SGCWEBSOCKETS · TECHNICAL DOCUMENT

Firewall

IP-based firewall component for sgcWebSockets servers — block, allow-list and rate-limit by address and CIDR range.

eSeGeCe

WWW.ESEGECE.COM

Overview

TsgcWebSocketFirewall implements a comprehensive firewall component that protects WebSocket servers against a wide range of security threats.

At a glance

COMPONENT CLASS `TsgcWebSocketFirewall`	STANDARDS / SPEC —
TRANSPORTS TCP, TLS	PLATFORMS Windows, macOS, Linux, iOS, Android
FRAMEWORKS VCL, FireMonkey, Lazarus / FPC	EDITION Standard / Professional / Enterprise

Features

- Native Delphi implementation with full ANSI/Unicode support.

Technical specification

Component class	`TsgcWebSocketFirewall` (unit `sgcWebSocket_Firewall`)
Frameworks	VCL, FireMonkey, Lazarus / FPC
Platforms	Windows, macOS, Linux, iOS, Android

Main properties

The principal published / public properties used to configure and drive the component. Consult the online help for the full list.

`Enabled`	Master on/off switch for the whole firewall; when False every check is bypassed and all connections/messages are allowed.
`Blacklist`	Deny list of IP addresses and CIDR ranges whose connections are rejected before any other check.
`Whitelist`	Allow list of trusted IP addresses and CIDR ranges; whitelisted IPs bypass all other firewall checks.
`CustomRules`	User-defined rule engine that combines IP, country, message and violation-count conditions with a configurable action.
`BruteForce`	Tracks failed authentication attempts per IP and bans addresses that exceed a configurable threshold within a time window.
`SQLInjection`	Scans inbound messages for SQL injection patterns and denies, allows or logs them according to Action.
`XSS`	Scans inbound messages for cross-site scripting patterns and denies, allows or logs them according to Action.
`PathTraversal`	Scans inbound messages for directory traversal patterns and denies, allows or logs them according to Action.
`CommandInjection`	Scans inbound messages for shell command injection patterns and denies, allows or logs them according to Action.
`PayloadLimit`	Rejects inbound messages whose size exceeds MaxSizeBytes to protect against oversized payload attacks.

Main methods

The principal public methods exposed by the component.

`IsConnectionAllowed()`	Core handshake filter. Evaluates every enabled connection-time protection module (bans, whitelist, blacklist, rate limit, GeoIP, custom rules) and returns whether the connection should be accepted.
`RegisterConnection()`	Records a new accepted connection for the given IP, incrementing the per-IP counters used by RateLimit and feeding the ThreatScore decay window.
`UnregisterConnection()`	Decrements the per-IP connection counter when a previously registered connection closes, freeing capacity for RateLimit.
`SaveBansToStream()`	Serializes the active ban list into any writable TStream, useful for persistence backends that are not file-based (databases, network sync, encrypted archives).
`LoadBansFromStream()`	Reads ban records from any TStream (memory, network, resource) in the same format produced by SaveBansToStream.
`ResetThreatScore()`	Clears the accumulated threat score for a single IP, giving it a clean slate against AutoBanThreshold without affecting other IPs.
`IsOriginAllowed()`	Checks a WebSocket Origin header against WebSocketProtection.AllowedOrigins (with wildcard support) and returns whether it is accepted.
`IsFrameSizeAllowed()`	Checks a single WebSocket frame size against WebSocketProtection.MaxFrameSize and returns whether it is within the configured limit.
`IsSubprotocolAllowed()`	Checks the requested WebSocket subprotocol name against WebSocketProtection.AllowedSubprotocols and returns whether it is accepted.
`RegisterFailedAttempt()`	Application-level hook that records a failed authentication attempt from the given IP, feeding the BruteForce tracker and potentially triggering an automatic ban.

Public events

The component exposes the following published events; consult the online help for full event-handler signatures.

`OnFiltered`	Fires when the firewall makes a filtering decision on a connection or message.
`OnResolveCountry`	Fires during GeoIP evaluation so the application can supply a custom country code for an IP.
`OnThreatScoreChanged`	property OnThreatScoreChanged: TsgcFirewallOnThreatScoreChanged; // TsgcFirewallOnThreatScoreChanged = procedure(Sender: TObject; const aIP: string; aOldScore, aNewScore: Integer) of object __property...
`OnViolation`	Fires when the firewall detects a specific security violation.

Quick Start

Drop the component on a form, configure the properties below and activate it. The snippet that follows shows the typical **Adding IPs at Runtime** configuration sourced from the online help.

> **About this scenario.** TsgcWebSocketFirewall provides IP-based access control through two complementary mechanisms: a blacklist that blocks specific IPs and a whitelist that grants unconditional access to trusted IPs.

Delphi (VCL / FireMonkey)

```
sgcWebSocketFirewall1.Blacklist.Enabled := True;
sgcWebSocketFirewall1.Blacklist.IPs.Add('192.168.1.100');
sgcWebSocketFirewall1.Blacklist.IPs.Add('10.0.0.0/8');
```

C++ Builder

```
sgcWebSocketFirewall1->Blacklist->Enabled = true;
sgcWebSocketFirewall1->Blacklist->IPs->Add("192.168.1.100");
sgcWebSocketFirewall1->Blacklist->IPs->Add("10.0.0.0/8");
```

.NET (C#)

```
server.Firewall.Blacklist.Enabled = true;
server.Firewall.Blacklist.IPs.Add("192.168.1.100");
server.Firewall.Blacklist.IPs.Add("10.0.0.0/8");
```

Common scenarios

The following scenarios are lifted verbatim from the online help. Each shows the configuration and method calls needed to drive the component through a specific real-world flow.

1 · Creating Rules

TsgcWebSocketFirewall includes a custom rules engine that allows you to define flexible filtering rules with conditions and actions. Rules are evaluated in order; the first matching rule determines the action taken on the connection or message.

Delphi (VCL / FireMonkey)

```
var
  Rule: TsgcFirewallRuleItem;
begin
  // Create a rule to block a specific subnet
  Rule := sgcWebSocketFirewall1.Rules.Add;
  Rule.Name := 'Block internal abuse subnet';
  Rule.Enabled := True;
  Rule.IPPattern := '10.0.5.0/24';
  Rule.ActionType := raDeny;

  // Create a rule to log messages containing "admin"
  Rule := sgcWebSocketFirewall1.Rules.Add;
  Rule.Name := 'Log admin access attempts';
  Rule.Enabled := True;
  Rule.MessagePattern := 'admin';
  Rule.ActionType := raLog;
end;
```

C++ Builder

```
// Create a rule to block a specific subnet
TsgcFirewallRuleItem *Rule = sgcWebSocketFirewall1->Rules->Add();
Rule->Name = "Block internal abuse subnet";
Rule->Enabled = true;
Rule->IPPattern = "10.0.5.0/24";
Rule->ActionType = raDeny;

// Create a rule to log messages containing "admin"
Rule = sgcWebSocketFirewall1->Rules->Add();
Rule->Name = "Log admin access attempts";
Rule->Enabled = true;
Rule->MessagePattern = "admin";
Rule->ActionType = raLog;
```

.NET (C#)

```
// Create a rule to block a specific subnet
var rule = server.Firewall.Rules.Add();
rule.Name = "Block internal abuse subnet";
rule.Enabled = true;
rule.IPPattern = "10.0.5.0/24";
rule.ActionType = TsgcFirewallRuleAction.raDeny;

// Create a rule to log messages containing "admin"
rule = server.Firewall.Rules.Add();
rule.Name = "Log admin access attempts";
rule.Enabled = true;
rule.MessagePattern = "admin";
rule.ActionType = TsgcFirewallRuleAction.raLog;
```

2 · Custom Patterns

TsgcWebSocketFirewall includes built-in pattern-based detection for SQL injection and Cross-Site Scripting (XSS) attacks in WebSocket messages. These modules scan incoming message content and take action when a threat pattern is detected.

Delphi (VCL / FireMonkey)

```
sgcWebSocketFirewall1.SQLInjection.Enabled := True;
sgcWebSocketFirewall1.SQLInjection.CustomPatterns.Add('WAITFOR DELAY');
sgcWebSocketFirewall1.SQLInjection.CustomPatterns.Add('BENCHMARK\(');
sgcWebSocketFirewall1.SQLInjection.CustomPatterns.Add('SLEEP\(');
```

C++ Builder

```
sgcWebSocketFirewall1->SQLInjection->Enabled = true;
sgcWebSocketFirewall1->SQLInjection->CustomPatterns->Add("WAITFOR DELAY");
sgcWebSocketFirewall1->SQLInjection->CustomPatterns->Add("BENCHMARK\\(");
sgcWebSocketFirewall1->SQLInjection->CustomPatterns->Add("SLEEP\\(");
```

```
.NET (C#)
```

```
server.Firewall.SQLInjection.Enabled = true;
server.Firewall.SQLInjection.CustomPatterns.Add("WAITFOR DELAY");
server.Firewall.SQLInjection.CustomPatterns.Add("BENCHMARK\\(");
server.Firewall.SQLInjection.CustomPatterns.Add("SLEEP\\(");
```

3 · Firewall: Advanced Message Protection

TsgcWebSocketFirewall provides additional content inspection modules for payload size limiting, path traversal detection, and command injection detection. These modules complement the SQL injection and XSS detection features to provide comprehensive message-level protection.

```
Delphi (VCL / FireMonkey)
```

```
// Limit messages to 512 KB
sgcWebSocketFirewall1.PayloadLimit.Enabled := True;
sgcWebSocketFirewall1.PayloadLimit.MaxSizeBytes := 524288;
sgcWebSocketFirewall1.PayloadLimit.Action := faDeny;
```

```
C++ Builder
```

```
// Limit messages to 512 KB
sgcWebSocketFirewall1->PayloadLimit->Enabled = true;
sgcWebSocketFirewall1->PayloadLimit->MaxSizeBytes = 524288;
sgcWebSocketFirewall1->PayloadLimit->Action = faDeny;
```

```
.NET (C#)
```

```
// Limit messages to 512 KB
server.Firewall.PayloadLimit.Enabled = true;
server.Firewall.PayloadLimit.MaxSizeBytes = 524288;
server.Firewall.PayloadLimit.Action = TsgcFirewallAction.faDeny;
```

4 · Firewall: Rate Limiting and Flood Protection

TsgcWebSocketFirewall provides two complementary mechanisms for controlling traffic: connection rate limiting (controlling how many connections an IP can open) and message flood protection (controlling how many messages a connected client can send per second).

Delphi (VCL / FireMonkey)

```
// Allow maximum 5 connections per IP
sgcWebSocketFirewall1.RateLimit.Enabled := True;
sgcWebSocketFirewall1.RateLimit.MaxConnectionsPerIP := 5;
sgcWebSocketFirewall1.RateLimit.TimeWindowSec := 60;
```

C++ Builder

```
// Allow maximum 5 connections per IP
sgcWebSocketFirewall1->RateLimit->Enabled = true;
sgcWebSocketFirewall1->RateLimit->MaxConnectionsPerIP = 5;
sgcWebSocketFirewall1->RateLimit->TimeWindowSec = 60;
```

.NET (C#)

```
// Allow maximum 5 connections per IP
server.Firewall.RateLimit.Enabled = true;
server.Firewall.RateLimit.MaxConnectionsPerIP = 5;
server.Firewall.RateLimit.TimeWindowSec = 60;
```

5 · IsOriginAllowed Method

TsgcWebSocketFirewall includes protocol-level protections specific to WebSocket connections: origin validation, frame size limits, and subprotocol filtering. These features protect against cross-site WebSocket hijacking, memory exhaustion, and unauthorized protocol usage.

Delphi (VCL / FireMonkey)

```
if sgcWebSocketFirewall1.IsOriginAllowed('https://app.example.com') then
  Log('Origin is allowed')
else
  Log('Origin is blocked');
```

C++ Builder

```
if (sgcWebSocketFirewall1→IsOriginAllowed("https://app.example.com"))
  Log("Origin is allowed");
else
  Log("Origin is blocked");
```

```
.NET (C#)
```

```
if (server.Firewall.IsOriginAllowed("https://app.example.com"))
    Log("Origin is allowed");
else
    Log("Origin is blocked");
```

6 · Loading a GeoIP Database

TsgcWebSocketFirewall can filter connections by geographic location using country codes. Block or allow entire countries with a single configuration. The component resolves IP addresses to country codes using a loaded database or a custom event handler.

```
Delphi (VCL / FireMonkey)
```

```
sgcWebSocketFirewall1.LoadGeoIPDatabase('C:\data\geoip.csv');
```

```
C++ Builder
```

```
sgcWebSocketFirewall1→LoadGeoIPDatabase("C:\\data\\geoip.csv");
```

```
.NET (C#)
```

```
server.Firewall.LoadGeoIPDatabase("C:\\data\\geoip.csv");
```

Sources used to build this document

Every external claim links back to a primary source. The online-help references decode the canonical deep-link the company maintains for this component.

Online help — component page	www.esegece.com/help/sgcWebSockets/Components/TsgcWebSocketFirewall.htm
Delphi demo project (in the sgcWebSockets package)	`Demos\04.WebSocket_Other_Samples\13.Firewall`
Component page	www.esegece.com/products/websockets/components/resilience/firewall/
Product page	www.esegece.com/products/websockets/

> **Document scope.** This document covers the publicly-documented surface of the Firewall component shipped with sgcWebSockets. For full property, method and event reference consult the online help linked above.