SurfinGate 5.6
Security Target

ST Version 1.7
October 25, 2001

Prepared for:
Finjan Software Incorporated
2806 Zanker Road, Suite 201
San Jose, CA 951345

Prepared By:
Science Applications International Corporation
7125 Gateway Drive, Suite 300
Columbia, MD 21046

1. Security Target Introduction

This section identifies the Security Target (ST) and Target of Evaluation (TOE) identification, ST conventions, ST conformance claims, and the ST organization.

1.1 Security Target, TOE and CC Identification

ST Title – SurfinGate Security Target

ST Version – Version 1.7

TOE Identification – SurfinGate, Version 5.6

PP Identification – None

CC Identification – Common Criteria for Information Technology Security Evaluation, Version 2.1, August 1999, ISO/IEC 15408.

Keywords – malicious mobile code, filter, ActiveX, Java, VBScript, and JavaScript

1.2 CC Conformance Claims

This TOE conforms to the following specifications:

- Common Criteria for Information Technology Security Evaluation Part 2: Security functional requirements, Version 2.1, August 1999, ISO/IEC 15408-2.
- Common Criteria for Information Technology Security Evaluation Part 3: Security assurance requirements, Version 2.1, August 1999, ISO/IEC 15408-3, Evaluation Assurance Level 3.

1.3 Strength of Environment

SurfinGate provides for a level of protection that is appropriate for IT environments that require detection of malicious mobile code where SurfinGate can be appropriately protected from hostile attacks. The assurance requirements and the minimum strength of function were chosen to be consistent with that level of risk identified in the Mobile Code: Enclave Protection Profile (MCEPP). The assurance level is EAL 3 and the minimum strength of function is SOF-medium.

1.4 Conventions, Terminology, Acronyms

This section specifies the formatting information used in the Security Target.

1.4.1 Conventions

The following conventions have been applied in this document:

- Security Functional Requirements – Part 2 of the CC defines the approved set of operations that may be applied to functional requirements: iteration, assignment, selection, and refinement.
    - Iteration: allows a component to be used more than once with varying operations. In the ST, iteration is indicated by a letter in parenthesis placed at the end of the component. For example FDP_ACC.1 (a) and FDP_ACC.1 (b) indicate that the ST includes two iterations of the FDP_ACC.1 requirement, a and b.
    - Assignment: allows the specification of an identified parameter.

  - Selection: allows the specification of one or more elements from a list.
  - Refinement: allows the addition of details.

  The conventions for the assignment, selection and refinement operations are described in section 5.

- Other sections of the ST – Other sections of the ST use bolding to highlight text of special interest, such as captions.

1.4.2 Terminology

The following terminology is used in the Security Target:

Audit – The term audit is used to maintain consistency with the Common Criteria. Usual Finjan product literature uses the term log when describing the act of recording.

Executable - text or data embedded in or bound to documents or other data sets, and executed without explicit end-user initiation.

Host Operating System – The operating systems on which SurfinGate is installed.

Inspection – The state where mobile code will be examined by SurfinGate to determine if it meets the configured policy.

Mobile code - text or data conveyed over a computer network, embedded in or referenced from other network data, and executed without explicit end-user initiation on the recipient platform.

Response data – Web-based traffic requested by clients and received by SurfinGate.

Web-based traffic – network packets sent or received via a web browser interface.

1.4.3 Acronyms

The acronyms used in this Security Target are specified in Appendix A – Acronym List.

1.5 Security Target Overview and Organization

The SurfinGate Target of Evaluation (TOE) provides gateway security for malicious Web content including ActiveX, Java, VBScript and JavaScript. SurfinGate uses a sophisticated real-time, content-inspection process to identify and block malicious mobile code. SurfinGate provides central management to allow administrators to easily manage the filtering policies. The SurfinGate TOE provides the following security services: audit, user data protection, identification and authentication, security management, and protection of the TOE Security Functions (TSF).

The SurfinGate Security Target contains the following additional sections:

- TOE Description (Section 2)
- Security Environment (Section 3)
- Security Objectives (Section 4)
- IT Security Requirements (Section 5)
- TOE Summary Specification (Section 6)
- Protection Profile Claims (Section 7)
- Rationale (Section 8).

2. TOE Description

The TOE includes three components: SurfinGate Server, SurfinConsole, and the SurfinGate Database. The SurfinGate Server performs the malicious mobile code collection and analysis. The SurfinConsole is the management interface and the SurfinGate Database stores details of scanned code, security policy and logged events. The TOE and the SurfinGate product are equivalent terms.

2.1 Product Description

SurfinGate is a malicious mobile code detection and response system. It provides services to collect web-based traffic, analyze the collected web-based traffic against an administrator-defined policy, and respond to any security violations detected. The SurfinGate product consists of three components - SurfinGate Server, SurfinConsole, and the SurfinGate Database. There may be one or more SurfinGate Server components in the evaluated configuration.

The SurfinGate Server is a proxy that runs on Windows NT and Sun Solaris. The proxy intercepts all web-based traffic and performs a malicious mobile code analysis upon the web-based traffic. The types of analysis supported include: content inspection of ActiveX, Java, Visual Basic Script and JavaScript, URL filtering, and file extension filtering. Any code that violates SurfinGate's security policy is denied access to the network.

The SurfinGate Database provides a repository for the security policy and security violations. The SurfinGate Database runs on the same platform as the SurfinGate Server. The SurfinGate Server stores all of its analysis results in the Database and receives all policy information from the database. The SurfinGate Server receives policy updates on a regular basis reflecting changes the administrator makes to the policy stored in the database.

The SurfinConsole is a central tool for managing the security policies, controlling multiple SurfinGate servers and generating audit reports. The SurfinConsole runs on a Windows NT 4.0 Service Pack 4 or above.

2.2 Security Environment TOE Boundary

The TOE includes both physical and logical boundaries. Its operational environment is that of a networked environment with web-based traffic.

2.2.1 Logical Boundaries

SurfinGate is a policy-based, malicious mobile code detection system. It contains three components, SurfinGate Server, SurfinConsole, and the SurfinGate Database. Figure 1 SurfinGate shows the relationship among the components.

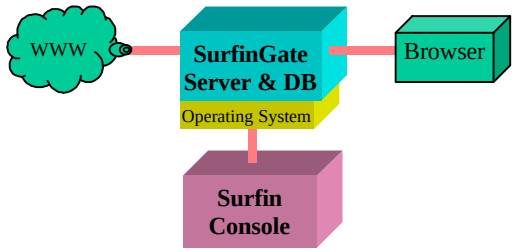

Figure 1 SurfinGate Components

In the evaluated configuration, all SurfinGate components run in a separate DMZ (Demilitarized Zone) segment of the network. A DMZ is typically created on a segment attached to a separate NIC on the firewall, or between two routing devices (such as router or firewall); see Figure 2. Typically, the firewall would be configured to allow internal users to access the SurfinGate server only to port 8080 (default proxy port), and block external access to the SurfinGate server, allowing the SurfinGate server to communicate to the outside over HTTP protocol (outbound only). The firewall needs to be configured to send outbound traffic to the mail sever to permit alert message to the administrator.

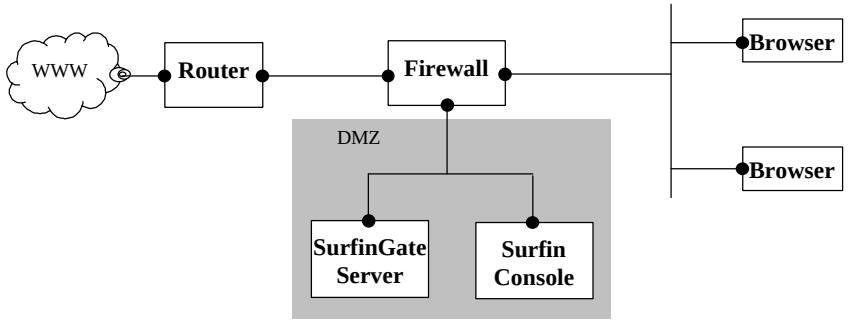

Figure 2 SurfinGate Security Architecture

The SurfinGate Server interacts with the operating system to collect the web-based traffic. The SurfinGate Server, and SurfinGate Database work together to perform scanning of all the web-based traffic received from the network. The SurfinConsole runs on a separate platform and is used to manage the SurfinGate Server, and SurfinGate Database. The remainder of this section described the security services provided by SurfinGate.

2.2.1.1 Audit

The SurfinGate product has the ability to audit and filter all web-based traffic. The SurfinGate product ensures the audit trail is protected and only administrators may view the audit data.

2.2.1.2 Information Flow Protection

SurfinGate has the ability to filter web-based traffic searching for the following types of malicious mobile code:

- ActiveX,
- Java,
- Visual Basic Script, and
- JavaScript.

In addition to potential malicious mobile code, SurfinGate can filter web-based traffic based on the following criteria:

- File Extensions, and
- URLs.

With the collected web-based traffic, SurfinGate analyzes the data to see if it matches any identified risks in the security policy. The security policy is an administrator defined policy that specifies what types of data to filter and what action to take if data violates the security policy. The security violations are logged in a database (i.e., audit trail) for future analysis.

2.2.1.3 Authentication

The administrator is the only user that directly accesses the SurfinGate product. The administrator is required to perform password authentication before accessing SurfinGate.

2.2.1.4 Security Management

SurfinGate includes a number of management functions to control access to the system and to manage the data collection and analysis. The management functions include configuring the security policy that determines what information will be filtered and audited. The management functions are controlled through possession of the administrator password.

2.2.1.5 TOE Protection Mechanisms

SurfinGate provides for a secure recovery if it crashes due to a power failure, operating system failure, or hardware failure.

2.2.2 Physical Boundaries

SurfinGate is an embedded product with no physical boundaries it controls. Within the evaluation, SurfinGate will be evaluated in the following two configurations:

1. SurfinGate Server and Database on the same Windows NT operating system with an Access database and with a SurfinConsole on a Windows NT operating system.
2. SurfinGate Server and Database on the same or different Sun Solaris operating systems with an Oracle database and with a SurfinConsole on a Windows NT operating system.

3. Security Environment

The TOE security environment consists of the threats to security, organizational security policies, and usage assumptions as they relate to SurfinGate.

SurfinGate provides for a level of protection that is appropriate for IT environments that require detection of malicious mobile code where SurfinGate can be appropriately protected from hostile attacks. SurfinGate is not designed to resist direct, hostile attacks; therefore, it must be embedded in or protected by other products designed to address threats that correspond with the intended environment. SurfinGate is suitable for use in both commercial and government environments.

3.1 Threats to Security

T.UNAUTH	Malicious mobile code may enter the IT System monitored by the TOE undetected.
T.FALASC	The TOE may fail to identify malicious mobile code based on data received.
T.FALACT	The TOE may fail to react to identified or suspected malicious mobile code.
T.IMPCON	An unauthorized user may inappropriately change the configuration of the TOE.
T.LOSSOF	An unauthorized user may attempt to remove or destroy data collected and produced by the TOE.
T.NOFAIL	After a system failure (i.e., power failure, hardware failure, and operating system failure), the TOE is restored to a permissive state and allows web-based traffic that would have been denied otherwise.

3.2 Organization Security Policies

An organizational security policy is a set of rules, practices, and procedures imposed by an organization to address its security needs. This section identifies the organizational security policies applicable SurfinGate.

P.ACCESS	All data collected and produced by the TOE shall only be used for authorized purposes.
P.IDAUTH	Administrators must authenticate before accessing any TOE functions or data.
P.MANAGE	The TOE shall provide a set of administrative tools to manage the TOE's functions and data.

3.3 Secure Usage Assumptions

This section contains assumptions regarding the security environment and the intended usage of the TOE.

3.3.1 Personnel Assumptions

A.MANAGE	There will be one or more competent individuals assigned to manage the TOE and the security of the information it contains.

A.NOEVIL	The administrators are not careless, wilfully negligent, or hostile, and will follow and abide by the instructions provided by the TOE documentation.

3.3.2 Physical Assumptions

A.LOCATE	The processing resources of the TOE will be located within controlled access facilities, which will prevent unauthorized physical access.
A.PROTCT	The TOE hardware and software critical to security policy enforcement will be protected from unauthorized physical modification.

3.3.3 Host Operating System and Configuration Assumptions

A.FWALL	A firewall will direct all web-based traffic through the SurfinGate product.
A.DMZ	SurfinGate will reside in a Demilitarized Zone (DMZ) to protect it from direct attacks.
A.APPS	SurfinGate will be the only application running on its host server.
A.MAIL	The mail server on the SurfinGate network will accept only outgoing mail from the SurfinGate product and will deliver mail properly.
A.ADMINS	The host operating system for SurfinGate will only permit access by trusted users.
A.I&A	Users of the host operating system are identified and authenticated.
A.SEP	The host operating system will provide mechanisms to isolate the TOE Security Functions (TSF) and assure that TSF components cannot tampered with or bypassed. The TSF components are 1) the files used by SurfinGate to store its data and 2) the TOE processes managing the mobile code scanning.
A.FILES	All of the SurfinGate related files and directories (including executables, run-time libraries, audit logs) are protected from unauthorized access by the host operating system control mechanisms.
A.TIME	The host operating system will provide a reliable timestamp.

4. Security Objectives

This section defines the security objectives of SurfinGate and its supporting environment. Security objectives reflect the stated intent to counter identified threats and/or comply with any organizational security policies identified. All of the identified threats and organizational policies are addressed under one of the categories below.

4.1 IT Security Objectives

O.MEDIAT	The TOE will scan all data received.
O.DETECT	The TOE will detect malicious mobile code.
O.REACT	The TOE will react to any malicious mobile code in accordance with the TSP to alert the administrator of a potential security violation.
O.ACCESS	The TOE must allow authorized users to access only appropriate TOE functions and data.
O.EADMIN	The TOE must include a set of functions that allow effective management of its functions and data.
O.IDAUTH	The TOE must be able to authenticate administrators prior to allowing access to TOE functions and data.
O.FAIL	The TOE must provide a means for secure recovery from failures.

4.2 TOE Non-IT Security Objectives

The TOE's general operating environment must satisfy the following objectives. These objectives do not levy any IT requirements but are satisfied by procedural or administrative measures.

O.CREDEN	Those responsible for the TOE must ensure that all access credentials are protected by the users in a manner that is consistent with IT security.
O.INSTAL	Those responsible for the TOE must ensure that the TOE is delivered, installed, managed, and operated in a manner which is consistent with IT security.
O.PERSON	Personnel working as authorised administrators shall be carefully selected and trained for proper operation of the TOE.
O. PHYCAL	Those responsible for the TOE must ensure that those parts of the TOE critical to security policy are protected from any physical attack.

4.3 Security Objectives for the IT Environment

The host operating system shall support the security objectives of the TOE as follows:

O.OSAUTH	The host operating system shall identify and authenticate users prior to providing access to any TOE facilities.

O.ACCESS	The host operating system shall provide the access control mechanisms required to protect the TOE's data and system files.
O.TIME	The host operating system shall provide a reliable timestamp the TOE can use for accurately tracking audit events.
O.SEP	The host operating system shall provide mechanisms to isolate the TOE Security Functions (TSF) and assure that TSF components cannot tampered with or bypassed.
O.OSFAIL	The host operating system and database must provide a means for secure recovery from failures.

5. IT Security Requirements

5.1 TOE Security Functional Requirements

This section specifies the security functional requirements (SFRs) for the TOE. This section organizes the SFRs by CC class. The functional security requirements for the ST consist of the following SFRs, summarized in Table 1 Security Functional Requirements. All SFRs were drawn from part 2 of the Common Criteria.

Functional Components	
FAU_ARP.1	Security alarms
FAU_GEN.1	Audit data generation
FAU_SAA.1	Potential violation analysis
FAU_SAR.1	Audit review
FAU_SAR.2	Restricted audit review
FAU_SAR.3	Selectable audit review
FAU_STG.1	Protected audit trail storage
FDP_IFC.2	Complete information flow control
FDP_IFF.1	Simple security attributes
FIA_UAU.2	User authentication before any action
FIA_UAU.7	Protected authentication feedback
FMT_MOF.1	Management of security functions behaviour
FMT_MSA.3	Static attribute initialization
FMT_MTD.1	Management of TSF data
FPT_FLS.1	Failure with preservation of secure state

Table 1 Security Functional Requirements

The CC permits four functional component operations—assignment, refinement, selection, and iteration — to be performed on security functional requirements. This section highlights the four operations in the following manner:

- assignment: allows the specification of an identified parameter. Indicated with bold;
- refinement: allows the addition of details. Indicated with bold italics text;
- selection: allows the specification of one or more elements from a list. Indicated with underlined text; and
- iteration: allows a component to be used more than once with varying operations. Not used in this ST.

5.1.1 Security audit (FAU)

5.1.1.1 Security Alarms (FAU_ARP.1)

5.1.1.1.1 FAU_ARP.1.1

The TSF shall take an action to **block the execution of malicious mobile code, add a record to the audit log and alert the administrator if so indicated by the TSP** upon detection of a potential security violation.

5.1.1.2 Audit data generation (FAU_GEN.1)

5.1.1.2.1 FAU_GEN.1.1

The TSF shall be able to generate an audit record of the following auditable events:

a. Start-up and shutdown of the audit functions;

b. All auditable events for the not specified level of audit; and

c. **Client requests and detection of malicious mobile code in response data**.

5.1.1.2.2 FAU_GEN.1.2

The TSF shall record within each audit record at least the following information:

a. Date and time of the event, type of event, subject identity, and the outcome (success or failure) of the event; and

b. For each event type, the additional information specified below:

- **Client requests – Address or host name of client, request identification or URL**
- **Detection of malicious mobile code in response data – Type of mobile code detected**

5.1.1.3 Potential violation analysis (FAU_SAA.1)

5.1.1.3.1 FAU_SAA.1.1

The TSF shall be able to apply a set of rules in monitoring the audited events and based upon these rules indicate a potential violation of the TSP.

5.1.1.3.2 FAU_SAA.1.2

The TSF shall enforce the following rules for monitoring audited events:

a) Accumulation or combination of **detected mobile code violation from a single source** known to indicate a potential security violation;

b) **No other rules**.

Application Note: This requirement addresses SurfinGate's ability to maintain a set of known sources of malicious mobile code.

5.1.1.4 Audit review (FAU_SAR.1)

5.1.1.4.1 FAU_SAR.1.1

The TSF shall provide ***the*** **administrator** with the capability to read **all information** from the audit records.

5.1.1.4.2 FAU_SAR.1.2

The TSF shall provide the audit records in a manner suitable for the user to interpret the information.

5.1.1.5 Restricted audit review (FAU_SAR.2)

5.1.1.5.1 FAU_SAR.2.1

The TSF shall prohibit all users read access to the audit records, except those users that have been granted explicit read-access.

5.1.1.6 Selectable audit review (FAU_SAR.3)

5.1.1.6.1 FAU_SAR.3.1

The TSF shall provide the ability to perform searches and sorting of audit data based on **any audit field in the audit records**.

5.1.1.7 Protected audit trail storage (FAU_STG.1)

5.1.1.7.1 FAU_STG.1.1

The TSF shall protect the stored audit records from unauthorized deletion.

5.1.1.7.2 FAU_STG.1.2

The TSF shall be able to prevent modifications to the audit records.

5.1.2 User Data Protection

5.1.2.1 Complete information flow control (FDP_IFC.2)

5.1.2.1.1 FDP_ IFC.2.1

The TSF shall enforce the **mobile code flow policy** on **the subjects, namely client hosts, and information, namely response data, and** all operations that cause that information to flow to and from subjects covered by the SFP.

5.1.2.1.2 FDP_ IFC.2.2

The TSF shall ensure that all operations that cause any information in the TSC to flow to and from any subject in the TSC are covered by an information flow control SFP.

5.1.2.2 Simple security attributes (FDP_IFF.1)

5.1.2.2.1 FDP_IFF.1.1

The TSF shall enforce the **mobile code flow policy** based on the following types of subject and information security attributes: **mobile code type, identity of external source, signature verification status, and identity of client**.

5.1.2.2.2 FDP_IFF.1.2

The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold:

a) If the mobile code type is explicitly marked as permitted, then the transfer of response data is permitted.

b) If the mobile code type is marked for inspection, the response data is permitted if no malicious mobile code is detected.

c) If the mobile code type is marked as blocked, then the response data is rejected.

5.1.2.2.3 FDP_IFF.1.3

The TSF shall enforce the additional information flow control SFP rules: **none**.

5.1.2.2.4 FDP_IFF.1.4

The TSF shall provide the additional SFP capabilities: **none**.

5.1.2.2.5 FDP_IFF.1.5

The TSF shall explicitly authorize an information flow based on the following rule:

a) **The administrator may permit a client access to a type of mobile code.**

5.1.2.2.6 FDP_IFF.1.6

The TSF shall explicitly deny an information flow based on the following rule:

a) **The administrator may deny a client access to a type of mobile code.**

5.1.3 Identification and Authentication (FIA)

5.1.3.1 User authentication before any action (FIA_UAU.2)

5.1.3.1.1 FIA_UAU.2.1

The TSF shall require each user to be successfully authenticated before allowing any other TSF-mediated actions on behalf of that user.

5.1.3.2 Protected authentication feedback (FIA_UAU.7)

5.1.3.2.1 FIA_UAU.7.1

The TSF shall provide only **obscured feedback** to the user while the authentication is in progress.

5.1.3.3 Verification of Secrets (FIA_SOS.1)

5.1.3.3.1 FIA_SOS.1.1

The TSF shall provide a mechanism to verify that secrets meet the following:

For each attempt to use the authentication mechanism, the probability that a random attempt will succeed is less than one in 250,000,000,000,000.

5.1.4 SECURITY MANAGEMENT (FMT)

5.1.4.1 Management of security functions behavior (FMT_MOF.1)

5.1.4.1.1 FMT_MOF.1.1

The TSF shall restrict the ability to disable, enable, and modify the behavior of the **mobile code flow policy** to **the administrator**.

5.1.4.2 Static attribute initialization (FMT_MSA.3)

5.1.4.2.1 FMT_MSA.3.1

The TSF shall enforce the **mobile code flow policy** to provide restrictive default values for security attributes that are used to enforce the SFP.

5.1.4.2.2 FMT_MSA.3.2

The TSF shall allow the **administrator** to specify alternative initial values to override the default values when an object or information is created.

5.1.4.3 Management of TSF data (FMT_MTD.1)

5.1.4.3.1 FMT_MTD.1.1

The TSF shall restrict the ability to **query and add audit data**, and shall restrict the ability to **query and modify all other TOE data** to **administrators**.

5.1.5 PROTECTION OF THE TOE SECURITY FUNCTIONS (FPT)

5.1.5.1 Failure with preservation of secure state (FPT_FLS.1)

5.1.5.1.1 FPT_FLS.1.1

The TSF shall preserve a secure state when the following types of failures occur: **power failures and system crashes**.

5.1.6 Strength of Function Requirement

The minimum strength of function level for the security functional requirements is SOF-medium.

5.2 Security Requirements for the IT Environment

The IT Environment shall support the following security requirements.

5.2.1 User Data Protection (FDP) Requirements

5.2.1.1 Discretionary Access Control Policy (FDP_ACC.1)

5.2.1.1.1 FDP_ACC.1.1

The IT Environment shall enforce the **Discretionary Access Control Policy** on **[subjects - processes**] **acting on the behalf of users, [Named objects – Files]; and all operations among subjects and objects covered by the DAC policy.** ***The DAC policy shall be able to control access to users and groups of users.***

5.2.2 IDENTIFICATION AND AUTHENTICATION (FIA)

5.2.2.1 Timing of authentication (FIA_UAU.1)

5.2.2.1.1 FIA_UAU.1.1

The IT Environment shall allow **use of the help function** on behalf of the user to be performed before the user is authenticated.

5.2.2.1.2 FIA_UAU.1.2

The IT Environment shall require each user to be successfully authenticated before allowing any other IT Environment -mediated actions on behalf of that user.

5.2.2.2 Timing of identification (FIA_UID.1)

5.2.2.2.1 FIA_UID.1.1

The IT Environment shall allow **use of the help function** on behalf of the user to be performed before the user is identified.

5.2.2.2.2 FIA_UID.1.2

The IT Environment shall require each user to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.

5.2.3 PROTECTION OF THE TOE SECURITY FUNCTIONS (FPT)

5.2.3.1 Failure with preservation of secure state (FPT_FLS.1)

5.2.3.1.1 FPT_FLS.1.1

The IT Environment shall preserve a secure state when the following types of failures occur: **power failures and system crashes**

5.2.3.2 Non-bypassability of the TSP (FPT_RVM.1)

5.2.3.2.1 FPT_RVM.1.1

The IT Environment shall ensure that IT Environment Security Policy enforcement functions are invoked and succeed before each function within the IT Environment Scope of Control is allowed to proceed

5.2.3.3 TSF domain separation (FPT_SEP.1)

5.2.3.3.1 FPT_SEP.1.1

The IT Environment shall maintain a security domain for its own execution that protects it from interference and tampering by untrusted subjects.

5.2.3.3.2 FPT_SEP.1.2

The IT Environment shall enforce separation between the security domains of subjects in the IT Environment Scope of Control.

5.2.3.4 Reliable Time Stamp (FPT_STM.1)

5.2.3.4.1 FPT_STM.1.1

The IT Environment shall be able to provide reliable time stamps for its own use.

5.3 TOE Security Assurance Requirements

The security assurance requirements for the TOE are the Evaluation Assurance Level (EAL) 3 components as specified in Part 3 of the Common Criteria. No operations are applied to the assurance components.

Assurance Class	Assurance Components
Configuration Management (ACM)	ACM_CAP.3 Authorization controls
	ACM_SCP.1 TOE CM coverage
Delivery and Operation (ADO)	ADO_DEL.1 Delivery procedures
	ADO_IGS.1 Installation, generation, and start-up procedures
Development (ADV)	ADV_FSP.1 Informal functional specification
	ADV_HLD.2 Security enforcing high-level design
	ADV_RCR.1 Informal correspondence demonstration
Guidance Documents (AGD)	AGD_ADM.1 Administrator guidance
	AGD_USR.1 User guidance
Life cycle support (ALC)	ALC_DVS.1 Identification of security measures
Tests (ATE)	ATE_COV.2 Analysis of Coverage
	ATE_DPT.1 Testing: high-level design
	ATE_FUN.1 Functional testing
	ATE_IND.2 Independent testing - sample
Vulnerability assessment (AVA)	AVA_MSU.1 Examination of guidance
	AVA_SOF.1 Strength of TOE security function evaluation
	AVA_VLA.1 Developer vulnerability analysis

Table 2 EAL 3 Assurance Components

5.3.1 Configuration Management (ACM)

5.3.1.1 Authorization Controls (ACM_CAP.3)

5.3.1.1.1 ACM_CAP.3.1D

The developer shall provide a reference for the TOE.

5.3.1.1.2 ACM_CAP.3.2D

The developer shall use a CM system.

5.3.1.1.3 ACM_CAP.3.3D

The developer shall provide CM documentation.

5.3.1.1.4 ACM_CAP.3.1C

The reference for the TOE shall be unique to each version of the TOE.

5.3.1.1.5 ACM_CAP.3.2C

The TOE shall be labeled with its reference.

5.3.1.1.6 ACM_CAP.3.3C

The CM documentation shall include a configuration list and CM plan.

5.3.1.1.7 ACM_CAP.3.4C

The configuration list shall describe the configuration items that comprise the TOE.

5.3.1.1.8 ACM_CAP.3.5C

The CM documentation shall describe the method used to uniquely identify the configuration items.

5.3.1.1.9 ACM_CAP.3.6C

The CM system shall uniquely identify all configuration items.

5.3.1.1.10 ACM_CAP.3.7C

The CM plan shall describe how the CM system is used.

5.3.1.1.11 ACM_CAP.3.8C

The evidence shall demonstrate that the CM system is operating in accordance with the CM plan.

5.3.1.1.12 ACM_CAP.3.9C

The CM documentation shall provide evidence that all configuration items have been and are being effectively maintained under the CM system.

5.3.1.1.13 ACM_CAP.3.10C

The CM system shall provide measures such that only authorized changes are made to the configuration items.

5.3.1.1.14 ACM_CAP.3.1E

The evaluator shall confirm that the information provided meets all the requirements for content and presentation of evidence

5.3.1.2 TOE CM Coverage (ACM_SCP.1)

5.3.1.2.1 ACM_SCP.1.1D

The developer shall provide CM documentation.

5.3.1.2.2 ACM_SCP.1.1C

The CM documentation shall show that the CM system, as a minimum, tracks the following: the TOE implementation representation, design documentation, test documentation, user documentation, administrator documentation, and CM documentation.

5.3.1.2.3 ACM_SCP.1.2C

The CM documentation shall describe how configuration items are tracked by the CM system.

5.3.1.2.4 ACM_SCP.1.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.3.2 Delivery and Operation (ADO)

5.3.2.1 Delivery Procedures (ADO_DEL.1)

5.3.2.1.1 ADO_DEL.1.1D

The developer shall document procedures for delivery of the TOE or parts of it to the user.

5.3.2.1.2 ADO_DEL.1.2D

The developer shall use the delivery procedures.

5.3.2.1.3 ADO_DEL.1.1C

The delivery documentation shall describe all procedures that are necessary to maintain security when distributing versions of the TOE to a user's site.

5.3.2.1.4 ADO_DEL.1.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence

5.3.2.2 Installation, generation, and start-up procedures (ADO_IGS.1)

5.3.2.2.1 ADO_IGS.1.1D

The developer shall document procedures necessary for the secure installation, generation, and start-up of the TOE.

5.3.2.2.2 ADO_IGS.1.1C

The documentation shall describe the steps necessary for secure installation, generation, and start-up of the TOE.

5.3.2.2.3 ADO_IGS.1.1E

The evaluator shall determine that the installation, generation, and start-up procedures result in a secure configuration.

5.3.3 Development (ADV)

5.3.3.1 Informal Functional Specification (ADV_FSP.1)

5.3.3.1.1 ADV_FSP.1.1D

The developer shall provide a functional specification.

5.3.3.1.2 ADV_FSP.1.1C

The functional specification shall describe the TSF and its external interfaces using an informal style.

5.3.3.1.3 ADV_FSP.1.2C

The functional specification shall be internally consistent.

5.3.3.1.4 ADV_FSP.1.3C

The functional specification shall describe the purpose and method of use of all external TSF interfaces, providing details of effects, exceptions and error messages, as appropriate.

5.3.3.1.5 ADV_FSP.1.4C

The functional specification shall completely represent the TSF.

5.3.3.1.6 ADV_FSP.1.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.3.3.1.7 ADV_FSP.1.2E

The evaluator shall determine that the functional specification is an accurate and complete instantiation of the TOE security functional requirements.

5.3.3.2 Security enforcing high-level design (ADV_HLD.2)

5.3.3.2.1 ADV_HLD.2.1D

The developer shall provide the high-level design of the TSF.

5.3.3.2.2 ADV_HLD.2.1C

The presentation of the high-level design shall be informal.

5.3.3.2.3 ADV_HLD.2.2C

The high-level design shall be internally consistent.

5.3.3.2.4 ADV_HLD.2.3C

The high-level design shall describe the structure of the TSF in terms of subsystems.

5.3.3.2.5 ADV_HLD.2.4C

The high-level design shall describe the security functionality provided by each subsystem of the TSF.

5.3.3.2.6 ADV_HLD.2.5C

The high-level design shall identify any underlying hardware, firmware, and/or software required by the TSF with a presentation of the functions provided by the supporting protection mechanisms implemented in that hardware, firmware, or software.

5.3.3.2.7 ADV_HLD.2.6C

The high-level design shall identify all interfaces to the subsystems of the TSF.

5.3.3.2.8 ADV_HLD.2.7C

The high-level design shall identify which of the interfaces to the subsystems of the TSF are externally visible.

5.3.3.2.9 ADV_HLD.2.8C

The high-level design shall describe the purpose and method of use of all interfaces to the subsystems of the TSF, providing details of effects, exceptions and error messages, as appropriate.

5.3.3.2.10 ADV_HLD.2.9C

The high-level design shall describe the separation of the TOE into TSP-enforcing and other subsystems.

5.3.3.2.11 ADV_HLD.2.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.3.3.2.12 ADV_HLD.2.2E

The evaluator shall determine that the high-level design is an accurate and complete instantiation of the TOE security functional requirements.

5.3.3.3 Informal correspondence demonstration (ADV_RCR.1)

5.3.3.3.1 ADV_RCR.1.1D

The developer shall provide an analysis of correspondence between all adjacent pairs of TSF representations that are provided.

5.3.3.3.2 ADV_RCR.1.1C

For each adjacent pair of provided TSF representations, the analysis shall demonstrate that all relevant security functionality of the more abstract TSF representation is correctly and completely refined in the less abstract TSF representation.

5.3.3.3.3 ADV_RCR.1.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.3.4 Guidance Documents (AGD)

5.3.4.1 Administrator Guidance (AGD_ADM.1)

5.3.4.1.1 AGD_ADM.1.1D

The developer shall provide administrator guidance addressed to system administrative personnel.

5.3.4.1.2 AGD_ADM.1.1C

The administrator guidance shall describe the administrative functions and interfaces available to the administrator of the TOE.

5.3.4.1.3 AGD_ADM.1.2C

The administrator guidance shall describe how to administer the TOE in a secure manner.

5.3.4.1.4 AGD_ADM.1.3C

The administrator guidance shall contain warnings about functions and privileges that should be controlled in a secure processing environment.

5.3.4.1.5 AGD_ADM.1.4C

The administrator guidance shall describe all assumptions regarding user behavior that are relevant to secure operation of the TOE.

5.3.4.1.6 AGD_ADM.1.5C

The administrator guidance shall describe all security parameters under the control of the administrator, indicating secure values as appropriate.

5.3.4.1.7 AGD_ADM.1.6C

The administrator guidance shall describe each type of security-relevant event relative to the administrative functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.

5.3.4.1.8 AGD_ADM.1.7C

The administrator guidance shall be consistent with all other documents supplied for evaluation.

5.3.4.1.9 AGD_ADM.1.8C

The administrator guidance shall describe all security requirements on the IT environment that are relevant to the administrator.

5.3.4.1.10 AGD_ADM.1.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence

5.3.4.2 User Guidance (AGD_USR.1)

5.3.4.2.1 AGD_USR.1.1D

The developer shall provide user guidance.

5.3.4.2.2 AGD_USR.1.1C

The user guidance shall describe the functions and interfaces available to the non-administrative users of the TOE.

5.3.4.2.3 AGD_USR.1.2C

The user guidance shall describe the use of user-accessible security functions provided by the TOE.

5.3.4.2.4 AGD_USR.1.3C

The user guidance shall contain warnings about user-accessible functions and privileges that should be controlled in a secure processing environment.

5.3.4.2.5 AGD_USR.1.4C

The user guidance shall clearly present all user responsibilities necessary for secure operation of the TOE, including those related to assumptions regarding user behavior found in the statement of TOE security environment.

5.3.4.2.6 AGD_USR.1.5C

The user guidance shall be consistent with all other documentation supplied for evaluation.

5.3.4.2.7 AGD_USR.1.6C

The user guidance shall describe all security requirements on the IT environment that are relevant to the user.

5.3.4.2.8 AGD_USR.1.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.3.5 Life Cycle Support (ALC)

5.3.5.1 Identification of security measures (ALC_DVS.1)

5.3.5.1.1 ALC_DVS.1.1D

The developer shall produce development security documentation.

5.3.5.1.2 ALC_DVS.1.1C

The development security documentation shall describe all the physical, procedural, personnel, and other security measures that are necessary to protect the confidentiality and integrity of the TOE design and implementation in its development environment.

5.3.5.1.3 ALC_DVS.1.2C

The development security documentation shall provide evidence that these security measures are followed during the development and maintenance of the TOE.

5.3.5.1.4 ALC_DVS.1.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.3.5.1.5 ALC_DVS.1.2E

The evaluator shall confirm that the security measures are being applied.

5.3.6 Security Testing (ATE)

5.3.6.1 Analysis of coverage (ATE_COV.2)

5.3.6.1.1 ATE_COV.2.1D

The developer shall provide an analysis of the test coverage.

5.3.6.1.2 ATE_COV.2.1C

The analysis of the test coverage shall demonstrate the correspondence between the tests identified in the test documentation and the TSF as described in the functional specification.

5.3.6.1.3 ATE_COV.2.2C

The analysis of the test coverage shall demonstrate that the correspondence between the TSF as described in the functional specification and the tests identified in the test documentation is complete.

5.3.6.1.4 ATE_COV.2.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.3.6.2 Testing: high-level design (ATE_DPT.1)

5.3.6.2.1 ATE_DPT.1.1D

The developer shall provide the analysis of the depth of testing.

5.3.6.2.2 ATE_DPT.1.1C

The depth analysis shall demonstrate that the tests identified in the test documentation are sufficient to demonstrate that the TSF operates in accordance with its high-level design.

5.3.6.2.3 ATE_DPT.1.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.3.6.3 Functional testing (ATE_FUN.1)

5.3.6.3.1 ATE_FUN.1.1D

The developer shall test the TSF and document the results.

5.3.6.3.2 ATE_FUN.1.2D

The developer shall provide test documentation.

5.3.6.3.3 ATE_FUN.1.1C

The test documentation shall consist of test plans, test procedure descriptions, expected test results and actual test results.

5.3.6.3.4 ATE_FUN.1.2C

The test plans shall identify the security functions to be tested and describe the goal of the tests to be performed.

5.3.6.3.5 ATE_FUN.1.3C

The test procedure descriptions shall identify the tests to be performed and describe the scenarios for testing each security function. These scenarios shall include any ordering dependencies on the results of other tests.

5.3.6.3.6 ATE_FUN.1.4C

The expected test results shall show the anticipated outputs from a successful execution of the tests.

5.3.6.3.7 ATE_FUN.1.5C

The test results from the developer execution of the tests shall demonstrate that each tested security function behaved as specified.

5.3.6.3.8 ATE_FUN.1.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.3.6.4 Independent testing – sample (ATE_IND.2)

5.3.6.4.1 ATE_IND.2.1D

The developer shall provide the TOE for testing.

5.3.6.4.2 ATE_IND.2.1C

The TOE shall be suitable for testing.

5.3.6.4.3 ATE_IND.2.2C

The developer shall provide an equivalent set of resources to those that were used in the developer's functional testing of the TSF.

5.3.6.4.4 ATE_IND.2.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.3.6.4.5 ATE_IND.2.2E

The evaluator shall test a subset of the TSF as appropriate to confirm that the TOE operates as specified.

5.3.6.4.6 ATE_IND.2.3E

The evaluator shall execute a sample of tests in the test documentation to verify the developer test results.

5.3.7 Vulnerability Assessment (VLA)

5.3.7.1 Validation of analysis (AVA_MSU.1)

5.3.7.1.1 AVA_MSU.1.1D

The developer shall provide guidance documentation.

5.3.7.1.2 AVA_MSU.2.2D

The developer shall document an analysis of the guidance documentation.

5.3.7.1.3 AVA_MSU.1.1C

The guidance documentation shall identify all possible modes of operation of the TOE (including operation following failure or operational error), their consequences and implications for maintaining secure operation.

5.3.7.1.4 AVA_MSU.1.2C

The guidance documentation shall be complete, clear, consistent and reasonable.

5.3.7.1.5 AVA_MSU.1.3C

The guidance documentation shall list all assumptions about the intended environment.

5.3.7.1.6 AVA_MSU.1.4C

The guidance documentation shall list all requirements for external security measures (including external procedural, physical and personnel controls).

5.3.7.1.7 AVA_MSU.1.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.3.7.1.8 AVA_MSU.1.2E

The evaluator shall repeat all configuration and installation procedures to confirm that the TOE can be configured and used securely using only the supplied guidance documentation.

5.3.7.1.9 AVA_MSU.1.3E

The evaluator shall determine that the use of the guidance documentation allows all insecure states to be detected.

5.3.7.2 Strength of TOE security function evaluation (AVA_SOF.1)

5.3.7.2.1 AVA_SOF.1.1D

The developer shall perform a strength of TOE security function analysis for each mechanism identified in the ST as having a strength of TOE security function claim.

5.3.7.2.2 AVA_SOF.1.1C

For each mechanism with a strength of TOE security function claim the strength of TOE security function analysis shall show that it meets or exceeds the minimum strength level defined in the PP/ST.

5.3.7.2.3 AVA_SOF.1.2C

For each mechanism with a specific strength of TOE security function claim the strength of TOE security function analysis shall show that it meets or exceeds the specific strength of function metric defined in the PP/ST.

5.3.7.2.4 AVA_SOF.1.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.3.7.2.5 AVA_SOF.1.2E

The evaluator shall confirm that the strength claims are correct.

5.3.7.3 Independent vulnerability analysis (AVA_VLA.1)

5.3.7.3.1 AVA_VLA.1.1D

The developer shall perform and document an analysis of the TOE deliverables searching for obvious ways in which a user can violate the TSP.

5.3.7.3.2 AVA_VLA.1.2D

The developer shall document the disposition of obvious vulnerabilities.

5.3.7.3.3 AVA_VLA.1.1C

The documentation shall show, for all identified vulnerabilities, that the vulnerability cannot be exploited in the intended environment for the TOE.

5.3.7.3.4 AVA_VLA.1.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.3.7.3.5 AVA_VLA.1.2E

The evaluator shall conduct penetration testing, building on the developer vulnerability analysis, to ensure obvious vulnerabilities have been addressed.

6. TOE Summary Specification

This chapter describes the SurfinGate security functions and associated assurance measures.

6.1 TOE Security Functions

6.1.1 Audit

6.1.1.1 Audit Generation

SurfinGate creates audit records to record mobile code policy violations. When the analysis engine discovers a policy violation, an audit record is created and written into the audit log. The audit record contains the name of the requesting client, the type of mobile code, date, type of event, and success or failure.

6.1.1.2 Audit Management

The SurfinConsole is used to manage the audit log. In order to use the SurfinConsole, a user must be an administrator and must log in using a password. The SurfinConsole provides a graphical interface to manage, view, and analyze the audit logs.

6.1.2 Authentication

SurfinGate provides authentication mechanism to authenticate users to the TOE. Users must authenticate themselves to the SurfinGate Console locally. The login process prompts a user for a password. The user enters a password that is represented as "*" characters during authentication. That information is compared against a locally stored password. If the password matches the entry, the user is permitted access; otherwise, the user is denied access. All users that successfully authenticate to the SurfinConsole are administrators of the system. There is only one valid account on the system – administrator.

The administrator guide recommends a password greater than eight characters (with 94 possible characters), which exceeds 6,000,000,000,000,000 password combinations. This password space exceeds the requirement of 250,000,000,000,000 possible password combinations.

6.1.3 Information Flow Protection

All web-based traffic is scanned for violations of the mobile code flow security policy. The mobile code policy can address the following types of mobile code:

- Java,
- JavaScript,
- Active X,
- VBScript,
- File Extensions, and
- URLs.

The mobile code flow security policy can permit or deny web-based traffic based on any of the characteristics identified above. The administrator selects which types of mobile code to filter and to what degree to perform filtering. Filters can be established in a general or specific manner. For example, the web-based traffic can be filtered based on a specific client or code type, or it can generally block a particular type of web-based traffic. SurfinGate has the ability to block a specific script on a page while permitting the remainder of the page to be displayed.

When web-based traffic is received, SurfinGate collects it and sends it to its database for analysis. If a particular type of web-based traffic has been marked disallow in the security policy, the analysis engine will generate a violation and react as configured by the administrator. If a malicious mobile code element is discovered, the appropriate code scanner statically scans it. This scan function enables the analysis engine to determine if the code element is malicious.

When the SurfinGate Server discovers a violation, it can react in several ways depending upon the administrator defined security policy. The types of reactions are:

- Block – The SurfinGate Server can block the download of malicious mobile code, block web sites, and block data transfers;
- Alert – Send a visual alert to the user and/or email alert to the administrator; and
- Log – Simply log the violation and continue processing. In all cases a log record will be generated; however, in some instances, data transfer may or may not continue.

6.1.4 Management

SurfinGate supports a number of policies and features that require appropriate management. With no exceptions, the security management functions are restricted to an administrator. The TOE supports security management functions for the following security policies and features:

Account Policy – The password management functions allow an administrator to change the password for the administrator login.

Mobile Code Flow Policy– The mobile code flow policy management functions allow an administrator to configure the filter options on the SurfinGate Server. These options include the types of malicious mobile code to scan for, URLs to block, IP addresses to monitor, users to monitor, and executables to block. The security policy determines which events take priority and what actions to take if the web-based traffic violated the security policy.

6.1.5 TOE Protection Mechanisms

SurfinGate will not function if it loses power or suffers a crash. Under those circumstances, when SurfinGate is restarted, all policy settings remain the same as they were before the crash.

Secure state is defined as the following. Initially, the secure state of the machine is to disallow all types of web-based mobile code. The administrator chooses which types of mobile code to permit. After a system failure, the security policy is restored to the same values it had before the system failure.

6.2 TOE Security Assurance Measures

The following assurance measures are applied to satisfy the Common Criteria EAL 3 assurance requirements:

- Configuration Management;
- Delivery and Operation;
- Development;
- Guidance Documents;
- Life Cycle Support;
- Security Testing; and,
- Vulnerability Assessment.

6.2.1 Configuration Management

The Configuration Management (CM) system applied by Finjan ensures each product release is assigned a unique identifier. The CM system also identifies each hardware and software item that composes the TOE. The documentation and other programs managed by the CM system include: design documentation, test documentation, tests, user guide, administrator guide, and the configuration management plan. The CM plan is documented within the following documents:

- Finjan Configuration Management Plan
- Finjan QA Department Procedures
- Finjan Product Release Procedures
- SurfinGate NT 5.5 Build procedure
- SurfinGate 5.6 Unix Solaris Build Procedure

Assurance Requirements Satisfied: ACM_CAP.3 and ACM_SCP.1

6.2.2 Delivery and Operation

Finjan has a set of Delivery and Operation documentation that describes the procedures for the delivery of the TOE. The documentation describes what is delivered with the TOE, instructions for installing and configuring the TOE, and warnings for the administrator to follow during installation. The Delivery and Operation documents are:

- Finjan Operation and Delivery Procedures
- SurfinGate Installation Manual

Assurance Requirements Satisfied: ADO_DEL.1 and ADO_IGS.1

6.2.3 Development

Finjan has a functional specification that describes the external interfaces of the TOE including the effects, exceptions, and error messages. There are also design documents that describe the security functions of the subsystems of the TOE. A correspondence exists that maps the high level design to the functional specification and the functional specification to the ST. The documents that meet the development assurance requirement:

- SurfinGate Interfaces
- High Level Description FHTTP
- Fdata Remote – High Level Design
- SurfinGate White Paper
- SurfinGate Process Management HLD
- Alerts on Scripts Blocking HLD
- Log, Audit Messages, and Alerts Generation SurfinGate
- Error Messages Guide SurfinGate 5.6
- SurfinConsole Administrator HLD
- SurfinGate Correspondence Documentation

Assurance Requirements Satisfied: ADV_FSP.1, ADV_HLD.2, and ADV_RCR.

6.2.4 Guidance Documents

The Guidance Documents provided by Finjan include both administrator and user manuals. The administrator manual describes the administrative functions and interfaces, provides guidance on how to administer the TOE securely, and contains warnings about functions and privileges that should be controlled. The user manual describes the functions and interfaces available to non-administrative users,

describes the user-accessible security functions, and contains warnings to users about functions that should be controlled. The guidance documents are:

- SurfinGate User Guide
- SurfinGate Release Notes

Assurance Requirements Satisfied: AGD_ADM.1 and AGD_USR.1

6.2.5 Life Cycle Support

The Life Cycle Support documentation describes how all the physical, procedural, personnel, and other security measures that are necessary to protect the confidentiality and integrity of the TOE design and implementation in it development environment are followed. The life cycle support document is

- Finjan Security Procedures
- Finjan M.I.S. - System Administration, Guidelines & procedures
- Product Development Security procedures

Assurance Requirements Satisfied: ALC_DVS.1

6.2.6 Security Testing

Finjan maintains a security test suite consisting of test plans, procedures, expected results, and actual results. Finjan has performed and documented an analysis that the test suite adequately tests the interfaces from both a coverage and depth perspective. A correspondence exists that maps the test suite to the functional specification. The test documents are:

- SurfinGate: QA ATP Report
- SurfinGate: QA ATP Security Report
- Acceptance Test Plan, SurfinGate release 5.6
- Acceptance Test Plan, Test Data Description

Assurance Requirements Satisfied: ATE_COV.2, ATE_DPT.1, ATE_FUN.1, and ATE_IND.2

6.2.7 Vulnerability Assessment

Finjan has provided guidance documents that identify all possible modes of operation of the TOE, their consequences, and implications for maintaining secure operation. The guidance documents list all assumptions about intended usage and the TOE's environment. The guidance document t is:

- SurfinGate User Guide

The Strength of Function analysis performed on the password mechanism is provided in the following Finjan document:

- Finjan Strength of Function Analysis

As part of its design and testing process, Finjan performs a vulnerability assessment. This analysis includes a search for obvious ways in which a user can violate the TSP. For all identified vulnerabilities, Finjan provides an explanation why the vulnerability cannot be exploited in the intended environment for the TOE. The following documents the vulnerability analysis:

- MCRC Procedures
- Vulnerability Assessments, SurfinGate and SurfinShield

Assurance Requirements Satisfied: AVA_MSU.1, AVA_SOF.1, and AVA_VLA.1

7. Protection Profile Claims

None.

8. Rationale

This section provides the rationale for completeness and consistency of the Security Target. The rationale addresses the following areas:

- Security Objectives;
- Security Functional Requirements;
- Security Functional Requirement Dependencies;
- Security Assurance Requirements;
- Explicitly Stated Requirements; and
- TOE Summary Specification.

8.1 Security Objectives Rationale

This section shows that all assumptions, threats, and organizational security policies are completely covered by TOE and environmental security objectives. Each TOE IT objective counters or addresses at least one organizational security policy, or threat. In addition, each Non-IT or environmental security objectives, addresses an assumption. The relationship of the TOE assumptions, threats and policies to IT security objectives is provided in Table 3 Security Objectives Rationale.

	O.MEDIAT	O.DETECT	O.REACT	O.ACCESS	O.EADMIN	O.IDAUTH	O.FAIL	O.CREDEN	O.INSTAL	O.PERSON	O.PHYCAL	O.OSAUTH	O.ACCESS	O.TIME	O.SEP	O.OSFAIL
T.UNAUTH	X															
T.FALASC		X														
T.FALACT			X													
T.IMPCON				X												
T.LOSSOF				X												
T.NOFAIL							X									X
P.ACCESS				X												
P.IDAUTH						X										
P.MANAGE					X											
A.MANAGE								X	X	X						
A.NOEVIL									X	X						
A.LOCATE											X					
A.PROTCT											X					
A.FWALL	X															
A.DMZ									X							
A.APPS															X	
A.MAIL				X												
A.ADMINS												X				
A.I&A												X				
A.SEP															X	
A.FILES												X	X			

	O.MEDIAT	O.DETECT	O.REACT	O.ACCESS	O.EADMIN	O.IDAUTH	O.FAIL	O.CREDEN	O.INSTAL	O.PERSON	O.PHYCAL	O.OSAUTH	O.ACCESS	O.TIME	O.SEP	O.OSFAIL
A.TIME														X		

Table 3 Security Objectives Rationale

T.UNAUTH Malicious mobile code may enter the IT System monitored by the TOE undetected.

To counter this threat, all web-based traffic needs to be monitored by the TOE. The O.MEDIAT objective ensures that all web-based traffic is monitored.

T.FALASC The TOE may fail to identify malicious mobile code based on data received.

To counter this threat, the TOE needs to be able to analyze the data it receives and draw conclusions based on the data. The O.DETECT objective counters this threat by requiring the TOE to detect malicious mobile code.

T.FALACT The TOE may fail to react to identified or suspected malicious mobile code.

To counter this threat, the TOE needs to react to detected malicious mobile code. The O.REACT objectives addresses this threat by ensuring the TOE provides a reaction mechanism.

T.IMPCON An unauthorized user may inappropriately change the configuration of the TOE.

To counter this threat, the TOE must limit the access to configuration settings of the TOE. The O.ACCESS objective ensures that only authorized users can access TOE functions and data.

T.LOSSOF An unauthorized user may attempt to remove or destroy data collected and produced by the TOE.

To counter this threat, the TOE must limit access to the data it collects and produces. The O.ACCESS objective ensures that only authorized users can access TOE functions and data.

T.NOFAIL After a system failure (i.e., power failure, hardware failure, and operating system failure), the TOE is restored to a permissive state and allows web-based traffic that would have been denied otherwise.

To counter this threat, the TOE needs to provide for a secure recovery. The O.FAIL objective ensures the TOE can recover securely from failures and the O.OSFAIL objective ensures the host operating system and database support the secure recovery.

P.ACCESS All data collected and produced by the TOE shall only be used for authorized purposes.

This policy supports limiting the use of TOE data. The O.ACCESS objective ensures that only authorized users may access TOE data and that those authorized users may only do so in an appropriate manner.

P.IDAUTH Administrators must authenticate before accessing any TOE functions or data.

This policy requires administrators to authenticate themselves to the TOE before performing any actions. The O.IDAUTH objective addresses this policy by requiring administrators to authenticate themselves to the TOE prior to performing any functions.

P.MANAGE The TOE shall provide a set of administrative tools to manage the TOE's functions and data.

This policy ensures the administrator will have adequate functions to manage the TOE. The O.EADMIN objective ensures the TOE has a set of functions to effectively manage its functions and data.

A.MANAGE **There will be one or more competent individuals assigned to manage the TOE and the security of the information it contains.**

This assumption is addressing the quality of the personnel that administer the TOE. The O.CREDEN objective contributes to this objective by ensuring that administrators will protect their authentication data so unauthorized users cannot gain access. The O.PERSON objective ensures that qualified persons are selected and trained to manage the TOE. Lastly, the O.INSTAL objective ensures the administrators install and manage the TOE according to the administrator guidance. Together these objectives ensure that the persons managing the TOE are qualified.

A.NOEVIL **The administrators are not careless, wilfully negligent, or hostile, and will follow and abide by the instructions provided by the TOE documentation.**

This assumption addresses hostile administrators. The O.PERSON objective ensures the administrators are trained to operate the TOE. The O.INSTAL objective ensures those properly selected administrators operate in a manner consistent with security. Together these objectives fulfil the assumption.

A.LOCATE **The processing resources of the TOE will be located within controlled access facilities, which will prevent unauthorized physical access.**

The O.PHYCAL objective ensures the TOE is protected from physical attack. This objective addresses the assumption that the TOE is located within controlled access facilities.

A.PROTCT **The TOE hardware and software critical to security policy enforcement will be protected from unauthorized physical modification.**

The O.PHYCAL objective ensures the TOE is protected from physical attack. This objective addresses the assumption that the TOE is protected from unauthorized physical modification.

A.FWALL **All network web-based traffic will be directed through the SurfinGate.**

This assumption ensures all web-based traffic is monitored. The O.MEDIAT objective ensures all web-based traffic received by the TOE is monitored.

A.DMZ **SurfinGate will reside in a Demilitarized Zone (DMZ) to protect it from direct attacks.**

This assumption ensures the TOE is protected from direct port attacks. The O. INSTAL objective ensures the TOE is installed properly.

A.APPS **SurfinGate will be the only application running on its host server.**

This assumption ensures other products do not interfere with the TOE. The O.SEP objective ensures the TOE is protected from interference from other applications.

A.MAIL **The mail server on the SurfinGate network will accept only outgoing mail from the SurfinGate product and will deliver mail properly.**

This assumption ensures that alert messages are sent to the administrator when generated. The O.REACT object ensures proper reaction mechanisms.

A.ADMINS **The host operating system for SurfinGate will only permit access by trusted users.**

This assumption ensures untrusted users do not have direct access to SurfinGate. The O.OSAUTH objective ensures that the underlying operating system performs identification and authentication of users.

A.I&A **Users of the underlying system are identified and authenticated.**

The O.OSAUTH objective ensures that the underlying operating system performs identification and authentication of users as stated by the assumption.

A.SEP **The underlying system will provide mechanisms to isolate the TOE Security Functions (TSF) and assure that TSF components cannot tampered with or bypassed. The TSF components are 1) the files used by SurfinGate to store its data and 2) the TOE processes managing the mobile code scanning.**

The O.SEP objective provides for mechanisms to isolate the TSF and prevent tampering with the TSF. This objective corresponds to the assumption that the underlying system provides isolation functions.

A.FILES **All of the SurfinGate related files and directories (including executables, run-time libraries, audit logs) are protected from unauthorized access by the system control mechanisms.**

This assumption addresses protection of the TOE system information. The O.ACCESS objective ensures the underlying system provides protection for the TOE data and system feels.

A.TIME **The underlying system will provide a reliable timestamp.**

This assumption ensures there is an accurate timestamp available for system use in the audit trail. The O.TIME objective addresses this assumption by requiring the underlying system provide a timestamp.

8.2 Security Requirements Rationale

This section provides evidence supporting the internal consistency and completeness of the components (requirements) in the Security Target. The relationship of the IT objectives to the security functional requirements is provided in Table 4 Requirements vs. Objectives Mapping. The rationale supporting this mapping is provided in the remainder of this section.

	FAU_ARP.1	FAU_GEN.1	FAU_SAA.1	FAU_SAR.1	FAU_SAR.2	FAU_SAR.3	FAU_STG.1	FDP_IFC.2	FDP_IFF.1	FIA_UAU.2	FIA_UAU.7	FIA_SOS.1	FMT_MOF.1	FMT_MSA.3	FMT_MTD.1	FPT_FLS.1
O.MEDIAT								X	X							
O.DETECT		X	X													
O.REACT	X															
O.ACCESS					X		X						X		X	
O.EADMIN				X		X								X		
O.IDAUTH										X	X	X				
O.FAIL																X

Table 4 Requirements vs. Objectives Mapping

O.MEDIAT **The TOE will scan all data received.**

The FDP_IFC.2 and FDP_IFF.1 requirements address this objective. They ensure the mobile code policy is applied to all web-based traffic monitored by SurfinGate.

O.DETECT **The TOE will detect malicious mobile code.**

The FAU_SAA.1 requirement ensures there is an administrator defined mobile code policy in place. The FAU_GEN.1 requirement applies the mobile code policy to the web-based traffic it receives. These two requirements together address this objective.

O.REACT **The TOE will react to any malicious mobile code in accordance with the TSP to alert the administrator of a potential security violation.**

The FAU_ARP.1 requirement provides for the creation of alerts and audit records when malicious mobile code is detected. The type of reaction is administrator-defined.

O.ACCESS **The TOE must allow authorized users to access only appropriate TOE functions and data.**

FAU_SAR.2 limits read access to the audit data to the administrator. FAU_STG.1 ensure the audit data is protected from modifications and deletions by anyone. The FMT_MOF.1 requirement limits the management of the mobile code access policy to the administrator. The FMT_MTD.1 requirement limits the administrator's actions with respect to the audit data and other system data. These requirements together address this objective.

O.EADMIN **The TOE must include a set of functions that allow effective management of its functions and data.**

FAU_SAR.1 provides tools for the administrator to analyze the audit trail. Additionally, FAU_SAR.3 ensures the administrator has the capability to search and sort the audit data. The FMT_MSA.3 requirement allows the administrator to override default values in the mobile code access policy. Together these requirements address this objective.

O.IDAUTH **The TOE must be able to authenticate administrators prior to allowing access to TOE functions and data.**

FIA_UAU.2 requires the administrator supply a password prior to accessing SurfinGate. FIA_UAU.7 supports the authentication objective by obscuring the password during the authentication process. FIA_SOS.1 ensures the password is not easily guessable. Together these requirements address this objective.

O.FAIL **The TOE must provide a means for secure recovery from failures.**

The FPT_FLS.1 requirement ensures SurfinGate preserves a secure state after a power failure or system crash. This requirement addresses the stated objective.

8.3 Environmental Security Functional Requirements Rationale

This section provides evidence supporting the internal consistency and completeness of the security functional requirements for the environment in the Security Target. The relationship of the environmental IT objectives to the environmental security functional requirements is provided in Table 5 Environmental Requirements vs. Objectives. The rationale supporting this mapping is provided in the remainder of this section.

	FDP_ACC.1	FIA_UAU.1	FIA_UID.1	FPT_FLS.1	FPT_RVM.1	FPT_SEP.1	FPT_STM.1
O.OSAUTH		X	X				
O.ACCESS	X						
O.TIME							X

	FDP_ACC.1	FIA_UAU.1	FIA_UID.1	FPT_FLS.1	FPT_RVM.1	FPT_SEP.1	FPT_STM.1
O.SEP					X	X	
O.OSFAL				X			

Table 5 Environmental Requirements vs. Objectives

O.OSAUTH **The system shall identify and authenticate users prior to providing access to any TOE facilities.**

The FIA_UID.1 and FIA_UAU.1 requirements ensure that the system must perform identification and authentication of all users. These requirements work together to satisfy this objective.

O.ACCESS **The system shall provide the access control mechanisms required to protect the TOE's data and system files.**

The FDP_ACC.1 requirement ensures there will be a discretionary access control (DAC) mechanism present to protect files. The details of the requirement are not necessary (i.e., FDP_ACF.1) to meet the stated objective. Rather, SurfinGate is only concerned that DAC is available and is not concerned with its implementation details such as permission bits or access control lists.

O.TIME **The system shall provide a reliable timestamp the TOE can use for accurately tracking audit events.**

FPT_STM.1 requires the system to provide reliable timestamps. These timestamps will ensure that the TOE has an accurate timestamp to use in the audit trail.

O.SEP **The system shall provide mechanisms to isolate the TOE Security Functions (TSF) and assure that TSF components cannot tampered with or bypassed.**

The FPT_RVM.1 and FPT_SEP.1 requirements provide assurance that the TOE is non-bypassable and is isolated from untrusted subjects. These two requirements ensure the objective is adequately addressed.

O.OSFAIL **The host operating system and database must provide a means for secure recovery from failures.**

The FPT_FLS.1 requirement ensures the IT Environment components can return to a secure state upon a system crash or power failure. This requirement directly satisfies this objective.

8.4 Security Assurance Requirements Rationale

This ST contains the assurance requirements from the CC EAL 3 assurance package and is based on good rigorous commercial development practices. This ST has been developed for a generalized environment with a low level of risk to the assets. The Security Objectives for the TOE were reviewed and EAL 3 was found to sufficient to address them.

8.5 Requirement Dependency Rationale

The ST satisfies all the requirement dependencies of the Common Criteria. Table 6 Requirement Dependencies lists each requirement from Section 5.1with a dependency and indicates whether the dependent requirement was included. For each dependency not included, an explanation is proved for its absence.

Functional Component	Dependency	Included
FAU_ARP.1	FAU_SAA.1	YES
FAU_GEN.1	FMT_STM.1	NO
FAU_SAA.1	FAU_GEN.1	YES
FAU_SAR.1	FAU_GEN.1	YES
FAU_SAR.2	FAU_SAR.1	YES
FAU_SAR.3	FAU_SAR.1	YES
FAU_STG.1	FAU_GEN.1	YES
FDP_IFC.2	FDP_IFF.1	YES
FDP_IFF.1	FDP_IFC.1	NO
	FMT_MSA.3	YES
FIA_UAU.2	FIA_UID.1	NO
FIA_UAU.7	FIA_UID.1	NO
FMT_MOF.1	FMT_SMR.1	NO
FMT_MSA.3	FMT_MSA.1	NO
	FMT_SMR.1	NO
FMT_MTD.1	FMT_SMR.1	NO
FPT_FLS.1	ADV_SPM.1	NO

Table 6 Requirement Dependencies

- FMT_STM.1 - SurfinGate does not provide timestamps of its own. It relies on the supporting operating system to provide it accurate timestamps. The assumption A.TIME addresses timestamps.
- FDP_IFC.1 – The FDP_IFC.2 requirement, which is hierarchical to FDP_IFC.1, has been included. The inclusion of FDP_IFC.2 which provides complete information flow control satisfies the subset information flow control requirement.
- FIA_UID.1 – SurfinGate has only one valid account – administrator. The administrator is required to supply a password to access the account so authentication-related requirements are applicable while identification requirements are implicit.
- FMT_SMR.1 – SurfinGate only has one valid user account and role – administrator. This requirement is unnecessary since all valid logins are the administrator.
- FMT_SMR.1 – This requirement assumes the mobile code access policy is used to protect the TOE when in fact it is not. The mobile code policy is used to protect the web-based traffic. The requirements is not included because it is not applicable to the protection of the TOE.
- ADV_SPM.1 – An informal security policy model is required to define the secure state of the TOE. Section 6.1.5 provides the discussion of secure state so an informal policy model in not needed.

8.6 Explicitly Stated Requirements Rationale

The ST does not contain any explicitly stated requirements. Therefore, this section is not applicable.

8.7 TOE Summary Specification Rationale

This Section in conjunction with Section 6, the TOE Summary Specification, provides evidence that the security functions are suitable to meet the TOE security requirements. The collection of security functions work together to provide all of the security requirements. The security functions described in the TOE summary specification are all necessary for the required security functionality in the TSF. Table 7 Security

Functions vs. Requirements Mapping demonstrates the relationship between security requirements and security functions.

The only security mechanism that is realized by a probabilistic or permutational implementation is the password mechanism. By requiring passwords consist of eight characters (with at least 94 available characters, the password space exceeds 6,000,000,000,000,000,000 available combinations), the claim exceeds the minimum claim provided in the FIA_SOS.1 requirement.

	AUDIT	AUTHENTICATION	INFO FLOW	MANAGEMENT	SELF PROTECT
FAU_ARP.1	X				
FAU_GEN.1	X				
FAU_SAA.1	X				
FAU_SAR.1	X				
FAU_SAR.2	X				
FAU_SAR.3	X				
FAU_STG.1		X			X
FDP_IFC.2			X		
FDP_IFF.1			X		
FIA_UAU.2		X			
FIA_UAU.7		X			
FMT_MOF.1				X	
FMT_MSA.3			X		
FMT_MTD.1	X		X	X	
FPT_FLS.1					X

Table 7 Security Functions vs. Requirements Mapping

FAU_ARP.1 Security Alarms

The SurfinGate provides an alarm mechanism to alert the administrator of any policy violations detected. This function supports the ability for the administrator to configure the alarm type. The alarm can be visual or e-mail. The audit function addresses this requirement.

FAU_GEN.1 Audit Data Generation

SurfinGate generates audit records for all malicious mobile code it detects. For each audit record, SurfinGate includes the client request, type of mobile code, date, type of event, and success or failure. The audit function addresses this requirement.

FAU_SAA.1 Potential Violation Analysis

SurfinGate has a set of rules by which it processes audit data to determine if malicious mobile code has been detected. The administrator defines these rules and all violation conclusions are based upon this set of rules. The audit function addresses this requirement.

FAU_SAR.1 Audit Review

SurfinGate provides a graphical tool for the administrator to read all the information in the audit log. The tool displays the audit log so that the administrator may clearly read and understand all fields in the log. The audit function addresses this requirement.

FAU_SAR.2 Restricted Audit Review

SurfinGate only permits the administrator to log onto the console to review audit data. The administrator is the only user account that may access SurfinGate directly. The audit function addresses this requirement.

FAU_SAR.3 Selectable Audit Review

SurfinGate provides a graphical audit tool for the administrator to manage the audit log. The tool permits the administrator to sort and select audit data based on any field in the audit records. The audit function addresses this requirement.

FAU_STG.1 Protected Audit Trail Storage

SurfinGate stores the audit log in a password-protected database to prevent it from unauthorized access. The only means to access the audit log is via the console that is limited to administrative use only. The self protection and authentication functions address this requirement.

FDP_IFC.2 Complete Information Flow Control

SurfinGate monitors all web-based traffic that flows between clients and external hosts. SurfinGate scans all web-based traffic to determine if malicious mobile code is embedded in the web-based traffic. The information flow function addresses this requirement.

FDP_IFF.1 Simple Security Attributes

SurfinGate supports a security policy that permits the administrator to identify the types of web-based trafficpermitted to flow. The policy can permit or deny web-based traffic based on the following attributes: Mobile code type (e.g., Java, ActiveX), identity of external source, signature verification status, and client identification. The administrator has the option of permitting a client access to a particular type of information by specifying the client identity within the policy rules. The information flow function addresses this requirement.

FIA_UAU.2 User Authentication Before any Action

SurfinGate requires administrators to perform authentication before they may access any of its functions or data. The only valid account is the administrator account. The authentication function addresses this requirement.

FIA_UAU.7 Protected Authentication Feedback

When an administrator attempts to log into SurfinGate, the password is displayed as "*" characters to protect the password. This protects the password from observation by an unauthorized user. The authentication function addresses this requirement.

FIA_SOS.1 Verification of Secrets

The password space used by the TSF reduces the chance of guessing a password to less than 1 in 250,000,000,000,000.

FMT_MOF.1 Management of Security Functions Behavior

The SurfinGate security policy is configured through the SurfinConsole, to which only administrators have access. Through the SurfinConsole, administrators add, remove, and change values within the security policy. The management function addresses this requirement.

FMT_MSA.3 Static Attribute Initialization

The default mobile code security policy is to disallow all web-based traffic. The administrator has the ability to configure the policy to reflect the needs of the organization. The information flow function addresses this requirement.

FMT_MTD.1 Management of TSF Data

The administrator is the only valid account on SurfinGate. The administrator has the ability to read and add to the audit data and to modify the mobile code security policy and all other TOE data. The information flow, audit, and authentication functions address this requirement.

FPT_FLS.1 Failure with Preservation of Secure State

When SurfinGate has a system failure, it returns to the state in which it was before the failure. This preserves the system secure state. The self-protection function addresses this requirement.

APPENDIX A List of Acronyms

AGD	Administrator Guidance Document
CC	Common Criteria
DO	Delivery Operation
EAL	Evaluation Assurance Level
HTTP	Hyper Text Transfer Protocol
IDS	Intrusion Detection System
IP	Internet Protocol
IT	Information Technology
SF	Security Functions
SFR	Security Functional Requirements
ST	Security Target
TOE	Target of Evaluation
TSF	Target of Evaluation Security Functions