



## **hyperguard**

**Plug in your web application security!**

## Inhaltsverzeichnis

<b>1. Schutz von Web-Anwendungen? - Die Gründe.</b>	<b>3</b>
Web-Anwendungen sind lukrative Angriffsziele.	3
Einbrüche bedeuten enormen Image-Verlust.	3
Gesetze und Industrie-Standards sind einzuhalten.	3
<b>2. Herkömmlicher Schutz reicht nicht aus. Warum?</b>	<b>4</b>
Anwendungsschicht vs. Transportschichten	4
<b>3. Typische Schwachstellen von Web-Anwendungen</b>	<b>6</b>
Überprüfung aller Eingaben	6
Session Handling	7
State	7
Angreifbarkeit aus dem Netz	8
Authentizität der Applikation	8
<b>4. Fazit: Web-Anwendungen benötigen speziellen Schutz.</b>	<b>9</b>
<b>5. Web Application Security mit hyperguard</b>	<b>10</b>
Umfassender Schutz auf Web-Anwendungs-Ebene	10
Kernfunktionalitäten von hyperguard	10
Kundennutzen im Überblick	11
<b>6. Wie funktioniert hyperguard? Ein Überblick.</b>	<b>12</b>
Web-Server Plug-In	12
Optimale Kombination von Regel- und Anomalie-basiertem Schutz	12
<b>7. Die Kern-Funktionsbereiche im Überblick</b>	<b>13</b>
Request Verification	13
Secure Session Handling	13
Secure State Management	14
Site Usage Enforcement	14
Serverside Phishing Prevention, Detection and Termination	15
<b>8. Hohe Benutzerfreundlichkeit und niedriger Verwaltungsaufwand</b>	<b>15</b>
<b>9. Zusammenfassung</b>	<b>15</b>
<b>10. Glossar</b>	<b>16</b>

## 1. Schutz von Web-Anwendungen? - Die Gründe.

### Web-Anwendungen sind lukrative Angriffsziele.

Web-Anwendungen bilden das technische Fundament für E-Business. Sie sind die entscheidenden Schnittstellen zwischen Internet und vertraulicher Unternehmensinformation; denn sie "vermitteln" die Kommunikation zwischen dem Web-Browser des Kunden und den Backend-Systemen wie z.B. der SAP-Infrastruktur des Anbieters.

In anderen Worten: Früher mußten Angreifer in das interne Firmennetzwerk eindringen, um an die Unternehmensgeheimnisse in den Backend-Systemen zu gelangen; heute werden viele dieser vertraulichen Daten firmenintern zusammengeführt und für E-Business-Zwecke Web-Anwendungen "anvertraut". Diese haben oft sogar direkten Zugriff auf die Backend-Systeme – und sind durch manipulierte Abfragen angreifbar.

### Einbrüche bedeuten enormen Image-Verlust.

Für E-Business-Geschäftsmodelle ist das Vertrauen der Anwender in die Sicherheit der zugrunde liegenden Systeme unabdingbare Geschäftsgrundlage. Dieses Vertrauen wird derzeit durch eine zunehmende Zahl erfolgreicher Einbrüche, die nur z.T. auch in der Öffentlichkeit bekannt werden, empfindlich gestört.

Beispielsweise titelt heise.de am 23.10.2005 unter der Überschrift „Millionenschäden durch Phishing“ auszugsweise:

*„Nach einer Meldung des Nachrichtenmagazins Focus liegen den deutschen Landeskriminalämtern (LKA) mehr als 1000 Fälle von Kunden vor, deren Bank-Zugangsdaten zu betrügerischen Überweisungen missbraucht wurden. Der geschätzte Schaden summiert sich auf insgesamt 4,5 Millionen Euro.“*

Am 6.10. 2005 hat sich gemäß heise.de „die größte der nordeuropäischen Banken, die Nordea-Bank, einer Phishing-Welle gebeugt und das Online-Banking-Portal [...] kurzfristig vom Netz genommen.“

Die rein finanziellen Verluste erscheinen – gerade für Großunternehmen – noch überschaubar, aber das Vertrauen der Kunden muss oft teuer wiedergewonnen werden.

### Gesetze und Industrie-Standards sind einzuhalten.

In Deutschland sind u.a. das Datenschutzgesetz, das Gesetz zur Kontrolle und Transparenz (KonTraG) und Basel II zu nennen, die insbesondere geeignete Maßnahmen zur Risikovorsorge verlangen und ggfs. sogar mit einer persönlichen Haftung des Managements verbunden sind. Zudem sind Industrie-Standards einzuhalten, beispielsweise MasterCards Payment Card Industry (PCI) Data Security Standard oder VISAs Cardholder Information Security Program (CISP), deren Nicht-Einhaltung mit z.T. sehr hohen Strafen verbunden sein kann.

**Web-Anwendungen = Schnittstellen zwischen Internet und vertraulichen Unternehmensdaten**

**Millionenschäden durch Phishing**

**Abschalten des Online-Portals als „letzte“ Gegenwehr**

**Deutschland:**

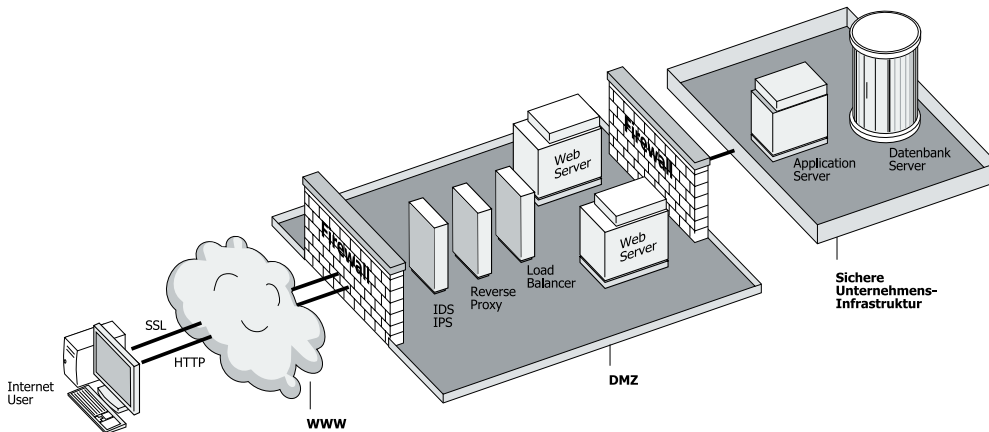
**KontraG, Basel II**

**Internationale Standards**

## 2. Herkömmlicher Schutz reicht nicht aus. Warum?

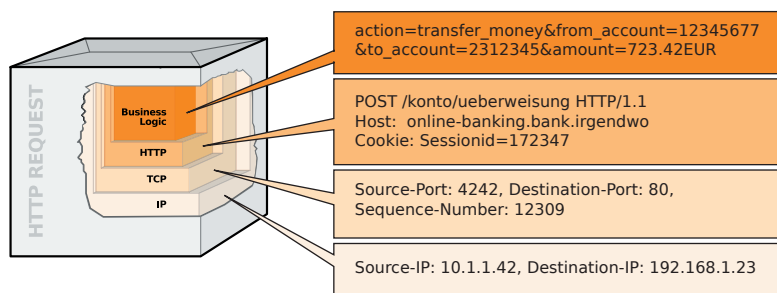
### Anwendungsschicht vs. Transportschichten

Zur Beantwortung dieser Frage hilft ein Blick in die technischen Grundlagen der Kommunikation im Internet, dargestellt anhand einer typischen E-Business-Infrastruktur:



Die Anfragen des Internetnutzers werden auf Seiten des Betreibers durch traditionelle Sicherheitskomponenten wie Firewall, Reverse Proxy und Intrusion Detection Systeme überprüft, bevor sie über die Webserver zur Applikation gelangen.

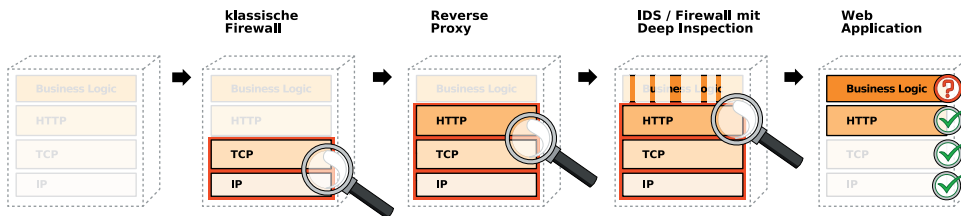
Eine typische E-Banking-Transaktion kann z. B. aus folgendem Request bestehen. In der folgenden Darstellung sind die Daten auf die jeweiligen Protokollschichten im Internet aufgeschlüsselt:



Derzeit typische E-Business-Infrastruktur inkl. üblicher Schutz

Paket-Aufbau eines typischen Requests im Internet

Dieser Request wird auf seinem Weg vom Client zum Backend-System typischerweise wie folgt auf potentielle Angriffe analysiert:



Wie die obige Darstellung deutlich zeigt, wird der Business-Logic-Layer – eventuell mit Ausnahme von einfacher Mustererkennung durch IDS/IPS, Reverse Proxy-Server oder Deep-Inspection-Firewalls – nicht bzw. nicht ausreichend analysiert.

Dies liegt vor allem an folgenden Tatsachen:

- Die genannten Sicherheitssysteme wurden – historisch bedingt – zum Schutz der unteren sog. Transportschicht entwickelt – und hier leisten sie auch sehr gute Arbeit.
- Jede Web-Anwendung benutzt letztlich HTTP nur noch um eine eigene Applikations-spezifische Logik mitzutransportieren. Letztlich muss deshalb jede Anwendung spezifisch geschützt werden.
- Wichtige Daten werden oft SSL-verschlüsselt übertragen und können deshalb von herkömmlichen IT-Sicherheitslösungen nur schwer untersucht werden.

Vereinfachend ausgedrückt:

Die Web-Anwendungen sprechen eine Sprache, die Firewalls, Reverse Proxy Server und IDS/IPS-Systeme nicht kennen.

Hinzu kommt, dass die Vielfalt der angebotenen Web-Script-Sprachen, Application-Frameworks und Webtechnologien eine fast unbegrenzte Anzahl von Sicherheitslücken erzeugt – eine ideale Ausgangsposition für Hacker.

**Kein Schutz des Business-Logic-Layers**

**Jede Web-Anwendung benötigt spezifischen Schutz**

**Web-Anwendungen sprechen eine andere Sprache als traditionelle Sicherheitssysteme**

### 3. Typische Schwachstellen von Web-Anwendungen

Diese resultieren im Kern aus der netzwerkbasierter Natur von Webapplikationen. Im folgenden sollen die größten Problemfelder aufgezeigt werden.

#### Überprüfung aller Eingaben

Die meisten Programmierer und ihre Programme haben ein sehr menschliches Problem: Sie vertrauen dem Benutzer der Applikation.

Das daraus resultierende klassische Problem ist der Buffer Overflow:

Eine Applikation erwartet ein Wort oder eine Zeile als Eingabe des Benutzers und ist nicht darauf vorbereitet, dass der Benutzer mehr als 1024 Zeichen oder gar 2 GigaByte an Daten eingibt. Dann werden von der Eingabe plötzlich andere Teile des Programms überschrieben. Im Bereich der Webapplikationen spielt der einfache Buffer-Overflow keine so grosse Rolle mehr, da sie meistens in Java, PHP, Perl oder anderen Sprachen mit automatischer Speicherverwaltung geschrieben werden.

Dafür treten immer dann Probleme auf, wenn Eingabedaten des Benutzers (oder des Angreifers) ungeprüft an andere Komponenten des Gesamtsystems weitergeleitet werden.

Hieraus resultiert dann die Problemklasse der Command-Injection-Angriffe; am bekanntesten dürfte hier die SQL-Injection sein. Hierbei kann ein Angreifer direkt Kommandos auf der zur Webapplikation gehörenden Datenbank ausführen, fremde Daten auslesen oder manipulieren.

Erschwerend kommt hinzu, dass es keineswegs offensichtlich ist, welche Daten tatsächlich Benutzereingaben sind. Oft wird fälschlicherweise angenommen, dass von der Applikation gesetzte Cookies, versteckte Eingabefelder oder z. B. die Namen von Auswahlboxen in einem Webformular nur unverändert wieder als Eingabe erscheinen können.

Tatsächlich sind aber alle genannten Beispiele von einem Angreifer frei manipulierbar. Selbst der Hostname des Clients kann unter Umständen für Angriffe benutzt werden.

**Traue keiner Eingabe des Benutzers!**

**Typische Angriffe:**

**Injection-Angriffe**

**Parameter Manipulation**

**Cookie Poisoning**

## Session Handling

Prinzipiell ist das HTTP Protokoll zustandslos. Das Web war ja ursprünglich nur als Medium gedacht, um Inhalte zu publizieren. Die Idee, damit komplexere Applikationen wie einen Online Shop zu realisieren, kam erst später auf. Deshalb besteht die Kommunikation - vom Standpunkt der Applikation aus - aus vielen einzelnen unabhängigen HTTP-Anfragen.

Eine Web-Anwendung wie ein Online Shop muß zusammengehörende HTTP Anfragen auch als zusammengehörig (zu einer Session gehörend) erkennen. Dafür müssen Informationen von einer Anfrage an die nächste übergeben werden.

Oftmals sind diese Sessions nicht besonders geschützt, das heisst ein Angreifer kann eine bestehende Session eines anderen Benutzers übernehmen (Session Hijacking). Da an diese Session in der Regel Authentisierungsinformationen gebunden sind, kann der Angreifer dann im Namen des fremden Nutzers agieren, zum Beispiel auf seine Rechnung Einkäufe tätigen.

## State

Fast jede Applikation heutzutage ist stateful (zustandsbehaftet), das heisst, es gibt einen inneren Zustand der Applikation, der im Verlauf der Benutzung aktualisiert wird und von dem das weitere Verhalten der Applikation abhängig ist.

Ein einfaches Beispiel hierfür ist der Warenkorb beim Online Shopping. Während bei klassischen Programmen dieser Zustand implizit durch das Speicherabbild des Programms zur Laufzeit gegeben ist, muss dieser Zustand bei Webapplikationen explizit abgelegt werden. In der Regel werden dafür Cookies oder die Parameter in der URL benutzt. Diese können jedoch vom Benutzer manipuliert werden. Der Applikationsentwickler muss also eine klare Unterscheidung zwischen sicherheitsrelevanten und unkritischen Parametern treffen.

Diese Entscheidung ist nicht immer einfach und eindeutig zu treffen. Deshalb ist es manchmal möglich, dass ein Angreifer den Warenkorb eines Webshops editieren und die Preise der Artikel ändern kann. Wenn dann noch der gesamte Bestellvorgang automatisiert ist und keine Plausibilitätsprüfung durch einen Menschen erfolgt, hat der Betreiber des Webshops ein ernstes Problem.

**Sessions haben oft spezifische Schwachstellen**

**Gefahr:**

**Angreifer kann die Session eines anderen Benutzers übernehmen und mit den Rechten des Nutzers agieren.**

**HTTP ist stateless, aber jede moderne Anwendung benötigt Zustände.**

**Die möglichen Arten, den Zustand zu übertragen, müssen sicher implementiert sein.**

## **Angreifbarkeit aus dem Netz**

Klassische Applikationen sind nur angreifbar, wenn der Angreifer Zugriff auf den Rechner oder das lokale Netzwerk hat. Dafür gibt es erprobte Schutzmittel wie Türschlösser, User Accounts, Firewalls und Intrusion-Detection Systeme. Leider nutzen diese Techniken im Web Applikations-Bereich nur wenig. Die Applikation soll ja gerade aus der ganzen Welt zugreifbar sein. Das impliziert, dass sie auch aus der ganzen Welt angreifbar ist. Viele Webapplikationen setzen heute auf Standardsoftwarepaketen auf. Ist in diesen erst einmal ein Fehler gefunden, können in kurzer Zeit genau diese Fehler automatisch in allen Webapplikationen ausgenutzt werden. Nahezu jede Website wird heute von Angreifern regelmäßig nach bekannten Schwachstellen oder Konfigurationsfehlern des Administrators gescannt. Ein Blick in die Server-Logfiles lohnt sich immer...

Ein weiteres Problem mit Netzwerkanwendungen generell ist die Anfälligkeit der Verbindung zwischen Client und Server, hier also zwischen dem Webbrowser des Benutzers und dem Webserver, welcher die Applikation ausführt. Wenn nicht besondere Schutzmechanismen wie SSL benutzt werden, liegt die Kommunikation zwischen Benutzer und Server im Klartext vor. Zudem sind Internetverbindungen keine Punkt zu Punkt Verbindungen zwischen dem Client und dem Server. Die Datenpakete einer Verbindung können umgeleitet werden - und ein geschickter Angreifer kann diese über seinen Rechner laufen lassen. Das sind dann die sog. Man-In-The-Middle-Attacken, da der Angreifer zwischen den beiden berechtigten Parteien sitzt.

Technische Lösungen wie Verschlüsselung und Authentisierung mit SSL können hier helfen, werden aber oftmals falsch eingesetzt.

## **Authentizität der Applikation**

Bei Applikationen im Internet weiß der Benutzer im Gegensatz zu lokal auf seinem Computer installierten Applikation nicht genau, mit welchem Server und mit welcher Applikation er verbunden ist. Er kann nur das Aussehen der Applikation beurteilen, und das kann leicht auf einen anderen Server kopiert werden.

Dieses Problem ist unter den Namen Phishing und Pharming in den letzten Jahren mit zu einem Hauptproblem im Web Application Security Bereich geworden.

Die Benutzer werden durch Tricks auf fremde Webseiten gelockt, die der eigentlichen Webseite täuschend ähnlich sind. Der Benutzer wird überredet, auf dieser fremde Seite sensible Daten wie zum Beispiel Kontonummer, PIN und TAN einzugeben.

Auch hier gibt es technische Lösungen. SSL Zertifikate könne auf technischer Seite garantieren, dass der Webbrowser des Benutzers wirklich mit dem richtigen Server verbunden ist. Leider haben sich diese SSL Zertifikate in der Praxis aber als nicht ausreichend herausgestellt, da sie ein höheres Sicherheitsbewusstsein erfordern, als auf Benutzerseite vorhanden ist.

**Web-Applikationen sind im Netz verfügbar, also auch überall aus dem Netz angreifbar.**

**Kommunikation zwischen Client und Server benötigt besonderen Schutz gegen Man-in-The-Middle-Attacken**

**Phishing und Pharming als typische Angriffsmethoden in diesem Bereich**

## 4. Fazit: Web-Anwendungen benötigen speziellen Schutz.

Angriffe auf Web-Anwendungen haben sich in den letzten Monaten zu einem Kernthema im Bereich IT-Sicherheit entwickelt. Herkömmliche IT-Sicherheitslösungen wie Firewalls oder IDS bieten keinen ausreichenden Schutz gegen derartige Angriffe, da ihr Hauptziel der Schutz der Transportschichten ist.

Das Web, speziell das HTTP-Protokoll, wurde nicht konzipiert für komplexe Anwendungen, die heute aber im e-Business aus wirtschaftlichen Gründen eine Notwendigkeit geworden sind. Die daraus resultierenden Schwachstellen von Web-Anwendungen werden noch verstärkt durch die hohe Komplexität, verursacht von der Vielzahl von Web-Scripts, Frameworks und Webtechnologien.

Letztlich erfordert deshalb jede Web-Anwendung ein eigenes, auf ihre jeweilige Logik zugeschnittenes Schutzprofil. Dieses kann derzeit nur ein Produkt erreichen, das mit umfassender Erfahrung im Bereich der Web-Anwendungs-Sicherheit von Grund auf ausschließlich zum Schutz der Web-Anwendungsebene entwickelt wurde.

Wünschenswert ist also ein Web Application Protection System, das Requests speziell auf Business-Logic- und HTTP-Ebene analysiert und darüberhinaus auch den Business Workflow über mehrere Requests hinweg kontrollieren sowie effiziente Gegenmaßnahmen im Falle eines erkannten Angriffs einleiten kann.

**Angriffe auf Web-Anwendungen werden durch klassische Sicherheitsprodukte nicht ausreichend erkannt.**

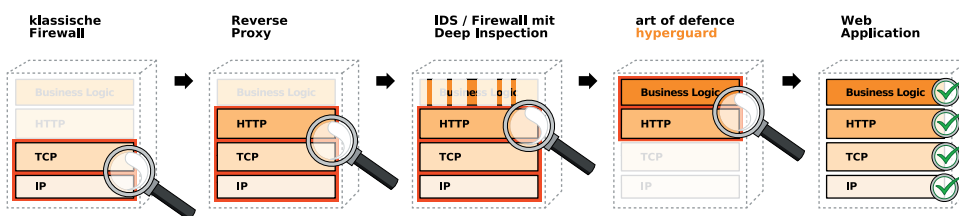
**Business-Logic-Layer und Business Workflow benötigen Schutz durch speziell hierfür entwickelte Protection Systeme**

## 5. Web Application Security mit hyperguard

### Umfassender Schutz auf Web-Anwendungs-Ebene

art of defence **hyperguard** bietet umfassenden Schutz von Web-Anwendungen vor Hacker-Angriffen und Würmern auf Applikationsebene und sichert so die vertraulichen Unternehmensinformationen aus den dahinter liegenden Backend-Systemen wie z. B. SAP-Infrastrukturen vor unerwünschtem Zugriff.

Damit erhöht **hyperguard** den Schutz von Firmennetzwerken durch gängige IT-Sicherheitssysteme wie Firewalls und Intrusion Detection/Prevention Systeme und schließt die derzeit größte Sicherheitslücke einer typischen E-Business-Infrastruktur.



hyperguard =

umfassender Schutz auf  
Anwendungs-Ebene

Schutz auf allen Ebenen dank  
hyperguard

### Kernfunktionalitäten von hyperguard

Die Kernfunktionalitäten von **hyperguard** sind:

- "Härtung" von Web-Anwendungen
- Regelung und Kontrolle des Zugriffs auf Web-Anwendungen
- Auditing, im Sinne von Feststellen, Analyse und Aufbereitung von versuchten Angriffen für Reporting-Zwecke

Kernfunktionalitäten

## Kundennutzen im Überblick

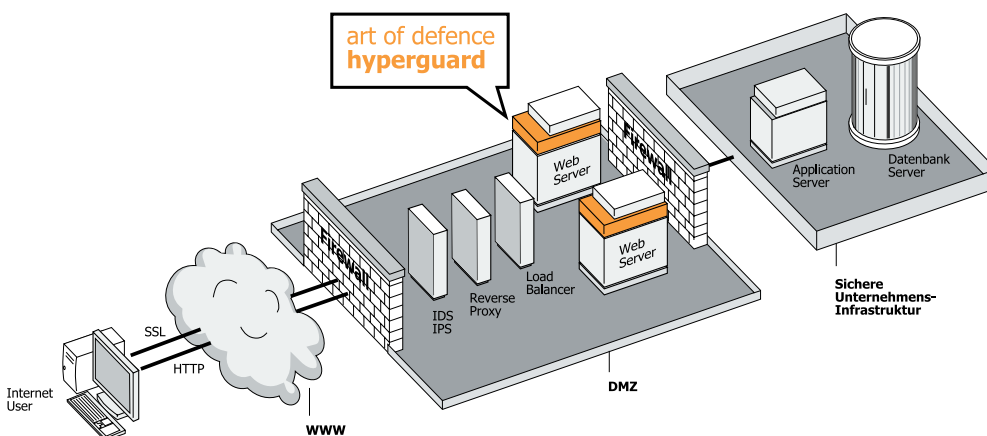
- Erhöhter Schutz von vertraulichen Kundendaten und interner Unternehmensinformation
- Vermeidung „negativer Publicity“
- Nachgewiesene Einhaltung von Gesetzes-Auflagen (Basel II, KonTraG), Industrie-Standards (z.B. PCI) oder Service-Level-Agreements dank detaillierter Reporting-Funktionalitäten
- Einfache Einbindung in bestehende Security-Policies und -Prozesse
- Niedriger Verwaltungsaufwand dank :
  - *automatischer Bedrohungs-Anpassung mittels „Instant-Feedback“*
  - *einfacher Installation ohne Eingriff in die Netzwerkinfrastruktur*
  - *einfachem Konfigurationsmanagement*
  - *hoher Benutzerfreundlichkeit*
- Senkung der Betriebskosten durch:
  - *strukturiertes Einspielen von Updates und einfachere Administration*
  - *Wegfall bzw. deutliche Reduktion von manuellen Plausibilitätskontrollen z.B. bei Online-Bestellungen oder Online-Banking-Transaktionen*

Kundennutzen aus wirtschaftlicher Sicht

## 6. Wie funktioniert hyperguard? Ein Überblick.

### Web-Server Plug-In

**hyperguard** wird als Software-Plug-In in den vorhandenen Webserver installiert. Dadurch sind insbesondere keine Änderungen der bestehenden Netzwerk-Infrastruktur notwendig. Zudem sitzt **hyperguard** genau an der richtigen Stelle, um ohne weiteren Aufwand auch Daten zu analysieren, die bis hierher SSL-verschlüsselt übertragen wurden.



### Optimale Kombination von Regel- und Anomalie-basiertem Schutz

Bevor ein Request eine Web-Anwendung erreicht, wird er von **hyperguard** entgegengenommen und analysiert. Dabei bewertet eine optimierte Kombination von White-/Black- & Grey-List-Verfahren, Mustererkennung von bekannten Angriffen und verschiedenen statistischen Methoden jeden Request hinsichtlich seiner potentiellen Gefährlichkeit.

Jeder Request wird dabei eindeutig einer der drei folgenden Klassen zugeordnet:

- legitime Requests; diese werden an die Web-Anwendung weitergegeben.
- Eindeutige Angriffe; diese werden abgewehrt – zudem werden möglichst viele Daten zur Identifikation und Verfolgung der Angreifer gespeichert.
- Requests, deren Gefährlichkeit lokal nicht endgültig beurteilt werden kann; diese werden je nach lokaler Bewertung und installierter Security-Policy abgelehnt oder weitergegeben. Zudem werden sie intern protokolliert und zur weiteren Bewertung z. B. von nachfolgenden Requests des gleichen Nutzers herangezogen.

Plug-In im Webserver

Keine Änderung der Netzwerk-Infrastruktur

Optimale Einbindung in bestehende E-Business-Infrastruktur

Bewertung von Requests „vor der Web-Anwendung“

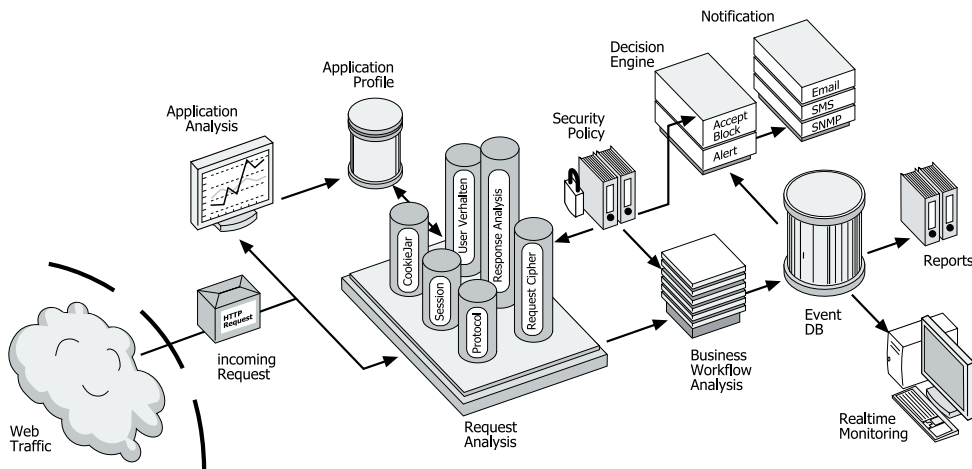
Klassifikation der Requests in drei Klassen:

legitime Anfragen

Angriffe

Nicht eindeutig klassifizierbare Anfragen

## 7. Die Kern-Funktionsbereiche im Überblick



### Die Kern-Funktionsbereiche im Überblick

#### Anmerkung:

In diesem eher technischen Teil werden gezwungenermaßen viele technische Fachausdrücke verwendet. Diese werden im Kapitel "Glossar" näher erläutert.

### Request Verification

**Protocol Anomaly Detection** erkennt jegliche Abweichungen vom HTTP bzw. HTTPS-Standard und lehnt entsprechende Requests ab.

**Request Restriction** erlaubt es, die möglichen Felder eines HTTP-Requests nur auf diejenigen Bereiche einzuschränken, die für die dahinter stehende Web-Anwendung tatsächlich benötigt werden.

**Input Validation** fängt die potentiellen Schwachstellen einer Web-Anwendung auf, die ihre Ursache darin haben, dass der Programmierer Benutzer-Eingaben vertraut. Hier wird insbesondere die ganze Klasse von sog. Command-Injection-Angriffen abgewehrt.

**Malicious Code Detection** erkennt bekannten und bisher unbekanntem Schadcode durch das optimale Zusammenspiel von regel- und anomaliebasierter Analyse.

### Secure Session Handling

**hyperguard** implementiert ein eigenes sicheres Session-Handling durch die Benutzung von kryptographisch gesicherten Session-IDs. Zusätzlich kann auch die sichere SSL Session-ID hinzugezogen werden. Dieses sichere Session-Handling verhindert Angriffe wie Session-ID-Guessing und damit das Session-Hijacking.

**Session Timeouts** sind vom Administrator individuell an das normale Benutzerverhalten der Applikation anpassbar; damit kann das Zeitfenster für mögliche Angriffe auf die Session durch z. B. Session-Riding minimiert werden.

### Detaillierte Analyse von HTTP-Requests auf HTTP- und Business-Logic-Layer

### Eigenes sicheres Session-Handling

### Grundlage zur Absicherung des Business Workflows

Auf der Basis dieses sicheren Session-Handlings bietet **hyperguard** weitere Sicherheitsdienste an. Die wesentliche technische Grundlage hierfür ist, dass jede Session eines Nutzers individuell gehalten und beobachtet wird.

## Secure State Management

### Sicheres Cookie-Handling dank Secure CookieJar.

Applikationen benutzen oft unsichere Cookies, um ihre eigene Session oder den Status der Abarbeitung zu verwalten. **hyperguard** schützt diese unsicheren Applikations-Session-Daten durch eine integrierte Session Integrity Protection. Die Cookies der Applikation werden dabei nicht an den Webbrowser des Nutzers weitergeleitet, sondern bleiben im Secure CookieJar von **hyperguard** und werden dort der bestehenden sicheren Session zugeordnet. Von dort werden sie beim nächsten Request des Nutzers wieder an die Applikation weitergeleitet. So ist eine Manipulation dieser Daten durch einen Angreifer ausgeschlossen.

### Hidden Parameter Protection

ist durch eine Kombination von Session Management und Input Validation gegeben, da auch sog. "hidden parameters" bei jeder Interaktion erneut auf Manipulation geprüft werden.

## Site Usage Enforcement

**Enforcement of Entry Points** ermöglicht es, Usergruppen-spezifisch erlaubte Einstiegsseiten zu definieren. Einstiegsversuche auf anderen, in der Regel "tiefer liegenden" URLs werden erkannt, auf die erlaubte Einstiegsseite umgelenkt und zur weiteren Beobachtung festgehalten. Damit werden insbesondere sog. "Deep Links" auf wertvollen Content z. B. einer Online-Zeitschrift durch einen Wettbewerber verhindert, der damit insbesondere die eigentlich gewünschte Online-Werbung umgeht und zudem den Content für sein Angebot weiter benutzt.

**Verhinderung automatisierter Abfragen durch Skripte.** Automatisierte Anfragen werden durch eine Analyse des Zeitverhaltens der einzelnen Anfragen und des Verhaltens des benutzten User-Agents erkannt.

**Referer Checking** erschwert einerseits "Deep Linking" und kann die Web-Applikation vor Überlastung durch den sog. "slashdot"-Effekt wirksam schützen. Dieser tritt immer dann auf, wenn die Site unerwartet von einer viel gelesenen Nachrichtenseite verlinkt wird. Als Folge treten oft innerhalb von wenigen Stunden Überlast-Erscheinungen durch die ungewohnt vielen Zugriffe auf die Web-Applikation auf. **hyperguard** erkennt einen solchen Vorgang dynamisch und kann ggfs. eine ressourcen-schonende Mitteilung ausgeben.

Ein weiterer Kern-Anwendungsbereich des Referer Checkings ist die im folgenden dargestellte

Basis für weitere Sicherheitsdienste

Sicheres Cookie-Handling

Parameter Protection

Der Nutzer wird auf „gewünschte Bahnen“ gelenkt.

Unerwünschte Nutzer werden abgewehrt oder umgeleitet.

## Serverside Phishing Prevention, Detection and Termination

Dank des Referer Checkings lässt sich insbesondere das Verlinken von Content oder Bildern von anderen Webseiten aus beschränken, wie es oft bei heute "typischen" Phishing-Angriffen benutzt wird. Ein unerwartet starkes Anwachsen von Zugriffen mit einem bisher unbekanntem Referer – oder ganz ohne Referer - führt zur Alarmierung des Administrators und kann zudem automatisiert eine Warnung an den Nutzer ausliefern.

## 8. Hohe Benutzerfreundlichkeit und niedriger Verwaltungsaufwand

**Intuitive web-basierte Benutzeroberfläche** ermöglicht die einfache Definition und Verwaltung der verschiedenen Sicherheitsfunktionalitäten von **hyperguard**. Im sog. "Experten-Modus" können diese auch auf sehr granulearem Niveau eingestellt werden. Damit ist ein flexibles und gleichzeitig einfaches Konfigurationsmanagement gewährleistet.

**Implementierung als Webserver-Plugin** bedeutet, dass keine Änderungen der Netzwerk-Infrastruktur zur Einführung von **hyperguard** notwendig sind.

**Umfassende Reportingfunktionalitäten** bieten z. B. Realtime-Statistiken sowie diverse Auswertungen des Sicherheits-Zustands im zeitlichen Verlauf.

**Optimale Einbindung in bestehende Sicherheitsprozesse** u.a. durch Alerting in vorhandene Monitoring-Systeme. Daneben ist natürlich auch eine Benachrichtigung per SMS, Email oder Fax möglich.

**Skalierbarkeit** ist gegeben, da – als Webserver-Plugin – einfach mit der Zahl der vom Kunden eingesetzten Webserver skaliert. Für eine größere Anzahl von Webservern ist eine zentrale Administration der entsprechenden **hyperguard**- Module möglich.

## 9. Zusammenfassung

**hyperguard** wurde von Grund auf mit dem Hauptziel entwickelt, Web-Anwendungen vor Hacker-Angriffen und Würmern auf Applikationsebene umfassend zu schützen. Damit erhöht **hyperguard** den Schutz von Firmennetzwerken durch gängige IT-Sicherheitssysteme wie Firewalls und Intrusion Detection/Prevention Systeme und schließt die derzeit größte Sicherheitslücke einer typischen E-Business-Infrastruktur.

Die gesamte Software-Architektur von **hyperguard** wurde so gewählt, dass sie einerseits höchstmöglichen Schutz bietet, andererseits aber bestmöglich in bestehende Sicherheits-Infrastrukturen integriert und möglichst geringen Aufwand verursacht.

Erkennen und Beenden von Phishing-Angriffen

soweit auf Seiten des Servers möglich

hyperguard

Optimale Einbindung in bestehende Security-Prozesse mit geringem Aufwand

Beste Übersicht und Skalierbarkeit

Kein Risiko eines Bottlenecks

hyperguard

erweitert den Schutz durch klassische Sicherheitslösungen um den umfassenden Schutz auf Anwendungs-Ebene

## 10. Glossar

Begriff	Definition
<b>Phishing</b>	<p>Das Vortäuschen einer fremden Identität (zum Beispiel einer Bank) um das Opfer dazu zu bewegen, sensible Daten wie Kontonummer, PIN und TAN an den Angreifer preiszugeben.</p> <p>Üblicherweise heute durch eMails realisiert, die das Opfer verleiten, eine in der Mail angegebene URL anzuklicken und dort seine Daten einzugeben. Sowohl die Mail als auch die Website werden im bekannten Design z.B. der Bank gehalten.</p>
<b>Pharming</b>	<p>Das Manipulieren von Einträgen in der DNS-Infrastruktur des Internets. Damit wird die Umwandlung eines Hostnamens wie z.B. „online-banking.eine-bank.de“ auf die im Internet benutzte Addressierung über IP Adressen 10.1.1.42 modifiziert. Ein Angreifer kann dann den Zugriff auf die Seite der Bank auf seine eigenen Server umleiten und damit denselben Effekt wie bei Phishing-Angriffen erreichen.</p>
<b>Command Injection</b>	<p>Das Einschleusen von Kommandos (Befehlen) an ein Drittsystem (Datenbank, LDAP Server, Betriebssystem) über eine Webapplikation, die diese Kommandos als reine Daten einfach an das Drittsystem weiterreicht.</p>
<b>Session-ID-Guessing</b>	<p>Das Erraten des Wertes der Session-ID eines anderen Benutzers. Dies ist immer dann möglich, wenn die Generierung der Session-ID auf Applikationsseite unsicher ist.</p>
<b>Session-Hijacking</b>	<p>Die Übernahme einer Session eines Benutzers durch einen Angreifer. Hierbei kann dann der Angreifer mit den Rechten des Benutzers arbeiten.</p>
<b>Session-Riding</b>	<p>Das Einschmuggeln von Kommandos in eine bestehende Session eines Benutzers. Diese Kommandos werden dann mit den Rechten des angegriffenen Benutzers ausgeführt.</p>
<b>Cookie Manipulation</b>	<p>Das Ändern von Werten der Cookies, die die Webapplikation im Webbrowser des Benutzers hinterlegt, um einen Status zu speichern. Normalerweise sollte der Webbrowser diese Cookies unverändert an die Applikation zurückschicken.</p>

Begriff	Definition
<b>Hidden Parameter Manipulation</b>	Das Ändern von Werten, die als versteckte Eingabefelder auf Formularseiten vorhanden sind. Häufig werden diese benutzt, um einen Status zu speichern. Viele Applikationen verlassen sich darauf, dass der Webbrowser dieselben Werte wieder mitschickt.
<b>Referer</b>	Wenn ein Benutzer im Web auf einen Link oder auf einen Button klickt, dann schickt der Webbrowser die URL der Seite, auf der dieser Link vorhanden war, im sogenannten "Referer" Feld des HTTP-Headers mit. Damit kann eine Webapplikation erfahren, von wo der Benutzer auf die gerade ausgewählte Seite gekommen ist. Auch wenn diese Information leicht manipuliert werden kann, können hier dennoch gewisse Plausibilitätschecks durchgeführt und bestimmte Angriffe Dritter erkannt werden.
<b>Deep Linking</b>	Das Verlinken einer Website auf inhaltliche Details einer zweiten, fremden Site, welche so vom Betreiber der zweiten Site nicht vorgesehen war.
<b>User-Agent</b>	Der Webbrowser des Benutzers.
<b>Brute-Force-Attack</b>	Das Ausprobieren von allen möglichen Kombinationen einer Eingabe oder z. B. einer Session-ID, um so einen gültigen Wert zu erraten.
<b>Session</b>	Die zusammengehörigen Aktionen eines Benutzers in einer Webapplikation. Da das HTTP Protokoll ursprünglich nicht für Sessions vorgesehen war, werden diese von jeder Webapplikation selber implementiert. In der Regel wird jede Session durch eine eindeutige SessionID identifiziert, diese wird durch ein Cookie zum Webbrowser übertragen und in jedem HTTP-Request an die Applikation mitgeschickt. Sämtliche Authentisierungs- und Autorisierungsinformationen sind an diese Session-ID gebunden. Ein Angreifer, der eine fremde Session-ID erraten kann, kann als ein fremder Benutzer agieren.
<b>Application Entry Points</b>	Die URLs, auf denen ein normaler Benutzer die Website betreten sollte; normalerweise die Home-Page.

## **art of defence GmbH**

art of defence bietet Lösungen im Bereich Web-Anwendungs-Sicherheit. Kerngeschäft ist die Entwicklung und der Vertrieb von Software-Produkten zum Schutz von Websites und Datenbanken vor Hacker-Angriffen und Würmern auf Anwendungsebene. Damit ist art of defence erster deutscher Hersteller in diesem Bereich der IT-Sicherheit.

Ausführlichere Informationen finden Sie unter:

**[www.artofdefence.com](http://www.artofdefence.com)**

art of defence GmbH  
Bruderwöhrdstrasse 15 b  
93055 Regensburg

Telefon: +49 (0) 941 604 889 58  
Email: [info@artofdefence.com](mailto:info@artofdefence.com)

**Herausgeber**

**Kontakt**

hyperguard und art of defence sind Markennamen der art of defence GmbH.  
Alle anderen Warenzeichen sind Eigentum der jeweiligen Inhaber.

Copyright 2005-2006