server-side or by JavaScript in the risky application. We have tested many solutions which are using different detection methods such as behavioral patterns, web injects signatures, or user input analysis. Our research points out clearly: even "100% malware proof solutions" have serious implementation errors. It is only a matter of time when malware creators will start targeting their guns against these vulnerabilities, effectively bypassing or abusing costly countermeasures. Is it a road to failure or can we improve them?

2. Introduction

In this paper we present the concept of anti-malware solution, vulnerabilities we have found in products we tested and in the end the recommendations for vendors and for potential buyers of such solutions.

2.1. How does banking malware work ?

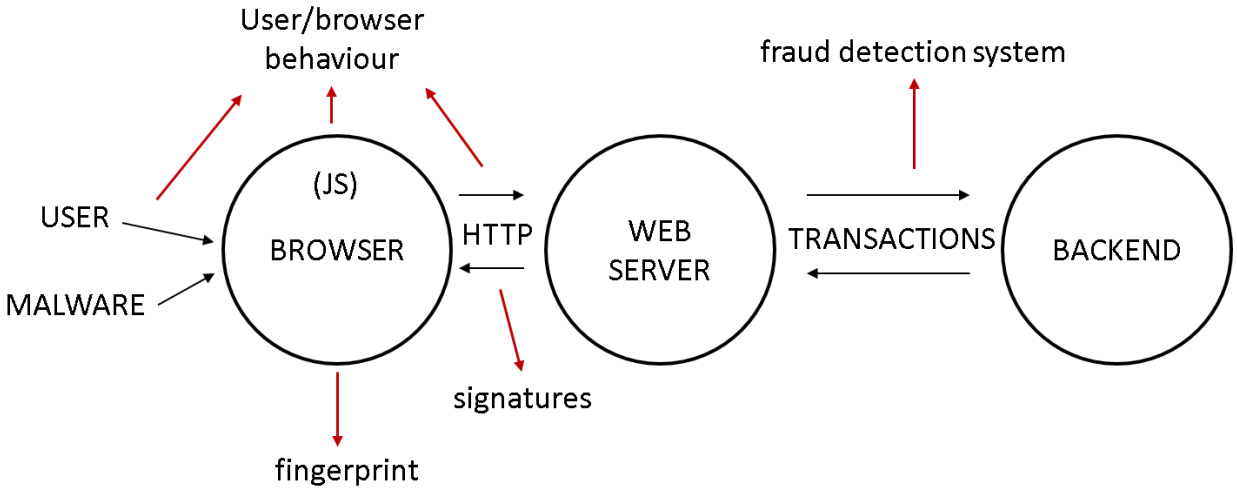
Most of banking malware steals confidential data or modify transaction parameters. Theoretically, a malware on an infected machine can be totally invisible for the user - tampering both user requests and responses received from the server. By doing this, it could change transaction parameters in a way that would raise an alert.only with a secure second channel showing transaction details (such as SMS).



However, a significant number of malwares use “web injects” – a short JavaScript added to server response, modifying authentication or authorization mechanism. A second popular way of modifying recipient account numbers is to tamper computer’s memory, so that each 26-digit number in memory gets changed to attacker’s chosen one. The last considered malware method is automating tasks after stealing data and sending payments in mass.

2.2. What is online malware detection ?

As mentioned before, the main concern for preventing malware from stealing money is to detect web injects. This can be done by adding a special JavaScript to a bank’s website, which will analyze DOM tree looking for web injects. In most cases, this JS code contain web inject signatures in form of regular expressions. Second observed method of detecting malware is to fingerprint the browser looking for suspicious settings and add-ons. Most advanced method is to build a fraud detection systems which can analyze previously mentioned methods output and also data from transaction system.



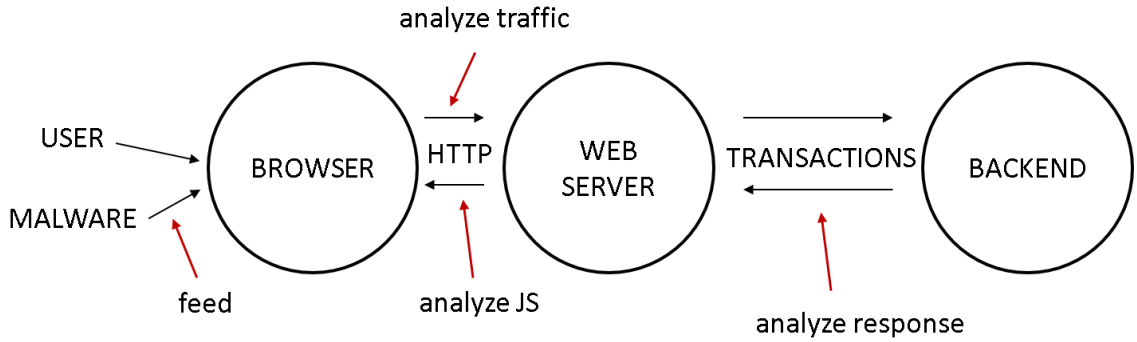
2.3. Limits

Each of these methods has it pros and cons. None of them will detect every malware. However, we believe that every malware countermeasure raises the bar for malware operators and at least prevents attacks from massively sold boxed malware. All of these methods are very dynamic and malware signatures can be changed very quickly. Nonetheless, behavioral systems are still based on a HTTP – HTML – JS stack and can therefore contain architecture and implementation errors, which we present in this report.

3. Vulnerabilities

3.1. Our approach

We fed tested solutions with malware looking for detection mechanisms and interesting requests, we analyzed obfuscated JavaScript and output in transactional systems.

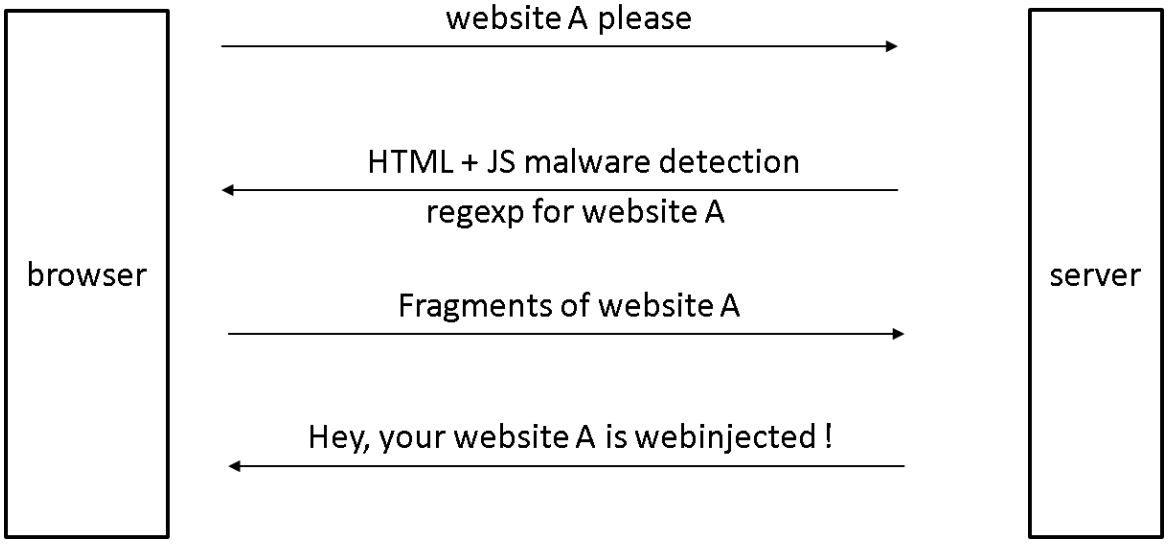


3.2. Observed mechanisms

When user opens a web application protected by anti-malware solution which has client side component at some point its JavaScript code is executed (for example after particular JavaScript file is loaded). The script contains code designed to check for malware signatures on the clients side.

The signature is a constant string, which contains a part of web inject (for example JavaScript function name which is added as a part of web inject). The code checks if the string is present in for example html code of the form or frame. Sometimes the type of the JavaScript object related to given string is also checked (for example if there is an object named "X" and it is a function).

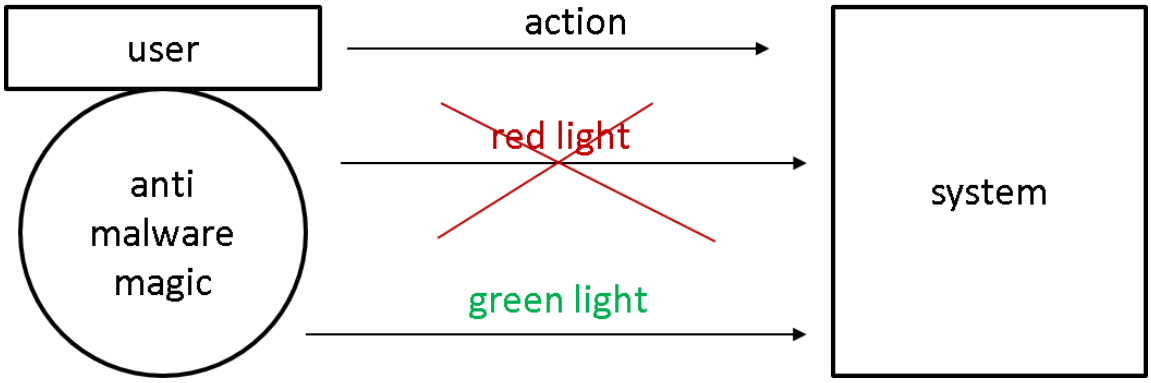
When all signatures are checked, the results are send to the server. Sometimes this is done after each check and only when the result was positive (malware found). Then, depending of the signature check result, next steps are performed. It the result is negative (no malware found), the user is allowed to proceed normally. Otherwise (a positive result), the system generates an alert about detected malware, and additional actions are performed. For example user is alerted about the situation using an out-of-bound channel (for example user gets a call from CC) and his session in the application is terminated.



In a few cases described communication scenario was performed for each page in the application, not only on the login page.

3.3. Disabling and bypassing mechanisms

The ground for first two of following vulnerabilities is because anti-malware solution used a “red light” system”. An alert was raised, and a request to the server was sent only if malware was detected. We advise that a “green light” should be used instead, so that a simple reverse inference could work: no signal – no go.

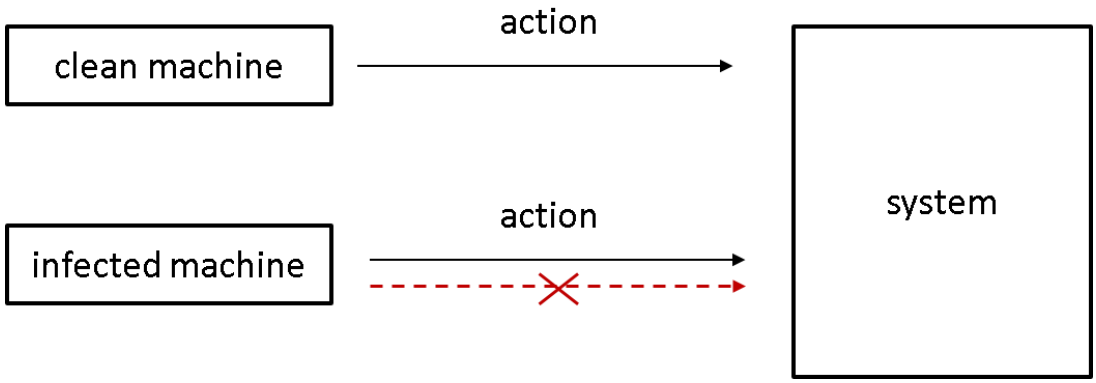


3.3.1. By cutting off the traffic

It seemed obvious for us to test how each of tested products reacts, when we will cut communication to particular URL address (related either to initialization phaes or to reporting phase). We were stunned by the results.

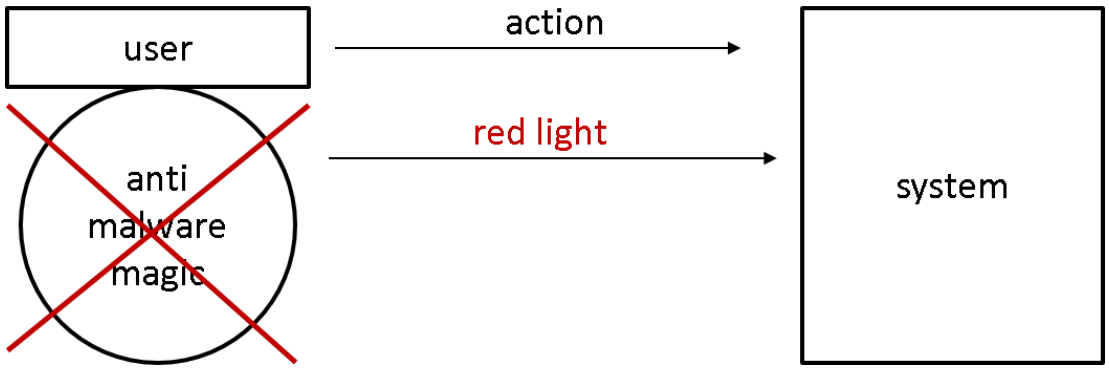
In most of the cases it was enough to disable an anti-malware solution. Simply drop a HTTP request to a given URL, or redirect it into nowhere was enough: both at initialization step of the solution and in the reporting step as well.

First failure could be connected with possible integration error (anti-malware didn't get info from the web application that new user session appeared from the web application), the second should definitely be reported by it (new user session, but no signature check result send for that session) after preconfigured timeout.



3.3.2. By too fast login

One of tested cases was very interesting. A very fast login (but still achievable by man) on an infected machine will result in no alert raised. This happened because anti-malware JS was not fully loaded and it didn’t execute. Because the solution did not predict such situation, malware was not detected.



3.4. Abusing mechanisms

3.4.1. By tampering signatures

On the client side it is a struggle between webinjects and anti-malware scripts. The signature check is done on the clients side, so therefore signatures itself must be known on the client side as well. We believe that the signatures and code responsible for the verification and sending back the result should be obfuscated to make the understanding of the mechanisms harder for an attacker.

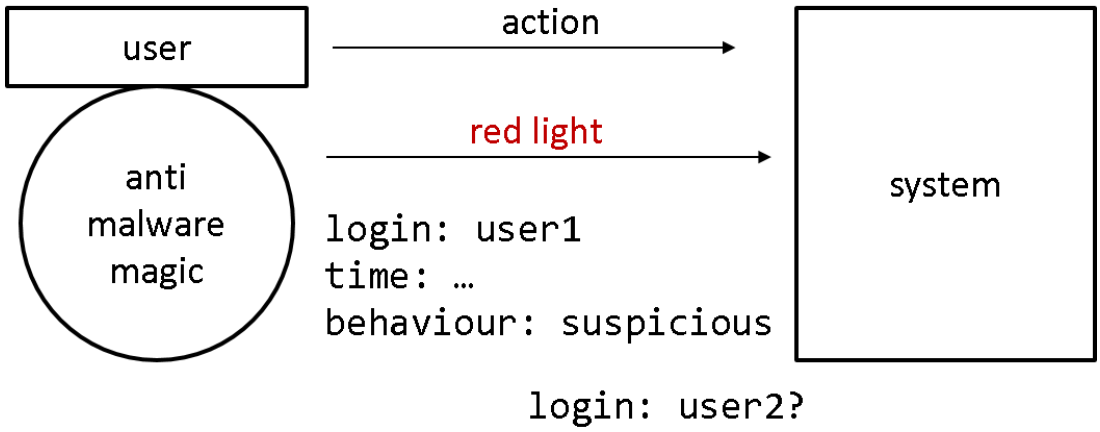
In few cases we found lack of obfuscation, or the obfuscation used was fast & easy to deobfuscate and analyze.

Here arises the question, if we are able to do it bullet-proof in JavaScript. The answer in our opinion is no. Due to its nature, each client site JavaScript based solution given enough resourced can be analyzed and possibly bypassed. Therefore it should be well concealed in the web application, and not so obvious to find/isolate from other JavaScript code and hard to bypass. The obfuscation algorithm should use a pattern of different obfuscation techniques, and to change this pattern in each time it is initialized (the JavaScript code is returned to the client side).

One to the most interesting findings in script which does signature check in one of tested solutions was the fact that results indicating "no malware present" state was known on the beginning. The array which holds results of the checks was initialized with some values, then all signatures where checked (one array entry for each) and if no signature was found the array was returned to the server. We were able to automate the attack, so that the script reported back always that there is no malware detected for given session.

3.4.2. By tampering alerts

Apart from bypassing solutions, one interesting case let us to abuse the whole reasoning backend system. This time, after spotting the malware, it was the anti-malware JS script which sent alert data back to backend system. It contained login name, date and few other details encrypted with RSA public key hardcoded in JavaScript. Tampering these values let us to abuse the system, raise false alerts and create a decoy for other attacks.



4. Conclusions

4.1. Recommendation for buyers

If you think about buying such a solutions there are few things to consider. First it is important to remember that it is another piece of software and as such it can contain vulnerabilities. So prepare test cases and execute them against test environment or hire skilled consultant to do that for you. Second thing - what is in our opinion one of the most important thing, when having such a solution - it is not a malware detection rate which is matters most. More important is how quick anti-malware can adapt itself to an ongoing attack - how fast new signatures stopping malware which is stealing your money in this very moment can be deployed? Keep it in mind when thinking about SLA with vendor.

Anti-malware solutions cannot be considered as only precaution against malware. It is just another layer of defense that can help your security team in keeping you and your users safe. Have a strategy, gather your team of specialist, gather knowledgebase, keep track on the news, keep an eye on the dark side. Anti-malware should be well integrated into your security monitoring and anti-fraud systems, collect stats about the users behavior, calculate risk connected with each user session and transaction, be adaptive.

4.2. Recommendation for vendors

Creating anti-malware solutions is a good thing but remember: they are not stupid, malware creators. They do not go away when they see their attacks fail because of some obfuscated JavaScript code. They have the resources, so watch your back. Test yourself, be few steps ahead with adjustments of your platform ready to be deployed. Follow “green light” rule when raising malware alert. Don't be afraid of the researchers, we just want to make the online world a little bit safer, so please, be willing to cooperate.

Jakub Kałużny (jkaluzny@securing.pl)
Mateusz Olejarka (molejarka@securing.pl)
SecuRing: info@securing.pl