



# Compass Security

## Finjan SurfinGate Analysis

### January 27, 2003

Document Name:	FinjanSurfinGate_Analysis_CSNC_V3.0.doc
Version:	V 3.0
Project Number:	
Author:	Jan P. Monsch, Compass Security AG Ivan Buetler, Compass Security AG
References:	Referenz
Date of Delivery:	January 27, 2003
Document Status:	PUBLIC

GLÄRNISCHSTR. 7  
POSTFACH 1671  
CH-8640 RAPPERSWIL

Tel. +41 55-214 41 60  
Fax +41 55-214 41 61  
info@csnc.ch www.csnc.ch



## Table of Contents

<b>1 INTRODUCTION</b> .....	<b>1</b>
1.1 Preamble	1
1.2 What is SurfinGate?	2
1.3 Test Environment	2
<b>2 EXECUTIVE SUMMARY</b> .....	<b>3</b>
2.1 File Type Enumeration	4
2.2 Active Content	4
2.3 Administration Configuration	5
2.4 Limitations	6
<b>3 SECURITY CONSIDERATIONS</b> .....	<b>8</b>
<b>4 TEST CASES</b> .....	<b>11</b>
4.1 HTML	11
4.2 JavaScript / VBScript	12
4.3 VBScript	15
4.4 EML	16
4.5 EICAR Virus Pattern String	17
4.6 EXE-Files	18
4.7 Office Documents	19
4.8 Java Applets	20
4.9 Inside-Out Attacks	21
4.10 ActiveX	21
4.11 HTTPS connections	22
<b>5 CONFIGURATION DATABASE</b> .....	<b>25</b>
5.1 Administrator Console Password	25
5.2 Oracle Password	26
<b>6 ADVISORIES</b> .....	<b>ERROR! BOOKMARK NOT DEFINED.</b>
6.1 SurfinGate 6.x Bypass Vulnerability	<i>Error! Bookmark not defined.</i>
6.2 Bypass SMTP Content Protection	<i>Error! Bookmark not defined.</i>
6.3 SurfinGate IP Address To Hostname URL Filter Bypassing Vulnerability	<i>Error! Bookmark not defined.</i>

## 1 Introduction

### 1.1 Preamble

After the last years of security investment, you felt yourself secure. You have a working firewall infrastructure installed, combined by a network- and host based IDS system. Log files are sent to a central server and arbitrary activities are recognized. The clock of your demilitarized system is in sync with the other systems and therefore prepared for incident handling. You have defined security policies, hardened operating and application servers – and if you look back – you feel confident. You **MUST** be secure.

Beside the never ending race condition, between new published remote executable buffer overflow or any similar dangerous hacking opportunity, where someone could send arbitrary malicious mobile code to your firewall protected systems and starting existing and own software components, you know the risk. Your IT staff is aware in monitoring upcoming security patches and you are professional in reducing the time slot where someone could successfully exploiting your computer resources and the protection hot fix installation procedure (in case the vulnerability is being published)

We have been focussed against hackers penetrating the computer resources directly. But what do we do against malicious mobile code? Yes – you are right. You have your perimeter anti-virus installed and mail and web content-filters applied. You are only accessing the Internet via corporate proxy including authentication. So what – where is the problem?

From our experience, there are plenty of problems only in the web and mail technology. Hackers are able to inject their code into mails, attachments or arbitrary websites. Talented hackers can direct their attacks at their victims in ways to bypass intrusion detection systems. Due to a case sensitivity problem of the Content-Type and Content-Disposition allows attackers to cause many commercial content filters to incorrectly not detect a message's attachments.

Finjan SurfinGate promises to protect you against malicious mobile code, such as MailSweeper or TrendMicro. We have had the chance to penetrate the SurfinGate 5.6 within our lab infrastructure and would like to share our knowledge. However – we have found several techniques to bypass the Finjan SurfinGate. Don't get in panic. We all know, anti-virus protection has some rest risks – sometimes the risk gives you the rest. Content filter technologies make very much sense to us and we are looking forward getting more developed versions from multiple vendors.

This article analysed Finjan's SurfinGate 5.6. During the test-phase, version 6.0 was released. We have introduced into changes where possible – but not everywhere.

## 1.2 What is SurfinGate?

Finjan SurfinGate is a Web content security and management tool for Windows and Unix-based operating systems.

SurfinGate is a security defence line for the corporate network. It is your firewall for Java, ActiveX, executables, documents, Java Script, VB Script and plug-ins. However, as opposed to current firewalls that are only capable of either allowing or blocking all active content requests, SurfinGate intelligent content inspection technology goes a step further. It enables the security administrator to grant access to productive downloadable and block or monitor hostile ones from entering the network.

## 1.3 Test Environment

The following software has been reviewed:

- SurfinGate 5.6 for Windows NT
- SurfinGate 5.6 for Solaris

Version 6.x is already available.

## 2 Executive Summary

Finjan SurfinGate provides powerful content filter capabilities. Compared to other content filter products, bypassing malicious mobile code becomes a challenge. Finjan's SurfinGate follows the black list<sup>1</sup> approach, whereby the white list<sup>2</sup> approach is recommended. Although SurfinGate does not block **all** kind of malicious content it still offers to some extent protection against simple attacks. But for more sophisticated attacks where the attacker knows that SurfinGate bypass techniques will success.

The product is a black box and for operators it is very difficult to configure and to supervise the correct operation of the content filter. Especially the black list approach for file content filter prevents tight configuration. White listing would be much easier to handle.

Firewalls are usually following the white list approach – Compass considers this approach as more secure. In some situations, eventually, the white list approach gives you a shine-security level. Configuring a white list approach content filter requires strong knowledge by administrators and engineers. If not given, a white list could easy become a zero-protection content filter, by enabling all default settings as a result of misunderstanding the offered technique.

We recommend using Finjan's SurfinGate for those sites, having time, knowledge and manpower maintaining the Finjan on a daily rate. It gives you additional security, when professional expert skills are given to the administrator.

This additional security protection is being paid by higher cost of ownership. Without calculating the TCO (total cost of ownership), we expect Finjan expensive in the daily usage.

While playing with SurfinGate during the test phase, we found the trace capabilities after denying certain content the most insufficient feature. It causes real headache searching through the denied webpage hunting for the place responsible triggering the deny rule. We have heard, this must be investigated in Version 6.x.

This report summarizes Finjan SurfinGate limitations, given in Version 5.6. As expected in today's security software, there are hints and tricks to bypass the content filter rule engine. Hopefully this article gives you a better understanding of Finjan's SurfinGate capabilities and restrictions.

A company using this production should not solely depend on the security of a content filter. Other factors are essential as well:

- Having the web browsers and email clients patched to the latest security patches,
- Training users in security awareness.
- Employing rules on how the user may utilize the Internet access.

---

<sup>1</sup> *black list approach: every packet passes by default except self-configured denied rules*

<sup>2</sup> *white list approach: every packet is denies by default except self-configured allow rules*

The test procedure includes:

- ❑ File type enumeration
- ❑ Active content protection
- ❑ Administration Configuration
- ❑ Logging

## 2.1 File Type Enumeration

### **Version 5.6**

SurfinGate determines the file type of executables and documents (e.g. Microsoft Word or Excel) simply by its file extension. The content of those documents itself is not analyzed. Furthermore archive files, like ZIP, are not taken apart for analysis.

Therefore it is very easy to bypass SurfinGate by just renaming such a file to an uncommon extension or wrapping it into a ZIP archive.

Since the file type is only determined by extension and no default deny-all-rule can be defined blocking all type of extensions is almost impossible.

- Protection is therefore limited to the extensions defined in the executable and document policy.

### **Version 6,x**

Some quick tests have shown that Finjan SurfinGate 6.xx has still the file content detection and blocking problems as in 5.6

## 2.2 Active Content

### **Version 5.6**

SurfinGate analyses ActiveX, Java Scripts, Visual Basic Scripts and Java applets with a parsing only technique. It searches for critical operations and system calls and then removes any content that does not comply with the policy defined. The code is therefore not executed in a sandbox. Further tests have shown that the JavaScript parser does not understand the JavaScript syntax. The simple inclusion of the word “hidden” causes a JavaScript to be removed.

Analysis has revealed that SurfinGate can be easily bypassed by obfuscating the critical code. For example: Critical JavaScript code was embedded in a HTML page as a HEX-encoded string together with a decoding function. When executing the JavaScript in the browser the HEX string is decoded into a normal string and feed to JavaScript’s eval() function. Since eval() is not blocked and SurfinGate uses a parsing only technique the critical code can successfully execute. SurfinGate does not offer a feature to deny execution of the eval() function. Similar properties have been discovered for Java applets.

- Protection with SurfinGate is the best when the critical code is not obfuscated in any way. When an attacker obfuscates his code SurfinGate will not be able to detect it.

**Version 6.x**

Some quick tests have shown that Finjan SurfinGate is now capable of analyzing JavaScript to some extent. The obfuscation technique used in 5.6 did not work with this version. No deeper tests have been conducted on how good the Finjan SurfinGate engine actually is.

## 2.3 Administration Configuration

Finjan SurfinGate offers a graphical console application that allows configuring and monitoring the content filter.

**Configuration**

Computer systems can be grouped together hierarchically and content rules can be defined on each node of the tree. This allows an administrator to define company wide policies that can be changed on each level on the tree.

If a company has high security needs it should think about having different policies for their users. But wild growth of dozens of policy should be avoided since administration becomes a nightmare.

For SurfinGate to handle this correctly the different user groups must be determinable by different IP addresses.

**File Content**

SurfinGate does not have the capability of white-listing file extensions and content types. This makes it difficult to block extensions which are unknown.

A company deploying SurfinGate should think about what file downloads are really essential for the operation of their business. Since SurfinGate does not allow white listing of content *block as many critical file extensions as you can.*

### **Active Content**

For different types of active content (JavaScript, Java, etc) the same GUI is used for configuration. The rules behave more like macro rules that group together different triggers. It is hard to associate a rule with the actual implementation of the code.

- There is no way to have a look which code statements trigger a rule and there is no way to add custom triggers. This would for example be useful for e.g. to block JavaScript's eval() function.

In general the rules for untrusted sites *must be set to very strict levels*. Although this does not give you full protection it is still more than none.

In particular deny active code access to:

- File system
- Network
- Windows registry
- Other applications.

If the user's surfing experience for often-visited sites is negatively influenced the administrator should consider reviewing the content of that site and add a white list URL where he weakens the rule.

### **Logging**

On blocking of content an entry in the log file is created. The entry shows the time, URL, security violation type. But there is no function that allows the administrator to see which part in the active content actually triggered filtering.

- Adapting SurfinGate's policy configuration is not a trivial task since logging is not very detailed.

## **2.4 Limitations**

### **HTTPS**

The content filter cannot inspect content wrapped in HTTPS. Therefore HTTPS connections must be blocked at the content filter. Otherwise the use of a content filter does not make sense.

On the Internet there are several websites available that allow anonymous surfing. They are often accessed by HTTPS. This way a content filter can be easily bypassed.




Here are examples of sites that allow anonymous surfing:




- <http://www.anonymizer.com>



### 3 Security Considerations

The table(s) in this chapter summarize the security issues found during the security review or penetration test. A definition for each column is given here:

Nr.	Weakness	Threat	Elimination	Rating
Each issue is consecutively numbered.	Explains the vulnerability or weakness found during testing.	Explains what could happen if the weakness is exploited.	Recommendation on how to correct the vulnerability.	Compass rating of the weakness and the corresponding threat:  : Low  : Medium  : High

Nr	Weakness	Threat	Elimination	Severity
1	Finjan solely relays on the file extension to determine content type of a file.	Any type of file can be downloaded if no filter has defined for it.	In Finjan itself: None. Actually there is no deny all rule for unknown file extensions.	
2	Finjan's JavaScript code analyzer is a parser. The code is not actively executed.	Finjan security check can be easily bypassed by obfuscating code and executing it with JavaScript's eval() function.	There is no way to block the invocation of the JavaScript eval() function.	
3	Finjan's Java applet code analyzer is a parser. The code is not actively executed.	Finjan security check can be easily bypassed by obfuscating code by utilizing Java's Reflection API for instance creation and method invocation.	There is no way to block the usage of the Java Reflection API.	

Nr	Weakness	Threat	Elimination	Severity
4	Usage of HTTP Tunnel does not cause an entry in the log file, although the tunnel cannot be established. This left the impression that the tunnel was just blocking accidentally.	HTTP Tunnels cannot be detected.	?	●●
5	Finjan triggers a “Forbidden Use Hidden Fields operation” if in the <SCRIPT> section the word “hidden” occurs.	Since false alarms are triggered the administrator could be forced to permit this operation.	?	●●
6	Passwords are weakly protected.	If a user has access to the storage location of the password the password can easily be decoded.	?	●●
7	Usage of HTTP Tunnel client causes Finjan to invoke a connection to the external site, but did not return the response to the client.	Tunnels are not blocked at first request.	?	●
8	If a JavaScript has been blocked there is no functionality that allows an administrator to have a look which code triggered the filtering.	For administrator fine-tuning the content filter is really hard.	?	●
9	No verbose logging can be activated where also allowed content is displayed.	Pinning down problems in the infrastructure is very difficult.	Use a network sniffer for problem solving.	●



## 4 Test Cases

The following chapters documents the tests that have been conducted for putting together this document. *The tests have not been exhaustive.*

### 4.1 HTML

<b>Purpose</b>	Find out how Surfingate reacts to malicious or abusive HTML pages.
<b>Set up</b>	Browsers: <ul style="list-style-type: none"> <li>• Mozilla 1.0 RC3</li> <li>• Internet Explorer 6.0.2600.0000, Q321232</li> </ul> Finjan: <ul style="list-style-type: none"> <li>• No special settings required.</li> </ul>

#	Description Test Case	Expected Result	Actual Result	✓ x	Pass Fail
1	DOS: HTML-Document with a refresh META-Tag set to 0 seconds.	Must Block	Not Filtered.	FAIL	
2	DOS: Infinite recursive frame sets. Test with Mozilla/Netscape	May Block	Not Filtered.	FAIL	
3	DOS: Infinite recursive iframes. Test with Mozilla/Netscape	May Block	Not Filtered.	FAIL	

## 4.2 JavaScript / VBScript

<b>Purpose</b>	Find out how SurfingGate reacts to JavaScript/VBScript that do malicious operations.
<b>Set up</b>	Browser: <ul style="list-style-type: none"> <li>Internet Explorer 6.0.2600.0000, Q321232</li> </ul>

#	Description Test Case	Expected Result	Actual Result	<span style="color: green;">✓</span> Pass <span style="color: red;">✗</span> Fail
1	JavaScript opening single pop up window.	Allow	Not blocked.	<b>PASS</b>
2	JavaScript opening multiple pop up windows in a loop	May Block	Not blocked.	<b>FAIL</b>
3	Same as #2, but code obfuscated.	May block	Not blocked.	<b>FAIL</b>
4	JavaScript opening single pop up window where toolbar is hidden. Finjan: Disable Permission "Remove Browser Element (menu, etc...)"	Must block	Blocked	<b>PASS</b>
5	Same as #4, but code obfuscated.	Must block	Blocked	<b>FAIL</b>
6	JavaScript that dynamically adds hidden fields to a form.	Must block	Blocked	<b>PASS</b>
7	Put the word "hidden" somewhere in the <SCRIPT>-section	The content filter must be context sensitive.	Every JavaScript which contains the word hidden is removed from the web page.	<b>FAIL</b>

### Result:

- Finjan just parses the JavaScript for keywords, but does not actually execute the code.
- Through obfuscating the code and using the eval() function, Finjan can be easily bypassed.
- Finjan does not really understand JavaScript syntax.

#### Detail #4: Original HTML page

```
<HTML>
<HEAD>
  <TITLE>Compass Security - Content Filter Testing - JavaScript</TITLE>
</HEAD>

<BODY>
<H1>Pop Up Window Without Toolbar</H1>
<P>
By clicking the button a single pop up window without toolbars.
</P>

<SCRIPT>
<!--
function popUpWindow() {
  window.open("PopUpWindow.html", "_blank", "toolbar=no");
}
-->
</SCRIPT>

<FORM>
  <INPUT TYPE="button" value="Open Pop Up Window Without Toolbar"
onClick="popUpWindow()"></INPUT>
</FORM>

</BODY>
</HTML>
```

#### Detail #4: HTML page after Finjan filtering:

```
<HTML>
<HEAD>
  <TITLE>Compass Security - Content Filter Testing - JavaScript</TITLE>
</HEAD>
<BODY>
<H1>Pop Up Window Without Toolbar</H1>
<P>
By clicking the button a single pop up window without toolbars.
</P>

<FORM>
  <INPUT TYPE="button" value="Open Pop Up Window Without Toolbar"
></INPUT>
</FORM>

</BODY>
</HTML>
```

Detail #7:

This simple script triggers a “Forbidden Use Hidden Fields operation” error and the script section gets removed. If the word “hidden” is replaced with some other word, like “blabli” the content filter lets the script pass. This means that the parser for this error does not really understand the JavaScript syntax.

```
<HTML>
<HEAD>
  <TITLE>Compass Security - Content Filter Testing</TITLE>
</HEAD>
<BODY>
<SCRIPT>
document.write("hidden");
</SCRIPT>
</BODY>
</HTML>
```

### 4.3 VBScript

#	Description Test Case	Expected Result	Actual Result	✓ Pass x Fail
1	VBScript which lists the content of the root directory.	Must Block	Blocked. Finjan Log: Operating System violation. Forbidden Access Other Applications operation.	PASS

#### Detail #1: Original HTML page

```

<HTML>
<HEAD>
<TITLE>Compass Security - Content Filter Testing - VBScript</TITLE>
</HEAD>
<BODY>
<H1>Displays the Content of the Root Directory</H1>

<P> This web page will list the files in your root directory if the
Scripting Run-time Library is installed and registered. If it is not
installed and registered, this web page generates a script error.
<P>

<H2>Files In The Root Directory (\)</H2>

<SCRIPT language="VBScript">
<!--

' This code should create a file system object, open the root file system
' of the current drive and display the list of files found there in the
' web page. If the Scripting Run-time Library is not installed, this
' script will fail on the line containing "CreateObject".

Dim fso, f, fl, fc
Set fso = CreateObject("Scripting.FileSystemObject")
Set f = fso.GetFolder("\")
Set fc = f.Files
For Each fl in fc
    document.write fl.name & "<br>"
Next

-->
</SCRIPT>

</BODY>
</HTML>

```

Detail #1: After Finjan filtering

```

<HTML>
<HEAD>
  <TITLE>Compass Security - Content Filter Testing - VBScript</TITLE>
</HEAD>

<BODY>
<H1>Displays the Content of the Root Directory</H1>

<P> This web page will list the files in your root directory if the
Scripting Run-time Library is installed and registered. If it is not
installed and registered, this web page generates a script error.
<P>

<H2>Files In The Root Directory (\)</H2>

</BODY>
</HTML>

```

#### 4.4 EML

<b>Purpose</b>	Find out how Surfingate handles EML files? Can it detect EML files in Archives?
<b>Set up</b>	Browsers: <ul style="list-style-type: none"> <li>• Mozilla 1.0 RC3</li> <li>• Internet Explorer 6.0.2600.0000, Q321232</li> </ul> Finjan: <ul style="list-style-type: none"> <li>• No special settings required.</li> </ul>

#	Description Test Case	Expected Result	Actual Result	✓ x	Pass Fail
1	Download a Plain eml-file	Must Block	Not blocked	FAIL	
2	Download a EML file in a ZIP archive	Must Block	Not blocked	FAIL	
3	Download a EML file in a ZIP archive where the archive is protected with the password "gugus"	Must Block	Not blocked	FAIL	
4	Download a EML File in TAR archive	Must Block	Not blocked	FAIL	

Result:

- Content of file is not analyzed.

#### 4.5 EICAR Virus Pattern String

<b>Purpose</b>	Find out how SurfingGate reacts to the EICAR test strings.
<b>Set up</b>	Browsers: <ul style="list-style-type: none"> <li>Internet Explorer 6.0.2600.0000, Q321232</li> </ul> SurfingGate: <ul style="list-style-type: none"> <li>Enable executable blocking in Policy.</li> </ul>

#	Description Test Case	Expected Result	Actual Result	✓ x	Pass Fail
1	Download eicar.com	Must Block	HTTP 403 (Forbidden) page. Filtered, because detected as executable.	PASS	
2	Download eicar.com.text	Must Block	Text String is displayed in the browser	FAIL	
3	Download eicar_com.zip	Must Block	Download possible	FAIL	
4	Download eicarcom2.zip	Must Block	Download possible	FAIL	

**Result:**

- #1 was blocked because in the Executable Policy the extension COM is defined.
- Content of files with extension TXT, ZIP is not analyzed.

**Counter measures:**

- Define Rules in the Document Policy to filter archive extensions, like ZIP, RAR, etc.

#### 4.6 EXE-Files

<b>Purpose</b>	Find out how SurfingGate reacts to EXE-files, either bare or packed into a ZIP archive.
<b>Set up</b>	Browser: <ul style="list-style-type: none"> <li>Internet Explorer 6.0.2600.0000, Q321232</li> </ul> SurfingGate: <ul style="list-style-type: none"> <li>Enabled executable blocking in Policy</li> </ul>

#	Description Test Case	Expected Result	Actual Result	✓ x	Pass Fail
	Download EXE file.	Must Block	HTTP 403 (Forbidden) page. Blocked because of Executable Policy	PASS	
	Download EXE file wrapped in ZIP	Must Block	Does not block.	FAIL	
	Download EXE file wrapped in RAR	Must Block	Does not block.	FAIL	
	Download SCR file	Must Block	HTTP 403 (Forbidden) page. Blocked because of Executable Policy	PASS	
	Download SCR file wrapped in ZIP	Must Block	Does not block.	FAIL	
	Download a CPL file.	Must Block	HTTP 403 (Forbidden) page. Blocked because of Executable Policy	PASS	
	Download a CPL file wrapped in ZIP	Must Block	Does not block	FAIL	
	Download an EXE-File which was renamed to an unknown file type	Must Block	Does not block.	FAIL	

Result:

- Content of files with extension TXT, ZIP is not analyzed.

Counter measures:

- Define rules in the Document Policy to filter archive extensions, like ZIP, RAR, etc.

#### 4.7 Office Documents

<b>Purpose</b>	Find out how SurfingGate reacts to Office Documents.
<b>Set up</b>	Browsers: <ul style="list-style-type: none"> <li>Internet Explorer 6.0.2600.0000, Q321232</li> </ul>

#	Description Test Case	Expected Result	Actual Result	✓ x	Pass Fail
1	Acrobat PDF document	Allow	Not blocked	PASS	
2	Word document without any Visual Basic Code. Finjan: Block Word Documents in Document Policy	May Block	Blocked	PASS	
3	Same as #2. Finjan: Block application auto-launch for Word documents in Document Policy.	?	Document downloaded and automatically started.	N/A	
4	Same as #2. Finjan Allow Word documents.	Allow	Document downloaded and automatically started.	PASS	
5	Word document wrapped in ZIP. Finjan: Block Word document download.	Must Block	Not blocked.	FAIL	
6	Word document wrapped in RAR. Finjan: Block Word document download.	Must Block	Not blocked.	FAIL	
7	Word document where clock.exe was embedded. Finjan: Block Word document download.	Must Block	Blocked HTTP 403.	PASS	
8	Same as #7, but wrapped in ZIP file. Finjan: Block Word document download	Must Block	Not blocked.	FAIL	
9	Same as #7, but extension renamed to REN.	Must Block	Not blocked.	FAIL	
10	Same as #2, but extension are renamed to REN	Must Block	Not blocked.	FAIL	

Result:

GLÄRNISCHSTR. 7  
POSTFACH 1671  
CH-8640 RAPPERSWIL

Tel. +41 55-214 41 60  
Fax +41 55-214 41 61  
info@csnc.ch www.csnc.ch

- #1 was blocked because in the Executable Policy the extension COM is defined.
- Content of files with extension ZIP is not analyzed.
- File extension of file is solely used for content type analysis. The content itself is not analyzed. This way Finjan can be bypassed by just changing file extension.

Counter measures:

- Define Rules in the Document Policy to filter archive extensions, like ZIP, RAR, etc.

#### 4.8 Java Applets

<b>Purpose</b>	Find out how Surfingate: <ul style="list-style-type: none"> <li>• Reacts to Java applets that do malicious operations?</li> <li>• Scans Java applets? Is the code really executed in a sandbox or is the code just parsed?</li> </ul>
<b>Set up</b>	Browser: <ul style="list-style-type: none"> <li>• Internet Explorer 6.0.2600.0000, Q321232</li> <li>• Java Runtime Environment 1.3 installed</li> <li>• Sandbox security in File java.policy weakened.</li> </ul>

#	Description Test Case	Expected Result	Actual Result	✓ x	Pass Fail
1	A Java applet that lists the content of C:\ No Java Reflection used.	Block	Blocked	PASS	
2	Same as #1. But use Java Reflection to hide the reference to java.io.File.	Block	Not blocked: SurfinGate is not able to detect usage of Java Reflection.	FAIL	
3	A Java applet that connects to <a href="http://www.csnc.ch">http://www.csnc.ch</a> by using java.net.URLConnection	Block	Blocked	PASS	
4	Same as #3: But use Java Reflection to hide the references to java.net.URL and java.net.URLConnection	Block	Not blocked: SurfinGate is not able to detect usage of Java Reflection.	FAIL	

Result:

- SurfinGate is not able to detect the usage of the Java Reflection API and therefore it can be bypassed.
- All files in a JAR-archive are scanned for critical code. If a class is found that is on the black list the class is removed from the JAR archive. If the applet is broken by removing the class SurfinGate replaces the whole applet with a applet containing a information message for the user.

#### 4.9 Inside-Out Attacks

<b>Purpose</b>	Find out how to transfer data from inside the Intranet into the Internet.
<b>Set up</b>	

#	Description Test Case	Expected Result	Actual Result	✓ x	Pass Fail
1	Try to build a tunnel in order to get a shell (cmd.exe). Experiment with different transfer mechanisms.	Block	not tested yet		
2	Try to build a tunnel by using the CONNECT command of the Proxy.	Block	not tested yet		
3	Post a Text-File to a Website.	Block	not tested yet		

#### 4.10 ActiveX

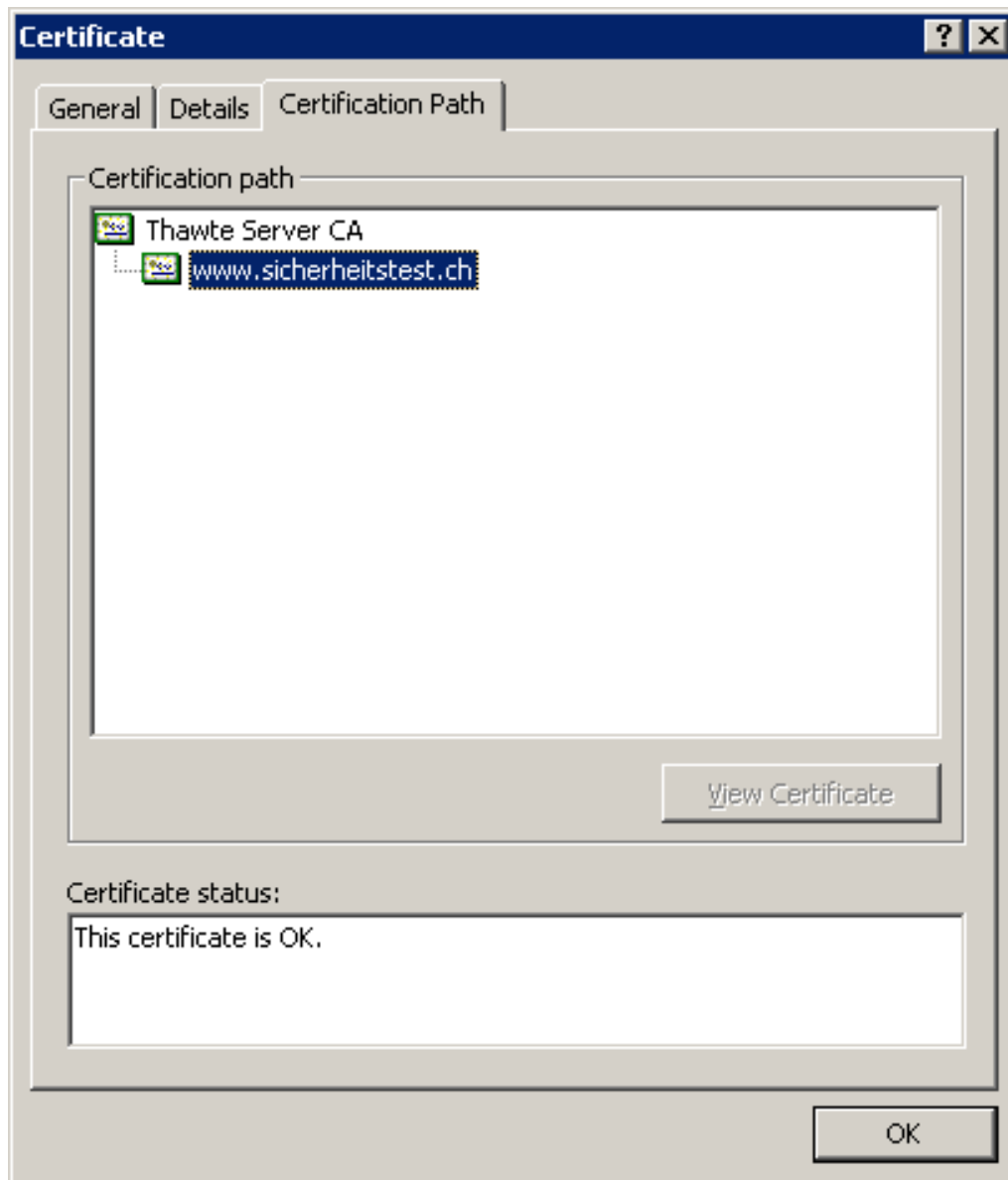
#	Description Test Case	Expected Result	Actual Result	✓ x	Pass Fail
1	<a href="http://www.sicherheitstest.ch/comp/tests/testnt.shtml">http://www.sicherheitstest.ch/comp/tests/testnt.shtml</a>	Block	Blocked. Finjan Log: "Operating System Violation/Forbidden Terminate Process operation."	<b>PASS</b>	

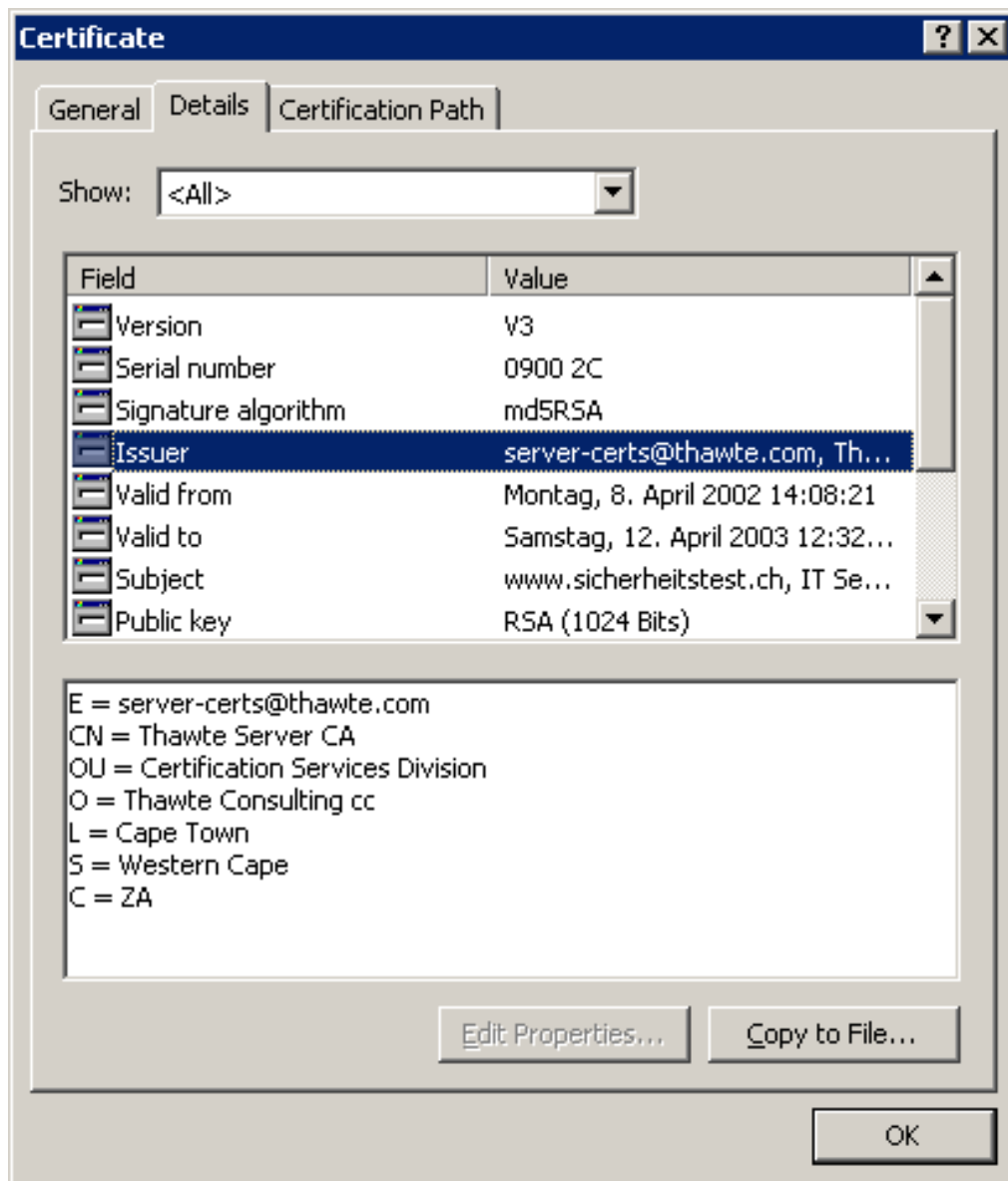
#### 4.11 HTTPS connections

<b>Purpose</b>	Find out if SurfinGate inspects HTTPS traffic?
<b>Set up</b>	Browsers: Internet Explorer 6.0.2600.0000, Q321232

#	Description Test Case	Expected Result	Actual Result	✓ x	Pass Fail
1	Finjan: Disable HTTPS blocking Access <a href="https://www.sicherheitstest.ch">https://www.sicherheitstest.ch</a>	Allow	Allowed. SurfinGate does not analyze HTTPS content.		PASS
2	Finjan: Enable HTTPS blocking Access <a href="https://www.sicherheitstest.ch">https://www.sicherheitstest.ch</a>	Block	Blocked. HTTP 501 "Not implemented"		PASS

Details #1:





## 5 Configuration Database

SurfinGate stores the content filter configuration in a database.

On the Windows platform the default installed database is a Microsoft Access file called SFGDatabase.mdb. The database protection password is 'FinjanBabe'.

On Solaris an Oracle Database Version 8.1.7 is installed with SurfinGate.

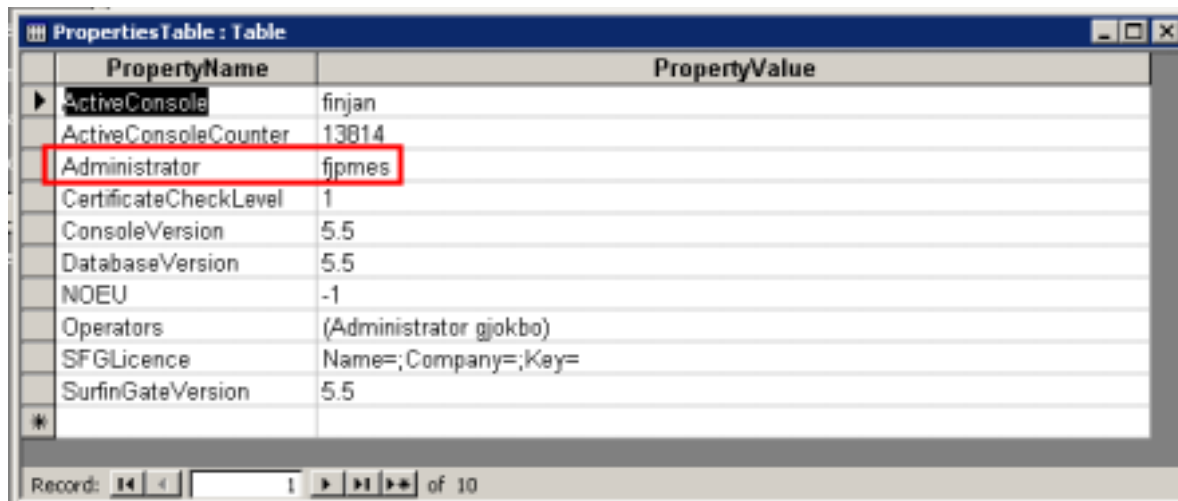
### 5.1 Administrator Console Password

The password that must be used to access the console is stored obfuscated in the PROPERTIESTABLE. In this example the password is 'finjan'.

Algorithm for password encryption:

$$\text{CHAR}_{\text{encrypted}}(n) = \text{CHAR}(\text{ACSCII}(\text{CHAR}_{\text{cleartext}}(n)) + n)$$

where n is the position of the character in the password beginning with 0.



PropertyName	PropertyValue
ActiveConsole	finjan
ActiveConsoleCounter	13814
Administrator	fjpmes
CertificateCheckLevel	1
ConsoleVersion	5.5
DatabaseVersion	5.5
NOEU	-1
Operators	(Administrator gjokbo)
SFGLicence	Name=; Company=; Key=
SurfinGateVersion	5.5

## 5.2 Oracle Password

If the SurfinGate console accesses a Solaris installation it uses the Oracle proprietary network protocol. To access the Oracle database the user must be authenticated with an Oracle user. The user credentials for Oracle are stored in a SurfinGate configuration file:

```
SurfinConsole.cfg:

# Configuration parameters.
# Mon Jul 29 17:27:09 2002
console_owner=SurfinGate
use_talk_back=false
db_language=FOCI
max_log_view=150
log_refresh=10
db_url=SFGDatabase
version=5.6
db_subprotocol=FINJAN:OCI7
build=247.0.1
db_user_name=surfinGate
db_password=676A6F337468
```

Algorithm for password encryption:

$$\text{CHAR}_{\text{encrypted}}(n) = \text{HEX}(\text{ASCII}(\text{CHAR}_{\text{cleartext}}(n)) + 1)$$

where n is the position of the character in the password.