Namespace IDSApp

Classes

IDSCore

Main Intrusion Detection System core engine that coordinates packet capture, processing, rule matching, and alert generation. Provides real-time network traffic analysis and threat detection using signature-based and behavioral analysis techniques.

Key Responsibilities:

- Packet capture from network interfaces or PCAP files
- Multi-threaded packet processing with worker queues
- Rule-based threat detection using enhanced signature engine
- Protocol parsing and analysis (HTTP, DNS, SSH, FTP, etc.)
- Alert generation and security event logging
- Performance monitoring and system statistics collection
- Flow tracking and behavioral analysis
- DDoS and port scan detection
- Resource management and cleanup

PacketCaptureWrapper

Wrapper class for packet capture data that provides structured access to packet information and metadata for efficient processing

PerformanceMonitor

Monitors and reports system performance metrics including packet processing statistics, memory usage, and rule matching performance. Generates periodic performance reports to help identify bottlenecks and optimize system performance.

Class IDSCore

Namespace: IDSApp

Assembly: IDSApp.dll

Main Intrusion Detection System core engine that coordinates packet capture, processing, rule matching, and alert generation. Provides real-time network traffic analysis and threat detection using signature-based and behavioral analysis techniques.

Key Responsibilities:

- Packet capture from network interfaces or PCAP files
- Multi-threaded packet processing with worker queues
- Rule-based threat detection using enhanced signature engine
- Protocol parsing and analysis (HTTP, DNS, SSH, FTP, etc.)
- Alert generation and security event logging
- Performance monitoring and system statistics collection
- Flow tracking and behavioral analysis
- DDoS and port scan detection
- Resource management and cleanup

```
public class IDSCore : IDisposable
```

Inheritance

object ← IDSCore

Implements

IDisposable

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

IDSCore()

Initializes a new instance of the IDS core with default configuration and performance settings. Sets up rule engines, protocol parsers, and internal data structures for efficient packet processing.

```
public IDSCore()
```

Remarks

Initialization Process:

1. Creates rule statistics tracker for performance monitoring
2. Calculates optimal worker thread count based on system resources
3. Initializes packet processing queue with configured capacity
4. Prepares protocol parsers for detailed traffic analysis
5. Sets up internal data structures for flow tracking and alerting

Methods

Dispose()

Releases all resources used by the IDS core and ensures proper shutdown

```
public void Dispose()
```

Remarks

Resource Cleanup:

1. Disposes of cancellation tokens and timers
2. Releases capture device resources
3. Clears all internal data structures
4. Disposes performance monitoring components
5. Suppresses finalization for proper garbage collection

GetMetrics()

Gets comprehensive system metrics for monitoring, analysis, and integration with external monitoring systems

```
public Dictionary<string, object> GetMetrics()
```

Returns

Dictionary<string, object>

Dictionary containing system metrics and performance indicators

Remarks

Available Metrics:

1. System state and operational status
2. Packet processing statistics and rates
3. Rule matching efficiency and counts
4. Error counts and system health
5. Resource usage and queue status
6. Worker thread information and load distribution
7. Memory consumption and garbage collection stats

GetStatus()

Gets current system status for monitoring and health checks

```
public string GetStatus()
```

Returns

string

Formatted status string with key performance indicators

IsInternalTraffic(string, string)

Determines if traffic between two IP addresses is internal network traffic

```
public bool IsInternalTraffic(string srcIp, string dstIp)
```

Parameters

`srcIp` string

Source IP address

`dstIp` string

Destination IP address

Returns

bool

True if both IPs are internal, otherwise false

Start()

Starts the IDS core system including packet capture, worker threads, and monitoring timers. Initializes rule engines and begins processing network traffic for threat detection.

```
public void Start()
```

Remarks

Startup Sequence:

1. Validates system state and prevents multiple startups
2. Initializes and loads detection rules from database
3. Creates worker threads for parallel packet processing
4. Starts packet capture from configured source (live interface or PCAP file)
5. Activates cleanup and statistics monitoring timers
6. Enables performance monitoring and system health checks

Exceptions

InvalidOperationException

Thrown when IDS is already running

Exception

Thrown when packet capture initialization fails

Stop()

Stops the IDS core system gracefully by terminating packet capture, stopping worker threads, and cleaning up resources. Ensures all pending packets are processed before shutdown.

```
public void Stop()
```

Remarks

Shutdown Sequence:

1. Stops packet capture from network interface
2. Cancels worker threads with graceful termination
3. Completes packet queue to prevent new additions
4. Waits for worker threads to finish processing
5. Stops maintenance timers and cleans up resources
6. Generates final performance statistics and reports

Class PacketCaptureWrapper

Namespace: IDSApp

Assembly: IDSApp.dll

Wrapper class for packet capture data that provides structured access to packet information and metadata for efficient processing

```
public class PacketCaptureWrapper
```

Inheritance

object ← PacketCaptureWrapper

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

PacketCaptureWrapper(RawCapture)

Initializes a new packet capture wrapper with raw packet data

```
public PacketCaptureWrapper(RawCapture raw)
```

Parameters

`raw` RawCapture

Raw packet capture data from capture library

Properties

Data

```
public byte[] Data { get; }
```

Property Value

byte[]

LinkLayerType

```
public LinkLayers LinkLayerType { get; }
```

Property Value

LinkLayers

Timestamp

```
public DateTime Timestamp { get; }
```

Property Value

DateTime

Class PerformanceMonitor

Namespace: IDSApp

Assembly: IDSApp.dll

Monitors and reports system performance metrics including packet processing statistics, memory usage, and rule matching performance. Generates periodic performance reports to help identify bottlenecks and optimize system performance.

```
public class PerformanceMonitor
```

Inheritance

object ← PerformanceMonitor

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

PerformanceMonitor()

Initializes a new instance of the PerformanceMonitor with default 60-second reporting interval

```
public PerformanceMonitor()
```

Methods

Dispose()

Disposes the performance monitor and stops the reporting timer

```
public void Dispose()
```

GeneratePerformanceReport()

Generates and logs a comprehensive performance report including key metrics such as packets processed, memory usage, and rule statistics

```
public void GeneratePerformanceReport()
```

RecordMetric(string, long)

Records a performance metric with the specified name and value

```
public void RecordMetric(string metricName, long value)
```

Parameters

`metricName` string

The name of the metric to record

`value` long

The value to add to the metric counter

Namespace IDSApp.BLL

Classes

SettingBLL

Provides business logic layer operations for application settings management with built-in caching for performance. This class serves as a facade to the data access layer with additional caching capabilities for frequently accessed settings.

Class SettingBLL

Namespace: IDSApp.BLL

Assembly: IDSApp.dll

Provides business logic layer operations for application settings management with built-in caching for performance. This class serves as a facade to the data access layer with additional caching capabilities for frequently accessed settings.

```
public static class SettingBLL
```

Inheritance

object ← SettingBLL

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

Delete(int)

Deletes a specific setting from the system and clears the settings cache.

```
public static bool Delete(int id)
```

Parameters

`id` int

The SettingID of the setting to delete.

Returns

bool

true if the setting was successfully deleted; otherwise, false.

GetAll()

Retrieves all settings from the system.

```
public static SettingCollection GetAll()
```

Returns

SettingCollection

A collection of Settings objects containing all configuration settings in the system.

GetAllCachedSettings()

Retrieves all settings as a dictionary from the cache.

```
public static Dictionary<string, string> GetAllCachedSettings()
```

Returns

Dictionary<string, string>

A dictionary containing all cached setting names and values.

GetById(int)

Retrieves a specific setting by its unique identifier.

```
public static Settings GetById(int id)
```

Parameters

`id` int

The SettingID of the setting to retrieve.

Returns

Settings

A Settings object if found; otherwise, null.

GetByName(string)

Retrieves a specific setting by its name.

```
public static Settings GetByName(string name)
```

Parameters

`name` string

The name of the setting to retrieve.

Returns

Settings

A Settings object if found; otherwise, null.

GetDeauthThreshold()

Retrieves the deauthentication attack detection threshold.

```
public static int GetDeauthThreshold()
```

Returns

int

The number of deauth frames required to trigger an alert.

GetEnablePerformanceLogging()

Retrieves the performance logging enablement setting.

```
public static bool GetEnablePerformanceLogging()
```

Returns

[bool]

true if performance logging is enabled; otherwise, false.

GetHttpBodyThreatThreshold()

Retrieves the HTTP body threat detection threshold.

```
public static int GetHttpBodyThreatThreshold()
```

Returns

[int]

The number of threat indicators in HTTP body required to trigger an alert.

GetInternalIpPrefix()

Retrieves the internal IP address prefix used for network segmentation.

```
public static string GetInternalIpPrefix()
```

Returns

[string]

The IP address prefix for internal network addresses.

GetNetworkInterface()

Retrieves the network interface setting for packet capture.

```
public static string GetNetworkInterface()
```

Returns

string

The name of the network interface to monitor.

GetPortScanThreshold()

Retrieves the port scan detection threshold.

```
public static int GetPortScanThreshold()
```

Returns

int

The number of port scan attempts required to trigger an alert.

GetSetting(string, string)

Retrieves a setting value from cache with optional default value.

```
public static string GetSetting(string key, string defaultValue = "")
```

Parameters

`key` string

The name of the setting to retrieve.

`defaultValue` string

The default value to return if the setting is not found.

Returns

string

The setting value if found; otherwise, the specified default value.

GetSetting<T>(string, T)

Retrieves a typed setting value from cache with optional default value.

```
public static T GetSetting<T>(string key, T defaultValue = default)
```

Parameters

`key` string

The name of the setting to retrieve.

`defaultValue` T

The default value to return if the setting is not found or conversion fails.

Returns

T

The typed setting value if found and convertible; otherwise, the specified default value.

Type Parameters

`T`

The type to convert the setting value to (e.g., int, bool, double).

Insert(string, string, string, string, string, DateTime)

Creates a new setting in the system and clears the settings cache.

```
public static bool Insert(string settingName, string settingValue, string dataType, string
category, string desc, DateTime lastModified)
```

Parameters

`settingName` string

The unique name identifier for the setting.

`settingValue` string

The value of the setting.

`dataType` string

The data type of the setting value (e.g., String, Integer, Boolean).

`category` string

The category grouping for the setting (e.g., Network, Security, Performance).

`desc` string

A description of what the setting controls.

`lastModified` DateTime

The date and time when the setting was last modified.

Returns

bool

true if the setting was successfully created; otherwise, false.

RefreshSettingsCache()

Forces a refresh of the settings cache by clearing and reloading all settings.

```
public static void RefreshSettingsCache()
```

Update(int, string, string, string, string, string, DateTime)

Updates an existing setting and clears the settings cache.

```
public static int Update(int id, string settingName, string settingValue, string dataType,
```

```
  string category, string desc, DateTime lastModified)
```

Parameters

`id` int

The SettingID of the setting to update.

`settingName` string

The updated setting name.

`settingValue` string

The updated setting value.

`dataType` string

The updated data type.

`category` string

The updated category.

`desc` string

The updated description.

`lastModified` DateTime

The updated last modified timestamp.

Returns

int

The number of rows affected by the update operation.

Namespace IDSApp.Collection

Classes

SettingCollection

Class SettingCollection

Namespace: IDSApp.Collection

Assembly: IDSApp.dll

```
public class SettingCollection : List<Settings>, IList<Settings>, ICollection<Settings>,
IReadOnlyList<Settings>, IReadOnlyCollection<Settings>, IEnumerable<Settings>, IList,
ICollection, IEnumerable
```

Inheritance

object ← List<Settings> ← SettingCollection

Implements

IList<Settings>, ICollection<Settings>, IReadOnlyList<Settings>, IReadOnlyCollection<Settings>, IEnumerable<Settings>, IList, ICollection, IEnumerable

Inherited Members

List<Settings>.Add(Settings) , List<Settings>.AddRange(IEnumerable<Settings>) , List<Settings>.AsReadOnly() , List<Settings>.BinarySearch(int, int, Settings, IComparer<Settings>) , List<Settings>.BinarySearch(Settings) , List<Settings>.BinarySearch(Settings, IComparer<Settings>) , List<Settings>.Clear() , List<Settings>.Contains(Settings) , List<Settings>.ConvertAll<TOutput>(Converter<Settings, TOutput>) , List<Settings>.CopyTo(int, Settings[], int, int) , List<Settings>.CopyTo(Settings[]) , List<Settings>.CopyTo(Settings[], int) , List<Settings>.EnsureCapacity(int) , List<Settings>.Exists(Predicate<Settings>) , List<Settings>.Find(Predicate<Settings>) , List<Settings>.FindAll(Predicate<Settings>) , List<Settings>.FindIndex(int, int, Predicate<Settings>) , List<Settings>.FindIndex(int, Predicate<Settings>) , List<Settings>.FindIndex(Predicate<Settings>) , List<Settings>.FindLast(Predicate<Settings>) , List<Settings>.FindLastIndex(int, int, Predicate<Settings>) , List<Settings>.FindLastIndex(int, Predicate<Settings>) , List<Settings>.FindLastIndex(Predicate<Settings>) , List<Settings>.ForEach(Action<Settings>) , List<Settings>.GetEnumerator() , List<Settings>.GetRange(int, int) , List<Settings>.IndexOf(Settings) , List<Settings>.IndexOf(Settings, int) , List<Settings>.IndexOf(Settings, int, int) , List<Settings>.Insert(int, Settings) , List<Settings>.InsertRange(int, IEnumerable<Settings>) , List<Settings>.LastIndexOf(Settings) , List<Settings>.LastIndexOf(Settings, int) , List<Settings>.LastIndexOf(Settings, int, int) , List<Settings>.Remove(Settings) , List<Settings>.RemoveAll(Predicate<Settings>) , List<Settings>.RemoveAt(int) , List<Settings>.RemoveRange(int, int) , List<Settings>.Reverse() , List<Settings>.Reverse(int, int) , List<Settings>.Slice(int, int) , List<Settings>.Sort() ,

List<Settings>.Sort(IComparer<Settings>), List<Settings>.Sort(Comparison<Settings>), List<Settings>.Sort(int, int, IComparer<Settings>), List<Settings>.ToArray(), List<Settings>.TrimExcess(), List<Settings>.TrueForAll(Predicate<Settings>), List<Settings>.Capacity, List<Settings>.Count, List<Settings>.this[int], object.Equals(object), object.Equals(object, object), object.GetHashCode(), object.GetType(), object.MemberwiseClone(), object.ReferenceEquals(object, object), object.ToString()

Namespace IDSApp.DAL

Classes

SettingDal

Class SettingDal

Namespace: IDSApp.DAL

Assembly: IDSApp.dll

```
public static class SettingDal
```

Inheritance

object ← SettingDal

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

Delete(int)

```
public static bool Delete(int id)
```

Parameters

`id` int

Returns

bool

GetAll()

```
public static SettingCollection GetAll()
```

Returns

SettingCollection

GetById(int)

```
public static Settings GetById(int id)
```

Parameters

`id` int

Returns

Settings

GetByName(string)

```
public static Settings GetByName(string name)
```

Parameters

`name` string

Returns

Settings

GetInternalIpPrefix()

```
public static string GetInternalIpPrefix()
```

Returns

string

Insert(string, string, string, string, string, DateTime)

```
public static bool Insert(string settingKey, string settingValue, string dataType, string
category, string desc, DateTime lastModified)
```

Parameters

settingKey string

settingValue string

dataType string

category string

desc string

lastModified DateTime

Returns

bool

Update(int, string, string, string, string, string, DateTime)

```
public static int Update(int id, string settingKey, string settingValue, string dataType,
string category, string desc, DateTime lastModified)
```

Parameters

id int

settingKey string

settingValue string

dataType string

category string

desc string

lastModified DateTime

Returns

int

Namespace IDSApp.Entity

Classes

DbLog

Represents a database operation log entry captured by the Intrusion Detection System. Contains detailed information about database queries, connections, and operations for security monitoring.

DhcpLog

Represents a DHCP (Dynamic Host Configuration Protocol) log entry captured by the Intrusion Detection System. Contains detailed information about DHCP network transactions for monitoring IP address allocation and potential network threats.

LdapLog

Represents an LDAP (Lightweight Directory Access Protocol) log entry captured by the Intrusion Detection System. Contains detailed information about directory service operations for monitoring authentication, directory queries, and detecting unauthorized access attempts or directory service attacks.

NetbiosLog

Represents a NetBIOS (Network Basic Input/Output System) log entry captured by the Intrusion Detection System. Contains information about NetBIOS name resolution and service discovery operations for monitoring Windows network browsing, host discovery, and detecting network reconnaissance activities.

Settings

Represents a configuration setting for the Intrusion Detection System. Stores system parameters, thresholds, and operational configurations that control IDS behavior and functionality.

Signatures

Represents a security signature used for intrusion detection. Contains information about attack patterns, network traffic characteristics, and detection rules.

TftpLog

Represents a log entry for TFTP (Trivial File Transfer Protocol) activity. Contains information about the operation, file transferred, transfer size, source and destination IPs, timestamp, session, and status.

Class DbLog

Namespace: IDSApp.Entity

Assembly: IDSApp.dll

Represents a database operation log entry captured by the Intrusion Detection System. Contains detailed information about database queries, connections, and operations for security monitoring.

```
public class DbLog
```

Inheritance

object ← DbLog

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

DbLog()

Initializes a new instance of the DbLog class with default values.

```
public DbLog()
```

DbLog(DbLog)

Initializes a new instance of the DbLog class as a copy of an existing DbLog object.

```
public DbLog(DbLog d)
```

Parameters

`d` DbLog

Source DbLog object to copy from

DbLog(int, int, string, string, string, string, string, string, string, DateTime, string, string, decimal)

Initializes a new instance of the DbLog class with specified parameters.

```
public DbLog(int dbLogId, int logId, string engine, string command, string databaseName,
string username, string queryText, string sourceIp, string destinationIp, DateTime
timestamp, string sessionId, string status, decimal executionTime)
```

Parameters

`dbLogId` int

Unique identifier for the database log entry

`logId` int

Reference identifier linking to the main system log

`engine` string

Database engine type

`command` string

Database command type

`databaseName` string

Name of the target database

`username` string

Username that executed the operation

`queryText` string

Full SQL query text

`sourceIp` string

Source IP address of the request

`destinationIp` string

Destination IP address of database server

`timestamp` DateTime

When the operation occurred

`sessionId` string

Database session identifier

`status` string

Execution status of the query

`executionTime` decimal

Time taken to execute the query

Properties

Command

Database command type (e.g., SELECT, INSERT, UPDATE, DELETE, DROP, CREATE)

```
public string Command { get; set; }
```

Property Value

string

DatabaseName

Name of the database where the operation was performed

```
public string DatabaseName { get; set; }
```

Property Value

string

DbLogId

Unique identifier for the database log entry

```
public int DbLogId { get; set; }
```

Property Value

int

DestinationIp

Destination IP address of the database server

```
public string DestinationIp { get; set; }
```

Property Value

string

Engine

Database engine type (e.g., SQL Server, MySQL, Oracle, PostgreSQL)

```
public string Engine { get; set; }
```

Property Value

string

ExecutionTime

Time taken to execute the query in milliseconds or seconds

```
public decimal ExecutionTime { get; set; }
```

Property Value

decimal

LogId

Reference identifier linking to the main system log entry

```
public int LogId { get; set; }
```

Property Value

int

QueryText

Full SQL query text that was executed

```
public string QueryText { get; set; }
```

Property Value

string

SessionId

Database session identifier for tracking connection sessions

```
public string SessionId { get; set; }
```

Property Value

string

SourceIp

Source IP address where the database request originated

```
public string SourceIp { get; set; }
```

Property Value

string

Status

Execution status of the query (e.g., Success, Failed, Timeout, Error)

```
public string Status { get; set; }
```

Property Value

string

Timestamp

Exact timestamp when the database operation occurred

```
public DateTime Timestamp { get; set; }
```

Property Value

DateTime

Username

Username of the account that executed the database operation

```
public string Username { get; set; }
```

Property Value

string

Methods

Clone()

Creates a deep copy of the current DbLog instance.

```
public DbLog Clone()
```

Returns

DbLog

A new DbLog object that is an exact copy of the current instance

Class DhcpLog

Namespace: IDSApp.Entity

Assembly: IDSApp.dll

Represents a DHCP (Dynamic Host Configuration Protocol) log entry captured by the Intrusion Detection System. Contains detailed information about DHCP network transactions for monitoring IP address allocation and potential network threats.

```
public class DhcpLog
```

Inheritance

object ← DhcpLog

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

DhcpLog()

Initializes a new instance of the DhcpLog class with default values.

```
public DhcpLog()
```

DhcpLog(DhcpLog)

Initializes a new instance of the DhcpLog class as a copy of an existing DhcpLog object.

```
public DhcpLog(DhcpLog d)
```

Parameters

`d` DhcpLog

Source DhcpLog object to copy from

DhcpLog(int, int, string, string, string, string, string, string, string, DateTime, string, string, int)

Initializes a new instance of the DhcpLog class with specified parameters.

```
public DhcpLog(int dhcpLogId, int logId, string messageType, string transactionId, string clientIp, string offeredIp, string serverIp, string sourceIp, string destinationIp, DateTime timestamp, string sessionId, string status, int leaseDuration)
```

Parameters

`dhcpLogId` int

Unique identifier for the DHCP log entry

`logId` int

Reference identifier linking to the main system log

`messageType` string

DHCP message type

`transactionId` string

Unique transaction ID for request-response correlation

`clientIp` string

Current IP address of the DHCP client

`offeredIp` string

IP address being offered or assigned

`serverIp` string

IP address of the DHCP server

`sourceIp` string

Source IP address of the DHCP packet

`destinationIp` string

Destination IP address of the DHCP packet

`timestamp` DateTime

When the DHCP transaction occurred

`sessionId` string

Session identifier for tracking conversations

`status` string

Transaction status

`leaseDuration` int

Duration of IP address lease in seconds

Properties

ClientIp

Current IP address of the DHCP client (may be 0.0.0.0 for initial requests)

```
public string ClientIp { get; set; }
```

Property Value

string

DestinationIp

Destination IP address of the DHCP packet

```
public string DestinationIp { get; set; }
```

Property Value

string

DhcpLogId

Unique identifier for the DHCP log entry

```
public int DhcpLogId { get; set; }
```

Property Value

int

LeaseDuration

Duration of the IP address lease in seconds

```
public int LeaseDuration { get; set; }
```

Property Value

int

LogId

Reference identifier linking to the main system log entry

```
public int LogId { get; set; }
```

Property Value

int

MessageType

DHCP message type (e.g., DISCOVER, OFFER, REQUEST, ACK, NAK, RELEASE)

```
public string MessageType { get; set; }
```

Property Value

string

OfferedIp

IP address being offered or assigned to the client

```
public string OfferedIp { get; set; }
```

Property Value

string

ServerIp

IP address of the DHCP server handling the request

```
public string ServerIp { get; set; }
```

Property Value

string

SessionId

Session identifier for tracking DHCP conversations

```
public string SessionId { get; set; }
```

Property Value

string

SourceIp

Source IP address of the DHCP packet

```
public string SourceIp { get; set; }
```

Property Value

string

Status

Transaction status (e.g., Success, Failed, Pending, Error)

```
public string Status { get; set; }
```

Property Value

string

Timestamp

Timestamp when the DHCP transaction occurred

```
public DateTime Timestamp { get; set; }
```

Property Value

DateTime

TransactionId

Unique transaction ID to correlate DHCP request-response pairs

```
public string TransactionId { get; set; }
```

Property Value

string

Methods

Clone()

Creates a deep copy of the current DhcpLog instance.

```
public DhcpLog Clone()
```

Returns

DhcpLog

A new DhcpLog object that is an exact copy of the current instance

Class LdapLog

Namespace: IDSApp.Entity

Assembly: IDSApp.dll

Represents an LDAP (Lightweight Directory Access Protocol) log entry captured by the Intrusion Detection System. Contains detailed information about directory service operations for monitoring authentication, directory queries, and detecting unauthorized access attempts or directory service attacks.

```
public class LdapLog
```

Inheritance

object ← LdapLog

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

LdapLog()

Initializes a new instance of the LdapLog class with default values.

```
public LdapLog()
```

LdapLog(LdapLog)

Initializes a new instance of the LdapLog class as a copy of an existing LdapLog object.

```
public LdapLog(LdapLog l)
```

Parameters

`l` LdapLog

Source LdapLog object to copy from

LdapLog(int, int, string, string, string, string, string, DateTime, string, string)

Initializes a new instance of the LdapLog class with specified parameters.

```
public LdapLog(int ldapLogId, int logId, string operation, string distinguishedName, string
resultCode, string sourceIp, string destinationIp, DateTime timestamp, string sessionId,
string status)
```

Parameters

`ldapLogId` int

Unique identifier for the LDAP log entry

`logId` int

Reference identifier linking to the main system log

`operation` string

LDAP operation performed

`distinguishedName` string

Distinguished Name of the directory object

`resultCode` string

LDAP result code indicating operation outcome

`sourceIp` string

Source IP address of the client

`destinationIp` string

Destination IP address of the LDAP server

`timestamp` DateTime

When the LDAP operation occurred

`sessionId` string

Unique session identifier for tracking connections

`status` string

Operation status

Properties

DestinationIp

Destination IP address of the LDAP directory server

```
public string DestinationIp { get; set; }
```

Property Value

string

DistinguishedName

Distinguished Name (DN) of the directory object being accessed or modified

```
public string DistinguishedName { get; set; }
```

Property Value

string

LdapLogId

Unique identifier for the LDAP log entry

```
public int LdapLogId { get; set; }
```

Property Value

int

LogId

Reference identifier linking to the main system log entry

```
public int LogId { get; set; }
```

Property Value

int

Operation

LDAP operation performed (e.g., Bind, Search, Add, Delete, Modify, Compare)

```
public string Operation { get; set; }
```

Property Value

string

ResultCode

LDAP result code indicating operation outcome (e.g., success, invalid credentials, no such object)

```
public string ResultCode { get; set; }
```

Property Value

string

SessionId

Unique session identifier for tracking LDAP client connections

```
public string SessionId { get; set; }
```

Property Value

string

SourceIp

Source IP address of the client making the LDAP request

```
public string SourceIp { get; set; }
```

Property Value

string

Status

Operation status (e.g., Success, Failed, Partial, Error, Timeout)

```
public string Status { get; set; }
```

Property Value

string

Timestamp

Timestamp when the LDAP operation occurred

```
public DateTime Timestamp { get; set; }
```

Property Value

DateTime

Methods

Clone()

Creates a deep copy of the current LdapLog instance.

```
public LdapLog Clone()
```

Returns

LdapLog

A new LdapLog object that is an exact copy of the current instance

Class NetbiosLog

Namespace: IDSApp.Entity

Assembly: IDSApp.dll

Represents a NetBIOS (Network Basic Input/Output System) log entry captured by the Intrusion Detection System. Contains information about NetBIOS name resolution and service discovery operations for monitoring Windows network browsing, host discovery, and detecting network reconnaissance activities.

```
public class NetbiosLog
```

Inheritance

object ← NetbiosLog

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

NetbiosLog()

Initializes a new instance of the NetbiosLog class with default values.

```
public NetbiosLog()
```

NetbiosLog(NetbiosLog)

Initializes a new instance of the NetbiosLog class as a copy of an existing NetbiosLog object.

```
public NetbiosLog(NetbiosLog n)
```

Parameters

`n` NetbiosLog

Source NetbiosLog object to copy from

NetbiosLog(int, int, string, string, string, string, string, string, DateTime, string, string)

Initializes a new instance of the NetbiosLog class with specified parameters.

```
public NetbiosLog(int netbiosLogId, int logId, string queryName, string queryType, string response, string responderIp, string sourceIp, string destinationIp, DateTime timestamp, string sessionId, string status)
```

Parameters

`netbiosLogId` int

Unique identifier for the NetBIOS log entry

`logId` int

Reference identifier linking to the main system log

`queryName` string

NetBIOS name being queried or resolved

`queryType` string

Type of NetBIOS query

`response` string

Response data from the NetBIOS query

`responderIp` string

IP address of the system responding to the query

`sourceIp` string

Source IP address initiating the query

`destinationIp` string

Destination IP address for the query

`timestamp` DateTime

When the NetBIOS operation occurred

`sessionId` string

Unique session identifier for tracking conversations

`status` string

Operation status

Properties

DestinationIp

Destination IP address for the NetBIOS query (often broadcast address)

```
public string DestinationIp { get; set; }
```

Property Value

string

LogId

Reference identifier linking to the main system log entry

```
public int LogId { get; set; }
```

Property Value

int

NetbiosLogId

Unique identifier for the NetBIOS log entry

```
public int NetbiosLogId { get; set; }
```

Property Value

int

QueryName

NetBIOS name being queried or resolved

```
public string QueryName { get; set; }
```

Property Value

string

QueryType

Type of NetBIOS query (e.g., NBSTAT, NBTNAME, SESSION, DATAGRAM)

```
public string QueryType { get; set; }
```

Property Value

string

ResponderIp

IP address of the system responding to the NetBIOS query

```
public string ResponderIp { get; set; }
```

Property Value

string

Response

Response data from the NetBIOS query, including resolved names and services

```
public string Response { get; set; }
```

Property Value

string

SessionId

Unique session identifier for tracking NetBIOS conversations

```
public string SessionId { get; set; }
```

Property Value

string

SourceIp

Source IP address initiating the NetBIOS query

```
public string SourceIp { get; set; }
```

Property Value

string

Status

Operation status (e.g., Success, Failed, Timeout, NoResponse)

```
public string Status { get; set; }
```

Property Value

string

Timestamp

Timestamp when the NetBIOS operation occurred

```
public DateTime Timestamp { get; set; }
```

Property Value

DateTime

Methods

Clone()

Creates a deep copy of the current NetbiosLog instance.

```
public NetbiosLog Clone()
```

Returns

NetbiosLog

A new NetbiosLog object that is an exact copy of the current instance

Class Settings

Namespace: IDSApp.Entity

Assembly: IDSApp.dll

Represents a configuration setting for the Intrusion Detection System. Stores system parameters, thresholds, and operational configurations that control IDS behavior and functionality.

```
public class Settings
```

Inheritance

object ← Settings

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Category

Category grouping for organizational purposes (e.g., "Detection", "Logging", "Network", "Performance")

```
public string Category { get; set; }
```

Property Value

string

DataType

Data type of the setting value (e.g., "int", "bool", "string", "double", "DateTime")

```
public string DataType { get; set; }
```

Property Value

string

Desc

Description explaining the purpose and usage of the setting

```
public string Desc { get; set; }
```

Property Value

string

Id

Unique identifier for the setting record

```
public int Id { get; set; }
```

Property Value

int

LastModified

Timestamp when this setting was last modified

```
public DateTime LastModified { get; set; }
```

Property Value

DateTime

SettingName

Name of the configuration setting (e.g., "DetectionThreshold", "LogRetentionDays", "AlertEmailEnabled")

```
public string SettingName { get; set; }
```

Property Value

string

SettingValue

Current value of the setting as a string (will be parsed according to DataType)

```
public string SettingValue { get; set; }
```

Property Value

string

Methods

Clone()

Creates a deep copy of the current Settings instance.

```
public Settings Clone()
```

Returns

Settings

A new Settings object that is an exact copy of the current instance

Class Signatures

Namespace: IDSApp.Entity

Assembly: IDSApp.dll

Represents a security signature used for intrusion detection. Contains information about attack patterns, network traffic characteristics, and detection rules.

```
public class Signatures
```

Inheritance

object ← Signatures

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

AttackName

Gets or sets the name of the attack detected by this signature.

```
public string AttackName { get; set; }
```

Property Value

string

ContentPattern

Gets or sets content patterns used for payload inspection.

```
public string ContentPattern { get; set; }
```

Property Value

string

Created_at

Gets or sets the timestamp when the signature was created.

```
public DateTime Created_at { get; set; }
```

Property Value

DateTime

DestIp

Gets or sets the destination IP address.

```
public string DestIp { get; set; }
```

Property Value

string

DestPort

Gets or sets the destination port number.

```
public string DestPort { get; set; }
```

Property Value

string

Direction

Gets or sets the traffic direction (inbound, outbound, etc.).

```
public string Direction { get; set; }
```

Property Value

string

Engine

Gets or sets the detection engine that uses this signature.

```
public string Engine { get; set; }
```

Property Value

string

Flow

Gets or sets the flow characteristics of the network traffic.

```
public string Flow { get; set; }
```

Property Value

string

Http

Gets or sets HTTP-specific detection patterns.

```
public string Http { get; set; }
```

Property Value

string

Protocol

Gets or sets the network protocol (TCP, UDP, etc.).

```
public string Protocol { get; set; }
```

Property Value

string

Rev

Gets or sets the revision number of the signature.

```
public int? Rev { get; set; }
```

Property Value

int?

RuleText

Gets or sets the rule text or detection pattern.

```
public string RuleText { get; set; }
```

Property Value

string

Severity

Gets or sets the severity level of the detected attack.

```
public string Severity { get; set; }
```

Property Value

string

Sid

Gets or sets the signature ID (SID).

```
public double Sid { get; set; }
```

Property Value

double

SignatureId

Gets or sets the unique identifier for the signature.

```
public int SignatureId { get; set; }
```

Property Value

int

SrcIp

Gets or sets the source IP address.

```
public string SrcIp { get; set; }
```

Property Value

string

SrcPort

Gets or sets the source port number.

```
public string SrcPort { get; set; }
```

Property Value

string

Tls

Gets or sets TLS/SSL-related detection patterns.

```
public string Tls { get; set; }
```

Property Value

string

Methods

Clone()

Creates a new object that is a copy of the current instance.

```
public Signatures Clone()
```

Returns

Signatures

A new Signatures object that is a copy of this instance.

Class TftpLog

Namespace: IDSApp.Entity

Assembly: IDSApp.dll

Represents a log entry for TFTP (Trivial File Transfer Protocol) activity. Contains information about the operation, file transferred, transfer size, source and destination IPs, timestamp, session, and status.

```
public class TftpLog
```

Inheritance

object ← TftpLog

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

TftpLog()

Initializes a new instance of the TftpLog class.

```
public TftpLog()
```

TftpLog(TftpLog)

Initializes a new instance of the TftpLog class by copying another instance.

```
public TftpLog(TftpLog t)
```

Parameters

`t` TftpLog

The TftpLog instance to copy.

TftpLog(int, int, string, string, int, string, string, DateTime, string, string)

Initializes a new instance of the TftpLog class with specified values.

```
public TftpLog(int tftpLogId, int logId, string operation, string filename, int
transferSize, string sourceIp, string destinationIp, DateTime timestamp, string sessionId,
string status)
```

Parameters

`tftpLogId` int

`logId` int

`operation` string

`filename` string

`transferSize` int

`sourceIp` string

`destinationIp` string

`timestamp` DateTime

`sessionId` string

`status` string

Properties

DestinationIp

Gets or sets the destination IP address of the TFTP transfer.

```
public string DestinationIp { get; set; }
```

Property Value

string

Filename

Gets or sets the name of the file involved in the TFTP operation.

```
public string Filename { get; set; }
```

Property Value

string

LogId

Gets or sets the associated general log entry ID.

```
public int LogId { get; set; }
```

Property Value

int

Operation

Gets or sets the TFTP operation performed (e.g., read, write).

```
public string Operation { get; set; }
```

Property Value

string

SessionId

Gets or sets the session ID for the TFTP transfer.

```
public string SessionId { get; set; }
```

Property Value

string

SourceIp

Gets or sets the source IP address of the TFTP transfer.

```
public string SourceIp { get; set; }
```

Property Value

string

Status

Gets or sets the status of the TFTP operation (e.g., success, error).

```
public string Status { get; set; }
```

Property Value

string

TftpLogId

Gets or sets the unique identifier for this TFTP log entry.

```
public int TftpLogId { get; set; }
```

Property Value

int

Timestamp

Gets or sets the timestamp when the TFTP log was recorded.

```
public DateTime Timestamp { get; set; }
```

Property Value

DateTime

TransferSize

Gets or sets the size of the file transferred in bytes.

```
public int TransferSize { get; set; }
```

Property Value

int

Methods

Clone()

Creates a deep copy of the current TftpLog instance.

```
public TftpLog Clone()
```

Returns

TftpLog

A new TftpLog object identical to the current instance.

Namespace IDSApp.Helper

Classes

BoyerMooreSearch

Provides efficient search methods for locating byte or character sequences within larger datasets, based on the Boyer-Moore algorithm.

BytePatternMatcher

Performs pattern matching on raw byte payloads using a simple byte comparison algorithm. Used for non-text protocols or binary signatures.

DnsConditions

EnhancedPcreValidator

EnhancedRuleParser

Advanced rule parsing and analysis system for Intrusion Detection System (IDS)

Main Responsibilities:

- Parse and extract conditions from security rules in multiple protocols
- Analyze network traffic against parsed rule conditions
- Validate rule syntax and semantics
- Extract metadata and content patterns from rules

Supported Protocols:

- HTTP/HTTPS traffic analysis
- TLS/SSL certificate inspection
- DNS query and response parsing
- ICMP packet analysis
- Network flow and connection tracking

Features:

- High-performance regex-based parsing
- Comprehensive protocol support
- Rule validation and error checking
- TLS fingerprinting and certificate analysis
- Content pattern extraction and normalization

EnhancedSignatureRuleEngine

Enhanced rule engine for signature-based threat detection with support for multiple rule formats, performance optimizations, and advanced pattern matching. Provides fast packet inspection using bloom filters and rule prioritization.

Features:

- PCRE pattern matching with validation and recovery
- Protocol-specific rule optimization and indexing
- Rule conflict resolution and prioritization
- Performance monitoring and statistical analysis
- Fast-path optimization using bloom filters
- Multi-threaded rule checking with load balancing
- Comprehensive error handling and validation

EnhancedSignatureRuleEngine.PcreContentMatcher

PCRE-based content matcher implementation for regex pattern matching

FlowConditions

HttpConditions

IPNetwork

Represents an IP network range in CIDR notation Supports both IPv4 and IPv6 network containment checks

IcmpConditions

NetworkConditions

OptimizedLogger

Provides optimized logging functionality for the IDS system with configurable log levels to reduce console clutter while maintaining essential statistics and critical information.

ParsedRuleConditions

PcreContentMatcher

Performs pattern matching using PCRE (Perl-Compatible Regular Expressions). Useful for detecting text-based signatures in HTTP, SMTP, or DNS payloads.

ProtocolPerformance

Represents performance metrics for a specific network protocol. Tracks packet counts, total bytes, and last update timestamp.

RuleBloomFilter

Bloom Filter implementation specialized for security rules Uses multiple hash functions to represent rules in compact bit array

RuleFastPathOptimizer

High-performance rule checking optimization system using Bloom Filters technology

Main Responsibilities:

- Create and manage Bloom Filters for fast rule filtering
- Reduce number of rules needing detailed inspection
- Provide automatic performance statistics and optimization
- Manage cache for inspection results

How It Works:

1. Group rules by protocol and common ports
2. Build Bloom Filters for each rule group
3. Use filters for quick match possibility checking
4. Skip detailed inspection when filters confirm no possible match

Features:

- Significant rule inspection performance improvement
- Automatic adaptation to rule changes
- Continuous system accuracy monitoring
- Automatic rebuild when efficiency decreases

RuleGroup

Represents a group of security rules with common characteristics Used for organized rule management and optimization

RuleMetadata

RuleOptimizationStats

Statistics container for rule optimization performance metrics Tracks filter count, rule count, memory usage, and accuracy estimates

RulePerformance

Represents performance statistics for a specific IDS rule. Tracks total checks, matches, average time, and efficiency.

RulePriorityManager

Manages rule priorities and detects conflicts among IDS rules. Ensures that when multiple rules match the same packet, the most specific rule is applied first.

RuleSpecificity

Represents the specificity of a rule with a score and contributing fields.

RuleStatistics

Provides real-time collection and analysis of rule and protocol performance statistics. Periodically saves data to the database for long-term analytics.

StatisticsSnapshot

Provides a summarized snapshot of the current IDS performance metrics. Useful for dashboards and monitoring.

TlsConditions

Interfaces

IContentMatcher

Defines an interface for pattern matchers that operate on raw packet payloads. Used for detecting signatures or malicious content across various protocols.

Enums

OptimizedLogger.LogLevel

Class BoyerMooreSearch

Namespace: IDSApp.Helper

Assembly: IDSApp.dll

Provides efficient search methods for locating byte or character sequences within larger datasets, based on the Boyer-Moore algorithm.

```
public static class BoyerMooreSearch
```

Inheritance

object ← BoyerMooreSearch

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

Contains(ReadOnlySpan<byte>, ReadOnlySpan<byte>)

Searches for a byte sequence within another byte sequence.

```
public static bool Contains(ReadOnlySpan<byte> haystack, ReadOnlySpan<byte> needle)
```

Parameters

`haystack` ReadOnlySpan<byte>

The main data buffer to search in.

`needle` ReadOnlySpan<byte>

The byte pattern to look for.

Returns

bool

`true` if the pattern exists in the buffer; otherwise, `false`.

Contains(ReadOnlySpan<char>, ReadOnlySpan<char>)

Searches for a character sequence within another string (case-insensitive).

```
public static bool Contains(ReadOnlySpan<char> haystack, ReadOnlySpan<char> needle)
```

Parameters

`haystack` ReadOnlySpan<char>

The text content to search in.

`needle` ReadOnlySpan<char>

The substring to look for.

Returns

bool

`true` if the substring exists within the text; otherwise, `false`.

Class BytePatternMatcher

Namespace: IDSApp.Helper

Assembly: IDSApp.dll

Performs pattern matching on raw byte payloads using a simple byte comparison algorithm. Used for non-text protocols or binary signatures.

```
public class BytePatternMatcher : IContentMatcher
```

Inheritance

object ← BytePatternMatcher

Implements

IContentMatcher

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

BytePatternMatcher(byte[])

Initializes a new instance of the BytePatternMatcher class.

```
public BytePatternMatcher(byte[] pattern)
```

Parameters

`pattern` byte[]

The byte pattern to search for within payloads.

Methods

Match(byte[], string)

Checks whether the payload contains the specified byte pattern.

```
public bool Match(byte[] payload, string protocol)
```

Parameters

`payload` byte[]

The packet payload as a byte array.

`protocol` string

The name of the protocol being analyzed.

Returns

bool

`true` if the pattern is found; otherwise, `false`.

Class DnsConditions

Namespace: IDSApp.Helper

Assembly: IDSApp.dll

```
public class DnsConditions
```

Inheritance

object ← DnsConditions

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Qname

```
public string Qname { get; set; }
```

Property Value

string

Qtype

```
public string Qtype { get; set; }
```

Property Value

string

RCode

```
public string RCode { get; set; }
```

Property Value

string

RData

```
public string RData { get; set; }
```

Property Value

string

RRType

```
public string RRType { get; set; }
```

Property Value

string

Class EnhancedPcreValidator

Namespace: IDSApp.Helper

Assembly: IDSApp.dll

```
public class EnhancedPcreValidator
```

Inheritance

object ← EnhancedPcreValidator

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

CleanPcrePattern(string)

```
public static string CleanPcrePattern(string pattern)
```

Parameters

`pattern` string

Returns

string

IsPatternMeaningful(string)

```
public static bool IsPatternMeaningful(string pattern)
```

Parameters

`pattern` string

Returns

bool

IsPatternTooShort(string)

```
public static bool IsPatternTooShort(string pattern)
```

Parameters

`pattern` string

Returns

bool

SmartPatternRecovery(string)

```
public static string SmartPatternRecovery(string pattern)
```

Parameters

`pattern` string

Returns

string

ValidateAndFixPcrePattern(string)

```
public static (bool isValid, string cleanedPattern, string recoveryAction)
ValidateAndFixPcrePattern(string pattern)
```

Parameters

`pattern` string

Returns

(bool isValid, string cleanedPattern, string recoveryAction)

Class EnhancedRuleParser

Namespace: IDSApp.Helper

Assembly: IDSApp.dll

Advanced rule parsing and analysis system for Intrusion Detection System (IDS)

Main Responsibilities:

- Parse and extract conditions from security rules in multiple protocols
- Analyze network traffic against parsed rule conditions
- Validate rule syntax and semantics
- Extract metadata and content patterns from rules

Supported Protocols:

- HTTP/HTTPS traffic analysis
- TLS/SSL certificate inspection
- DNS query and response parsing
- ICMP packet analysis
- Network flow and connection tracking

Features:

- High-performance regex-based parsing
- Comprehensive protocol support
- Rule validation and error checking
- TLS fingerprinting and certificate analysis
- Content pattern extraction and normalization

```
public class EnhancedRuleParser
```

Inheritance

object ← EnhancedRuleParser

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

EnhancedRuleParser()

```
public EnhancedRuleParser()
```

Methods

ParseConditions(Signatures)

Parse all conditions from a security rule into structured format Extracts HTTP, TLS, DNS, ICMP, network, and metadata conditions

```
public ParsedRuleConditions ParseConditions(Signatures rule)
```

Parameters

`rule` Signatures

The security rule to parse

Returns

ParsedRuleConditions

Structured rule conditions

ParseTlsPacket(byte[])

Parse TLS handshake packet and extract security information

Returns:

- SNI (Server Name Indication)
- TLS version
- Cipher suite
- Certificate fingerprint
- JA3 fingerprint (for client hello)
- Certificate subject and issuer
- Certificate validity dates

```
public (string sni, string version, string cipherSuite, string certFingerprint, string
ja3Fingerprint, string subject, string issuer, DateTime? notBefore, DateTime? notAfter)
ParseTlsPacket(byte[] payload)
```

Parameters

`payload` byte[]

Returns

(string sni, string version, string cipherSuite, string certFingerprint, string ja3Fingerprint, string subject, string issuer, DateTime? notBefore, DateTime? notAfter)

ValidateRule(Signatures, out List<string>)

Validate rule syntax and semantics Checks for valid patterns, IPs, ports, and protocol-specific conditions

```
public bool ValidateRule(Signatures rule, out List<string> validationErrors)
```

Parameters

`rule` Signatures

`validationErrors` List<string>

Returns

bool

Class EnhancedSignatureRuleEngine

Namespace: IDSApp.Helper

Assembly: IDSApp.dll

Enhanced rule engine for signature-based threat detection with support for multiple rule formats, performance optimizations, and advanced pattern matching. Provides fast packet inspection using bloom filters and rule prioritization.

Features:

- PCRE pattern matching with validation and recovery
- Protocol-specific rule optimization and indexing
- Rule conflict resolution and prioritization
- Performance monitoring and statistical analysis
- Fast-path optimization using bloom filters
- Multi-threaded rule checking with load balancing
- Comprehensive error handling and validation

```
public class EnhancedSignatureRuleEngine
```

Inheritance

object ← EnhancedSignatureRuleEngine

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

EnhancedSignatureRuleEngine()

Initializes a new enhanced signature rule engine with default configuration

```
public EnhancedSignatureRuleEngine()
```

Methods

CheckPacket(IPPacket, byte[], string, string, int, int, string)

Inspects a network packet against loaded detection rules to identify malicious activity. Uses multiple optimization techniques to ensure high-performance inspection even with large rule sets.

```
public List<Signatures> CheckPacket(IPPacket ipPacket, byte[] payload, string srcIp, string
dstIp, int srcPort, int dstPort, string protocolName)
```

Parameters

`ipPacket` IPPacket

IP packet to inspect for threats

`payload` byte[]

Packet payload data for content inspection

`srcIp` string

Source IP address for contextual analysis

`dstIp` string

Destination IP address for contextual analysis

`srcPort` int

Source port number for protocol analysis

`dstPort` int

Destination port number for protocol analysis

`protocolName` string

Network protocol name for rule filtering

Returns

List<Signatures>

List of matched signature rules describing detected threats

Remarks

Inspection Pipeline:

1. Pre-filtering using bloom filters and whitelists
2. Protocol and port-based rule selection
3. Content pattern matching with PCRE
4. Rule prioritization and conflict resolution
5. Statistical tracking and performance monitoring
6. Result deduplication and optimization

ClearRules()

Clears all loaded rules and resets the engine to initial state

```
public void ClearRules()
```

GetActiveRulesCount()

Gets the number of currently active rules

```
public int GetActiveRulesCount()
```

Returns

int

GetRuleById(int)

Retrieves a rule by its signature ID for detailed analysis

```
public Signatures GetRuleById(int signatureId)
```

Parameters

`signatureId` int

Returns

Signatures

IsInternalTraffic(string, string)

Determines if traffic between two IP addresses is internal network traffic

```
public bool IsInternalTraffic(string srcIp, string dstIp)
```

Parameters

`srcIp` string

`dstIp` string

Returns

bool

IsSuspiciousEmptyPacket(IPPacket)

Identifies suspicious empty packets that may indicate protocol anomalies

```
public bool IsSuspiciousEmptyPacket(IPPacket ipPacket)
```

Parameters

`ipPacket` IPPacket

Returns

bool

IsSuspiciousSmallPacket(IPPacket)

Identifies suspicious small packets that may indicate scanning or attacks

```
public bool IsSuspiciousSmallPacket(IPPacket ipPacket)
```

Parameters

`ipPacket` IPPacket

Returns

bool

LoadRules()

Loads and initializes detection rules from the database, building optimized indexes for fast packet inspection. Validates rule syntax and organizes rules by protocol, port, and content patterns for efficient matching.

```
public void LoadRules()
```

Remarks

Rule Loading Process:

1. Load rules from database and backup sources with error handling
2. Validate PCRE patterns and apply automatic fixes for common issues
3. Build protocol-based, port-based, and content-based indexes
4. Initialize bloom filters for fast-path optimization
5. Analyze rule conflicts and establish priority hierarchy
6. Generate comprehensive rule statistics and validation reports

LogInternalStats()

Logs internal engine statistics for monitoring and optimization

```
public void LogInternalStats()
```

PrintEnhancedPerformanceStats()

Enhanced performance statistics with additional metrics and analysis

```
public void PrintEnhancedPerformanceStats()
```

PrintPerformanceStats()

Prints comprehensive performance statistics including rule matching rates, processing times, and optimization effectiveness for system monitoring

```
public void PrintPerformanceStats()
```

PrintRuleStatistics()

Prints detailed rule statistics including distribution by protocol, port, and content patterns for system analysis and optimization

```
public void PrintRuleStatistics()
```

Class EnhancedSignatureRuleEngine.PcreContentMatcher

Namespace: IDSApp.Helper

Assembly: IDSApp.dll

PCRE-based content matcher implementation for regex pattern matching

```
public class EnhancedSignatureRuleEngine.PcreContentMatcher : IContentMatcher
```

Inheritance

object ← EnhancedSignatureRuleEngine.PcreContentMatcher

Implements

IContentMatcher

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

PcreContentMatcher(PcreRegex)

Initializes a new PCRE content matcher with compiled regex pattern

```
public PcreContentMatcher(PcreRegex regex)
```

Parameters

`regex` PcreRegex

Compiled PCRE regex pattern for efficient matching

Methods

IsMatch(byte[])

Matches payload against the PCRE pattern using UTF-8 encoding

```
public bool IsMatch(byte[] payload)
```

Parameters

`payload` byte[]

Binary payload to inspect

Returns

bool

True if pattern matches any part of the payload

Match(byte[], string)

Interface implementation for protocol-aware content matching

```
public bool Match(byte[] payload, string protocol)
```

Parameters

`payload` byte[]

`protocol` string

Returns

bool

Class FlowConditions

Namespace: IDSApp.Helper

Assembly: IDSApp.dll

```
public class FlowConditions
```

Inheritance

object ← FlowConditions

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Established

```
public bool Established { get; set; }
```

Property Value

bool

FromClient

```
public bool FromClient { get; set; }
```

Property Value

bool

FromServer

```
public bool FromServer { get; set; }
```

Property Value

bool

Stateless

```
public bool Stateless { get; set; }
```

Property Value

bool

ToClient

```
public bool ToClient { get; set; }
```

Property Value

bool

ToServer

```
public bool ToServer { get; set; }
```

Property Value

bool

Class HttpConditions

Namespace: IDSApp.Helper

Assembly: IDSApp.dll

```
public class HttpConditions
```

Inheritance

object ← HttpConditions

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

ContentPatterns

```
public List<string> ContentPatterns { get; set; }
```

Property Value

List<string>

Cookie

```
public string Cookie { get; set; }
```

Property Value

string

HeaderName

```
public string HeaderName { get; set; }
```

Property Value

string

Method

```
public string Method { get; set; }
```

Property Value

string

UriPattern

```
public string UriPattern { get; set; }
```

Property Value

string

UserAgent

```
public string UserAgent { get; set; }
```

Property Value

string

Interface IContentMatcher

Namespace: IDSApp.Helper

Assembly: IDSApp.dll

Defines an interface for pattern matchers that operate on raw packet payloads. Used for detecting signatures or malicious content across various protocols.

```
public interface IContentMatcher
```

Methods

Match(byte[], string)

Determines whether the specified payload matches a predefined pattern.

```
bool Match(byte[] payload, string protocol)
```

Parameters

`payload` byte[]

The byte array representing packet data.

`protocol` string

The name of the protocol being analyzed (e.g., HTTP, DNS).

Returns

bool

`true` if the pattern is found within the payload; otherwise, `false`.

Class IPNetwork

Namespace: IDSApp.Helper

Assembly: IDSApp.dll

Represents an IP network range in CIDR notation Supports both IPv4 and IPv6 network containment checks

```
public class IPNetwork
```

Inheritance

object ← IPNetwork

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Properties

NetworkAddress

```
public IPAddress NetworkAddress { get; set; }
```

Property Value

IPAddress

PrefixLength

```
public int PrefixLength { get; set; }
```

Property Value

int

Methods

Contains(IPAddress)

Check if IP address is contained within this network

```
public bool Contains(IPAddress address)
```

Parameters

`address` IPAddress

Returns

bool

ToString()

Returns a string that represents the current object.

```
public override string ToString()
```

Returns

string

A string that represents the current object.

TryParse(string, out IPNetwork)

Parse CIDR notation string into IPNetwork object Supports formats: "192.168.1.0/24", "2001:db8::/32"

```
public static bool TryParse(string cidr, out IPNetwork network)
```

Parameters

`cidr` string

`network` IPNetwork

Returns

bool

Class IcmpConditions

Namespace: IDSApp.Helper

Assembly: IDSApp.dll

```
public class IcmpConditions
```

Inheritance

object ← IcmpConditions

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Code

```
public string Code { get; set; }
```

Property Value

string

Type

```
public string Type { get; set; }
```

Property Value

string

Class NetworkConditions

Namespace: IDSApp.Helper

Assembly: IDSApp.dll

```
public class NetworkConditions
```

Inheritance

object ← NetworkConditions

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

DestinationIPs

```
public List<string> DestinationIPs { get; set; }
```

Property Value

List<string>

DestinationPorts

```
public List<string> DestinationPorts { get; set; }
```

Property Value

List<string>

SourceIPs

```
public List<string> SourceIPs { get; set; }
```

Property Value

List <string>

SourcePorts

```
public List<string> SourcePorts { get; set; }
```

Property Value

List <string>

Class OptimizedLogger

Namespace: IDSApp.Helper

Assembly: IDSApp.dll

Provides optimized logging functionality for the IDS system with configurable log levels to reduce console clutter while maintaining essential statistics and critical information.

```
public static class OptimizedLogger
```

Inheritance

object ← OptimizedLogger

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

CurrentLogLevel

```
public static OptimizedLogger.LogLevel CurrentLogLevel { get; set; }
```

Property Value

OptimizedLogger.LogLevel

Methods

Dispose()

Disposes the logger and flushes any remaining messages

```
public static void Dispose()
```

GetPacketCount()

Gets the current packet count for statistical purposes

```
public static long GetPacketCount()
```

Returns

long

IncrementPacketCounter()

Increments the global packet counter for statistics

```
public static void IncrementPacketCounter()
```

LogDebug(string)

Logs debug information (disabled by default to reduce console output)

```
public static void LogDebug(string message)
```

Parameters

`message` string

LogError(string)

Logs error messages (always enabled for critical issues)

```
public static void LogError(string message)
```

Parameters

`message` string

LogImportant(string)

Logs important system events (startup, shutdown, alerts)

```
public static void LogImportant(string message)
```

Parameters

`message` string

LogNetwork(string)

Logs network-related information (disabled by default)

```
public static void LogNetwork(string message)
```

Parameters

`message` string

LogPacketStats(string)

Logs packet statistics at configured intervals

```
public static void LogPacketStats(string message)
```

Parameters

`message` string

LogPerformance(string)

Logs performance metrics at configured intervals

```
public static void LogPerformance(string message)
```

Parameters

`message` string

LogQueue(string)

Logs queue operations (disabled by default)

```
public static void LogQueue(string message)
```

Parameters

`message` string

LogRule(string)

Logs rule processing information (reduced frequency)

```
public static void LogRule(string message)
```

Parameters

`message` string

LogRuleMatch(string)

Logs rule matching events with throttling to prevent spam

```
public static void LogRuleMatch(string message)
```

Parameters

`message` string

Enum OptimizedLogger.LogLevel

Namespace: IDSApp.Helper

Assembly: IDSApp.dll

```
public enum OptimizedLogger.LogLevel
```

Fields

DEBUG = 4

ERROR = 0

IMPORTANT = 1

PERFORMANCE = 3

STATS = 2

Class ParsedRuleConditions

Namespace: IDSApp.Helper

Assembly: IDSApp.dll

```
public class ParsedRuleConditions
```

Inheritance

object ← ParsedRuleConditions

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

ContentPatterns

```
public List<string> ContentPatterns { get; set; }
```

Property Value

List<string>

DnsConditions

```
public DnsConditions DnsConditions { get; set; }
```

Property Value

DnsConditions

FlowConditions

```
public FlowConditions FlowConditions { get; set; }
```

Property Value

FlowConditions

HttpConditions

```
public HttpConditions HttpConditions { get; set; }
```

Property Value

HttpConditions

IcmpConditions

```
public IcmpConditions IcmpConditions { get; set; }
```

Property Value

IcmpConditions

Metadata

```
public RuleMetadata Metadata { get; set; }
```

Property Value

RuleMetadata

NetworkConditions

```
public NetworkConditions NetworkConditions { get; set; }
```

Property Value

NetworkConditions

TlsConditions

```
public TlsConditions TlsConditions { get; set; }
```

Property Value

TlsConditions

Class PcreContentMatcher

Namespace: IDSApp.Helper

Assembly: IDSApp.dll

Performs pattern matching using PCRE (Perl-Compatible Regular Expressions). Useful for detecting text-based signatures in HTTP, SMTP, or DNS payloads.

```
public class PcreContentMatcher : IContentMatcher
```

Inheritance

object ← PcreContentMatcher

Implements

IContentMatcher

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

PcreContentMatcher(Regex)

Initializes a new instance of the PcreContentMatcher class.

```
public PcreContentMatcher(Regex regex)
```

Parameters

`regex` Regex

The regular expression pattern used for matching text payloads.

Methods

Match(byte[], string)

Checks whether the text content of the payload matches the configured regular expression.

```
public bool Match(byte[] payload, string protocol)
```

Parameters

`payload` byte[]

The raw packet payload.

`protocol` string

The protocol being analyzed (for contextual matching).

Returns

bool

`true` if the regex matches the text content; otherwise, `false`.

Class ProtocolPerformance

Namespace: IDSApp.Helper

Assembly: IDSApp.dll

Represents performance metrics for a specific network protocol. Tracks packet counts, total bytes, and last update timestamp.

```
public class ProtocolPerformance
```

Inheritance

object ← ProtocolPerformance

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

AdditionalInfo

```
public string AdditionalInfo { get; set; }
```

Property Value

string

LastUpdated

```
public DateTime? LastUpdated { get; set; }
```

Property Value

DateTime?

PacketCount

```
public long PacketCount { get; set; }
```

Property Value

long

Protocol

```
public string Protocol { get; set; }
```

Property Value

string

TotalBytes

```
public long TotalBytes { get; set; }
```

Property Value

long

Class RuleBloomFilter

Namespace: IDSApp.Helper

Assembly: IDSApp.dll

Bloom Filter implementation specialized for security rules Uses multiple hash functions to represent rules in compact bit array

```
public class RuleBloomFilter
```

Inheritance

object ← RuleBloomFilter

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

RuleBloomFilter(int)

```
public RuleBloomFilter(int filterSize = 16384)
```

Parameters

`filterSize` int

Methods

AddRule(Signatures)

Add a security rule to the Bloom Filter

```
public void AddRule(Signatures rule)
```

Parameters

`rule` Signatures

GetFalsePositiveProbability()

Calculate estimated false positive probability percentage

```
public double GetFalsePositiveProbability()
```

Returns

double

GetMemoryUsage()

Get memory usage of this Bloom Filter in bytes

```
public long GetMemoryUsage()
```

Returns

long

MightMatch(string, int, int, int, string)

Check if network parameters might match any rule in this filter Returns false for definite non-matches, true for possible matches

```
public bool MightMatch(string protocol, int port, int payloadLength, int srcPort = 0, string
flowDirection = "")
```

Parameters

`protocol` string

`port` int

`payloadLength` int

`srcPort` int

`flowDirection` string

Returns

bool

Class RuleFastPathOptimizer

Namespace: IDSApp.Helper

Assembly: IDSApp.dll

High-performance rule checking optimization system using Bloom Filters technology

Main Responsibilities:

- Create and manage Bloom Filters for fast rule filtering
- Reduce number of rules needing detailed inspection
- Provide automatic performance statistics and optimization
- Manage cache for inspection results

How It Works:

1. Group rules by protocol and common ports
2. Build Bloom Filters for each rule group
3. Use filters for quick match possibility checking
4. Skip detailed inspection when filters confirm no possible match

Features:

- Significant rule inspection performance improvement
- Automatic adaptation to rule changes
- Continuous system accuracy monitoring
- Automatic rebuild when efficiency decreases

```
public class RuleFastPathOptimizer
```

Inheritance

object ← RuleFastPathOptimizer

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

RuleFastPathOptimizer(bool, int, double)

Initialize Rule Fast Path Optimizer

```
public RuleFastPathOptimizer(bool debug = false, int rebuildIntervalMinutes = 10, double
fppThresholdPercent = 5)
```

Parameters

`debug` bool

Enable debug mode for detailed logging

`rebuildIntervalMinutes` int

Auto-rebuild interval in minutes

`fppThresholdPercent` double

False positive rate threshold before rebuild

Methods

BuildBloomFilters(List<Signatures>)

Build Bloom Filters from provided rules Groups rules by protocol and creates filters for common ports

```
public void BuildBloomFilters(List<Signatures> rules)
```

Parameters

`rules` List<Signatures>

List of security rules to optimize

GetOptimizationStats()

Get current optimization statistics

```
public RuleOptimizationStats GetOptimizationStats()
```

Returns

RuleOptimizationStats

PrintOptimizationInfo()

Print optimization information to console

```
public void PrintOptimizationInfo()
```

ShouldCheckRules(string, int, int, string, int, string)

Determine if rules should be checked for given network parameters Uses Bloom Filters for fast path optimization

⚠ CURRENTLY DISABLED: Returns true always to ensure all rules are checked until fast path system is fully validated

```
public bool ShouldCheckRules(string protocol, int dstPort, int payloadLength, string
ruleGroup, int srcPort = 0, string flowDirection = "")
```

Parameters

`protocol` string

`dstPort` int

`payloadLength` int

`ruleGroup` string

`srcPort` int

`flowDirection` string

Returns

bool

Class RuleGroup

Namespace: IDSApp.Helper

Assembly: IDSApp.dll

Represents a group of security rules with common characteristics Used for organized rule management and optimization

```
public class RuleGroup
```

Inheritance

object ← RuleGroup

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Protocol

```
public string Protocol { get; set; }
```

Property Value

string

Rules

```
public List<Signatures> Rules { get; set; }
```

Property Value

List<Signatures>

Class RuleMetadata

Namespace: IDSApp.Helper

Assembly: IDSApp.dll

```
public class RuleMetadata
```

Inheritance

object ← RuleMetadata

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

ClassType

```
public string ClassType { get; set; }
```

Property Value

string

Msg

```
public string Msg { get; set; }
```

Property Value

string

Priority

```
public int? Priority { get; set; }
```

Property Value

int?

Rev

```
public int? Rev { get; set; }
```

Property Value

int?

Severity

```
public string Severity { get; set; }
```

Property Value

string

Sid

```
public int? Sid { get; set; }
```

Property Value

int?

Class RuleOptimizationStats

Namespace: IDSApp.Helper

Assembly: IDSApp.dll

Statistics container for rule optimization performance metrics Tracks filter count, rule count, memory usage, and accuracy estimates

```
public class RuleOptimizationStats
```

Inheritance

object ← RuleOptimizationStats

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

FalsePositiveEstimate

```
public double FalsePositiveEstimate { get; set; }
```

Property Value

double

MemoryUsage

```
public long MemoryUsage { get; set; }
```

Property Value

long

TotalFilters

```
public int TotalFilters { get; set; }
```

Property Value

int

TotalRules

```
public int TotalRules { get; set; }
```

Property Value

int

Class RulePerformance

Namespace: IDSApp.Helper

Assembly: IDSApp.dll

Represents performance statistics for a specific IDS rule. Tracks total checks, matches, average time, and efficiency.

```
public class RulePerformance
```

Inheritance

object ← RulePerformance

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

AttackName

```
public string AttackName { get; set; }
```

Property Value

string

AvgProcessingTimeMs

```
public double AvgProcessingTimeMs { get; set; }
```

Property Value

double

EfficiencyScore

```
public double EfficiencyScore { get; set; }
```

Property Value

double

LastChecked

```
public DateTime? LastChecked { get; set; }
```

Property Value

DateTime?

LastMatch

```
public DateTime? LastMatch { get; set; }
```

Property Value

DateTime?

MatchRate

```
public double MatchRate { get; set; }
```

Property Value

double

Matches

```
public long Matches { get; set; }
```

Property Value

long

SignatureId

```
public int SignatureId { get; set; }
```

Property Value

int

TotalChecks

```
public long TotalChecks { get; set; }
```

Property Value

long

TotalProcessingTimeMs

```
public long TotalProcessingTimeMs { get; set; }
```

Property Value

long

Class RulePriorityManager

Namespace: IDSApp.Helper

Assembly: IDSApp.dll

Manages rule priorities and detects conflicts among IDS rules. Ensures that when multiple rules match the same packet, the most specific rule is applied first.

```
public class RulePriorityManager
```

Inheritance

object ← RulePriorityManager

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

AnalyzeRuleConflicts(List<Signatures>)

Analyzes a list of IDS rules to calculate their specificity and detect conflicts. Conflicting rules are grouped by similar characteristics (protocol, port, content pattern).

```
public void AnalyzeRuleConflicts(List<Signatures> rules)
```

Parameters

`rules` List<Signatures>

List of IDS rules (signatures) to analyze.

ApplyPriorities(List<Signatures>)

Filters and prioritizes matching rules based on calculated priorities and conflicts. Only the highest priority rule in each conflict group is kept.

```
public List<Signatures> ApplyPriorities(List<Signatures> matches)
```

Parameters

`matches` List<Signatures>

List of matching rules for a packet.

Returns

List<Signatures>

Prioritized and distinct list of rules.

PrintRuleAnalysis()

Prints the calculated specificity scores and fields for all rules.

```
public void PrintRuleAnalysis()
```

Class RuleSpecificity

Namespace: IDSApp.Helper

Assembly: IDSApp.dll

Represents the specificity of a rule with a score and contributing fields.

```
public class RuleSpecificity
```

Inheritance

object ← RuleSpecificity

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Score

```
public int Score { get; set; }
```

Property Value

int

SpecificFields

```
public List<string> SpecificFields { get; set; }
```

Property Value

List<string>

Class RuleStatistics

Namespace: IDSApp.Helper

Assembly: IDSApp.dll

Provides real-time collection and analysis of rule and protocol performance statistics. Periodically saves data to the database for long-term analytics.

```
public class RuleStatistics
```

Inheritance

object ← RuleStatistics

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

RuleStatistics()

Initializes a new instance of the RuleStatistics class. Starts a background timer to automatically persist statistics every 60 seconds.

```
public RuleStatistics()
```

Methods

GetCurrentSnapshot()

Generates a quick snapshot summary of current statistics for external monitoring or dashboards.

```
public StatisticsSnapshot GetCurrentSnapshot()
```

Returns

StatisticsSnapshot

A StatisticsSnapshot object containing key metrics.

PrintPerformanceReport()

Prints a detailed performance report for both rule and protocol statistics to the log output.

```
public void PrintPerformanceReport()
```

RecordProtocolPacket(string)

Records a packet occurrence for the specified protocol (legacy overload).

```
public void RecordProtocolPacket(string protocol)
```

Parameters

`protocol` string

The protocol name (e.g., TCP, UDP, DNS).

RecordProtocolPacket(string, int)

Records a packet occurrence and its size for a specific protocol.

```
public void RecordProtocolPacket(string protocol, int packetSize)
```

Parameters

`protocol` string

The protocol name.

`packetSize` int

The size of the captured packet in bytes.

RecordProtocolPacket(string, int, string)

Records protocol activity with optional additional information. Updates packet count, total bytes, and logs every 100 packets.

```
public void RecordProtocolPacket(string protocol, int packetSize, string additionalInfo)
```

Parameters

`protocol` string

The protocol name (e.g., TCP, HTTP).

`packetSize` int

Packet size in bytes.

`additionalInfo` string

Optional details such as source/destination.

RecordRuleHit(int, string, bool, long)

Records the performance result of a rule evaluation including matches, checks, and processing time. Calculates efficiency and match rate dynamically.

```
public void RecordRuleHit(int signatureId, string attackName, bool matched,
long processingTimeMs)
```

Parameters

`signatureId` int

Unique ID of the rule signature.

`attackName` string

Name of the associated attack.

`matched` bool

Whether the rule matched the packet.

`processingTimeMs` long

Processing time in milliseconds.

SaveStatsToDatabase()

Saves collected rule and protocol statistics to the database. Automatically invoked by the background timer.

```
public void SaveStatsToDatabase()
```

Class StatisticsSnapshot

Namespace: IDSApp.Helper

Assembly: IDSApp.dll

Provides a summarized snapshot of the current IDS performance metrics. Useful for dashboards and monitoring.

```
public class StatisticsSnapshot
```

Inheritance

object ← StatisticsSnapshot

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

ActiveProtocols

```
public int ActiveProtocols { get; set; }
```

Property Value

int

ActiveRules

```
public int ActiveRules { get; set; }
```

Property Value

int

Timestamp

```
public DateTime Timestamp { get; set; }
```

Property Value

DateTime

TotalPackets

```
public long TotalPackets { get; set; }
```

Property Value

long

TotalRuleChecks

```
public long TotalRuleChecks { get; set; }
```

Property Value

long

TotalRuleMatches

```
public long TotalRuleMatches { get; set; }
```

Property Value

long

Class TlsConditions

Namespace: IDSApp.Helper

Assembly: IDSApp.dll

```
public class TlsConditions
```

Inheritance

object ← TlsConditions

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Cipher

```
public string Cipher { get; set; }
```

Property Value

string

Issuer

```
public string Issuer { get; set; }
```

Property Value

string

Sni

```
public string Sni { get; set; }
```

Property Value

string

Subject

```
public string Subject { get; set; }
```

Property Value

string

Version

```
public string Version { get; set; }
```

Property Value

string

Namespace IDSApp.ProtocolParsing

Classes

BoyerMooreSearch

Efficient binary pattern search using Boyer-Moore algorithm

DatabaseParser

Database protocol parser for Intrusion Detection System

Main Responsibilities:

- Parse and analyze database network traffic for security threats
- Extract SQL queries, commands, and session information
- Detect suspicious database activities and potential attacks
- Generate security logs and alerts for database operations

Supported Database Engines:

- MySQL (port 3306)
- PostgreSQL (port 5432)
- SQL Server (port 1433)
- Generic SQL traffic analysis

Security Detection Capabilities:

- Dangerous SQL commands (DROP, DELETE, ALTER, etc.)
- Suspicious query patterns and keywords
- Long-running queries indicating potential attacks
- Unauthorized database access attempts
- SQL injection pattern detection

Features:

- Protocol-aware parsing based on destination ports
- Payload fingerprinting for attack correlation
- Session tracking and correlation
- Comprehensive logging with performance considerations
- Memory-efficient payload handling

DhcpParser

DHCP protocol parser for Intrusion Detection System

Main Responsibilities:

- Parse and analyze DHCP (Dynamic Host Configuration Protocol) network traffic
- Detect DHCP-based attacks and suspicious activities
- Monitor IP address assignment and lease management
- Identify rogue DHCP servers and IP spoofing attempts

DHCP Protocol Support:

- Message Types: DISCOVER, OFFER, REQUEST, ACK, NAK, RELEASE, DECLINE, INFORM
- Standard Ports: 67 (server), 68 (client)
- Option parsing and validation
- Lease duration monitoring

Security Detection Capabilities:

- Rogue DHCP server detection
- IP address spoofing in DHCP transactions
- Suspicious lease duration values
- Malicious content in DHCP options
- DHCP starvation attacks
- Unauthorized IP assignment attempts

Features:

- Comprehensive DHCP header parsing
- Option analysis with security validation
- Transaction tracking and correlation
- Timestamp validation and correction
- Payload fingerprinting for attack correlation

DnsParser

DNS protocol parser for Intrusion Detection System

Main Responsibilities:

- Parse and analyze DNS (Domain Name System) network traffic
- Extract DNS queries and responses for security analysis
- Detect DNS-based attacks and suspicious domain activities
- Monitor DNS tunneling and data exfiltration attempts

DNS Protocol Support:

- Query Types: A, AAAA, NS, CNAME, PTR, and other common record types
- Standard Port: 53 (UDP/TCP)
- DNS packet structure parsing (header, questions, answers)

- Domain name compression pointer resolution

Security Detection Capabilities:

- DNS tunneling detection through suspicious domain patterns
- Data exfiltration via DNS queries
- Malicious domain name patterns and DGA (Domain Generation Algorithm)
- Suspicious query types and response patterns
- DNS amplification attack indicators

Features:

- Complete DNS packet parsing with compression handling
- Question and answer section analysis
- IPv4 and IPv6 address parsing
- Suspicious domain pattern matching
- Comprehensive logging with forensic data

DnsParser.DnsRecord

Represents a DNS record with name, type, and response data Used for both questions and answers in DNS packets

EnhancedHttpHandler

Advanced HTTP protocol parsing and analysis system for Intrusion Detection System

Main Responsibilities:

- Parse and analyze HTTP/HTTPS traffic for security threats
- Extract and validate HTTP headers, methods, URLs, and body content
- Detect malicious patterns in HTTP requests and responses
- Integrate TLS/SSL certificate analysis for encrypted traffic
- Provide comprehensive threat scoring and pattern matching

Security Detection Capabilities:

- Cross-Site Scripting (XSS) attacks
- SQL Injection (SQLi) attempts
- Local File Inclusion (LFI) and path traversal
- Remote Code Execution (RCE) patterns
- Malicious file uploads and binary content
- Base64-encoded payloads and evasion techniques
- Suspicious HTTP headers and user agents

Features:

- Complete HTTP protocol parsing with validation
- Multi-layer threat detection (text, binary, regex patterns)
- TLS/SSL certificate fingerprinting and analysis
- Content-type aware inspection optimization
- Comprehensive logging with forensic data
- Performance-optimized pattern matching

FtpParseResult

Comprehensive result container for FTP parsing analysis Contains parsed command information and security assessment flags

FtpParser

Advanced FTP protocol parser for Intrusion Detection System

Main Responsibilities:

- Parse and analyze FTP (File Transfer Protocol) commands and transactions
- Extract filenames, paths, and command arguments with security context
- Detect suspicious FTP activities and potential attack patterns
- Normalize paths and identify path traversal attempts
- Flag credential exposure in FTP sessions

Security Detection Capabilities:

- Path traversal attacks using "../" and absolute paths
- Suspicious file extensions indicating malware/scripts
- Base64-encoded payloads in filenames or arguments
- Suspicious keywords (shell, cmd, webshell, etc.)
- Credential exposure in PASS commands
- Long filenames potentially hiding malicious content
- Percent-encoding evasion techniques

Features:

- Comprehensive FTP command parsing with argument extraction
- Path normalization and traversal detection
- File operation classification (RETR, STOR, DELE, etc.)
- Multiple encoding detection and handling
- Suspicious pattern matching with detailed reasoning

HttpParser

Advanced HTTP threat detection engine for Intrusion Detection System

Detection Methods:

- Text pattern matching for common attack signatures
- Regular expression analysis for complex attack patterns
- Binary pattern detection for executable content
- Base64 decoding and analysis for encoded payloads
- URL decoding for obfuscated attack vectors

Performance Features:

- Configurable inspection limits to prevent resource exhaustion
- Content-type filtering to skip non-relevant data
- Efficient Boyer-Moore algorithm for binary pattern matching
- Threat scoring with adjustable thresholds

HttpParser.ThreatResult

Threat analysis result container

HttpParserHelpers

HTTP parser utility functions

ImapParser

IMAP protocol parser for Intrusion Detection System

Main Responsibilities:

- Parse and analyze IMAP (Internet Message Access Protocol) traffic
- Monitor email client-server communications for security threats
- Detect suspicious IMAP commands and authentication attempts
- Analyze email content and attachments for malicious patterns
- Track session activity and command sequences

IMAP Protocol Support:

- Standard IMAP commands (LOGIN, SELECT, FETCH, STORE, APPEND, LOGOUT)
- Email folder operations and message management
- Authentication and session management
- Email content retrieval and analysis

Security Detection Capabilities:

- Suspicious command patterns and sequences
- Authentication brute force attempts
- Malicious payloads in email content
- Sensitive information extraction attempts
- Protocol anomalies and error conditions

Features:

- Comprehensive IMAP command and response parsing
- Configurable suspicious command detection
- Payload fingerprinting for content correlation
- Session tracking and attempt counting
- Real-time threat detection and logging

LdapParser

LDAP protocol parser for Intrusion Detection System

Main Responsibilities:

- Parse and analyze LDAP (Lightweight Directory Access Protocol) traffic
- Extract directory operations, authentication attempts, and search queries
- Detect suspicious LDAP activities and directory service attacks
- Monitor directory access patterns and privilege escalation attempts
- Analyze distinguished names and filter criteria for security threats

LDAP Protocol Support:

- Authentication operations (BindRequest, BindResponse)
- Directory search and query operations (SearchRequest, SearchResultEntry)
- Directory modification operations (Add, Modify, Delete)
- ASN.1 BER encoding parsing for LDAP packets
- Standard ports 389 (LDAP) and 636 (LDAPS)

Security Detection Capabilities:

- Suspicious distinguished name patterns
- Directory traversal and unauthorized access attempts
- LDAP injection and filter manipulation
- Privilege escalation through directory modifications
- Encrypted LDAPS traffic monitoring

Features:

- Complete ASN.1 BER protocol parsing
- Operation type extraction and classification
- Distinguished name analysis and validation
- Result code monitoring for failed operations
- Encrypted session detection and handling

NetbiosParser

NtpParser

Pop3Parser

RdpParseResult
Result container for RDP parsing operations Stores extracted RDP session data, authentication information, and security analysis results Provides utility methods for result validation and formatting

RdpParser

SmbParseResult
Result container for SMB parsing operations Stores extracted SMB command data, file information, and security analysis results Provides formatted output for logging and monitoring systems

SmbParser
Advanced SMB parser supporting SMB1/SMB2/SMB3, extracting command, filename, share, and performing IDS checks for suspicious activity.

SmtpParseResult
Result container for SMTP parsing operations Stores extracted email metadata, security analysis results, and suspicion reasons Provides formatted output for logging and email security monitoring

SmtpParser
Advanced SMTP parser that extracts From, To, Subject, decodes MIME headers, handles multi-line subjects, and performs basic IDS checks for suspicious content and attachments.

SnmpParser

SshParser

TelnetParseResult
Result container for Telnet parsing operations Stores extracted Telnet commands, connection information, and security analysis results Provides utility methods for result validation and formatted output

TelnetParser

TftpParser

TlsParser
Production-ready TLS parser:

- Parses TLS records and handshake (ClientHello, Certificate)
- Extracts SNI, computes a best-effort JA3 fingerprint

- Extracts certificate(s) from Certificate handshake and inspects them (self-signed, expired, SHA256 fingerprint)
- Performs lightweight deep-inspection on ASCII content (configurable)
- Logs results to DB via LogBLL.Insert and saves TLS-ERROR rows also via LogBLL.Insert
- Loads known-malicious JA3s from Ja3Repository (DB-backed)

TlsParser.TlsParseResult

Comprehensive result container for TLS parsing operations Stores extracted TLS handshake data, security analysis results, and threat intelligence matches

Class BoyerMooreSearch

Namespace: IDSApp.ProtocolParsing

Assembly: IDSApp.dll

Efficient binary pattern search using Boyer-Moore algorithm

```
public static class BoyerMooreSearch
```

Inheritance

object ← BoyerMooreSearch

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

Contains(ReadOnlySpan<byte>, byte[])

```
public static bool Contains(ReadOnlySpan<byte> haystack, byte[] needle)
```

Parameters

`haystack` ReadOnlySpan<byte>

`needle` byte[]

Returns

bool

Class DatabaseParser

Namespace: IDSApp.ProtocolParsing

Assembly: IDSApp.dll

Database protocol parser for Intrusion Detection System

Main Responsibilities:

- Parse and analyze database network traffic for security threats
- Extract SQL queries, commands, and session information
- Detect suspicious database activities and potential attacks
- Generate security logs and alerts for database operations

Supported Database Engines:

- MySQL (port 3306)
- PostgreSQL (port 5432)
- SQL Server (port 1433)
- Generic SQL traffic analysis

Security Detection Capabilities:

- Dangerous SQL commands (DROP, DELETE, ALTER, etc.)
- Suspicious query patterns and keywords
- Long-running queries indicating potential attacks
- Unauthorized database access attempts
- SQL injection pattern detection

Features:

- Protocol-aware parsing based on destination ports
- Payload fingerprinting for attack correlation
- Session tracking and correlation
- Comprehensive logging with performance considerations
- Memory-efficient payload handling

```
public class DatabaseParser : IDisposable
```

Inheritance

object ← DatabaseParser

Implements
IDisposable

Inherited Members
object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

DatabaseParser()

```
public DatabaseParser()
```

Properties

Body

Access to raw payload data

```
public byte[] Body { get; }
```

Property Value

byte[]

Methods

Dispose()

Clean up resources

```
public void Dispose()
```

Parse(byte[], string, string, int, int, string)

Parse database protocol packet and extract security-relevant information

Processing Steps:

1. Validate input parameters and payload size
2. Identify database engine based on port number
3. Extract SQL commands and query text
4. Detect suspicious patterns and dangerous operations
5. Log security events and parsed information

Supported Ports:

- 3306: MySQL
- 5432: PostgreSQL
- 1433: SQL Server

```
public void Parse(byte[] payload, string srcIp, string dstIp, int srcPort, int dstPort,
string sessionId = null)
```

Parameters

`payload` byte[]

`srcIp` string

`dstIp` string

`srcPort` int

`dstPort` int

`sessionId` string

Reset()

Reset parser state for processing new packet Clears all extracted information and payload buffer

```
public void Reset()
```

Class DhcpParser

Namespace: IDSApp.ProtocolParsing

Assembly: IDSApp.dll

DHCP protocol parser for Intrusion Detection System

Main Responsibilities:

- Parse and analyze DHCP (Dynamic Host Configuration Protocol) network traffic
- Detect DHCP-based attacks and suspicious activities
- Monitor IP address assignment and lease management
- Identify rogue DHCP servers and IP spoofing attempts

DHCP Protocol Support:

- Message Types: DISCOVER, OFFER, REQUEST, ACK, NAK, RELEASE, DECLINE, INFORM
- Standard Ports: 67 (server), 68 (client)
- Option parsing and validation
- Lease duration monitoring

Security Detection Capabilities:

- Rogue DHCP server detection
- IP address spoofing in DHCP transactions
- Suspicious lease duration values
- Malicious content in DHCP options
- DHCP starvation attacks
- Unauthorized IP assignment attempts

Features:

- Comprehensive DHCP header parsing
- Option analysis with security validation
- Transaction tracking and correlation
- Timestamp validation and correction
- Payload fingerprinting for attack correlation

```
public class DhcpParser : IDisposable
```

Inheritance

object ← DhcpParser

Implements
IDisposable

Inherited Members
object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

DhcpParser()

```
public DhcpParser()
```

Properties

Body

Access to raw payload data

```
public byte[] Body { get; }
```

Property Value

byte[]

Methods

Dispose()

Clean up resources

```
public void Dispose()
```

Parse(byte[], string, string, int, int, string)

Parse DHCP protocol packet and extract security-relevant information

Processing Steps:

1. Validate input parameters and payload size
2. Verify standard DHCP ports (67/68)
3. Parse DHCP header and options
4. Extract message type, transaction ID, and IP addresses
5. Detect suspicious patterns and security violations
6. Log security events and parsed information

Security Checks:

- Rogue DHCP server detection
- IP spoofing validation
- Suspicious lease durations
- Malicious option content

```
public void Parse(byte[] payload, string srcIp, string dstIp, int srcPort, int dstPort,
string sessionId = null)
```

Parameters

`payload` byte[]

`srcIp` string

`dstIp` string

`srcPort` int

`dstPort` int

`sessionId` string

Reset()

Reset parser state for processing new DHCP packet Clears all extracted information and payload buffer

```
public void Reset()
```

Class DnsParser

Namespace: IDSApp.ProtocolParsing

Assembly: IDSApp.dll

DNS protocol parser for Intrusion Detection System

Main Responsibilities:

- Parse and analyze DNS (Domain Name System) network traffic
- Extract DNS queries and responses for security analysis
- Detect DNS-based attacks and suspicious domain activities
- Monitor DNS tunneling and data exfiltration attempts

DNS Protocol Support:

- Query Types: A, AAAA, NS, CNAME, PTR, and other common record types
- Standard Port: 53 (UDP/TCP)
- DNS packet structure parsing (header, questions, answers)
- Domain name compression pointer resolution

Security Detection Capabilities:

- DNS tunneling detection through suspicious domain patterns
- Data exfiltration via DNS queries
- Malicious domain name patterns and DGA (Domain Generation Algorithm)
- Suspicious query types and response patterns
- DNS amplification attack indicators

Features:

- Complete DNS packet parsing with compression handling
- Question and answer section analysis
- IPv4 and IPv6 address parsing
- Suspicious domain pattern matching
- Comprehensive logging with forensic data

```
public class DnsParser : IDisposable
```

Inheritance

object ← DnsParser

Implements
IDisposable

Inherited Members
object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

DnsParser()

```
public DnsParser()
```

Properties

Body

Access to raw payload data

```
public byte[] Body { get; }
```

Property Value

byte[]

Methods

Dispose()

Clean up resources

```
public void Dispose()
```

Parse(byte[], string, string, int, int, string)

Parse DNS protocol packet and extract queries and responses

Processing Steps:

1. Validate input parameters and payload size
2. Verify standard DNS port (53)
3. Parse DNS header to get question and answer counts
4. Extract question section with domain names and types
5. Extract answer section with response data
6. Detect suspicious patterns in domains and responses
7. Log security events and parsed information

DNS Packet Structure:

- Bytes 0-1: Transaction ID
- Bytes 2-3: Flags
- Bytes 4-5: Question Count
- Bytes 6-7: Answer Count
- Byte 12+: Question and Answer sections

```
public (List<DnsParser.DnsRecord> Questions, List<DnsParser.DnsRecord> Answers) Parse(byte[]
payload, string srcIp, string dstIp, int srcPort, int dstPort, string sessionId = null)
```

Parameters

`payload` byte[]

`srcIp` string

`dstIp` string

`srcPort` int

`dstPort` int

`sessionId` string

Returns

(List<DnsParser.DnsRecord> Questions, List<DnsParser.DnsRecord> Answers)

Reset()

Reset parser state for processing new DNS packet Clears all extracted information and payload buffer

```
public void Reset()
```

Class DnsParser.DnsRecord

Namespace: IDSApp.ProtocolParsing

Assembly: IDSApp.dll

Represents a DNS record with name, type, and response data Used for both questions and answers in DNS packets

```
public class DnsParser.DnsRecord
```

Inheritance

object ← DnsParser.DnsRecord

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Name

```
public string Name { get; set; }
```

Property Value

string

Response

```
public string Response { get; set; }
```

Property Value

string

Type

```
public string Type { get; set; }
```

Property Value

string

Class EnhancedHttpHandler

Namespace: IDSApp.ProtocolParsing

Assembly: IDSApp.dll

Advanced HTTP protocol parsing and analysis system for Intrusion Detection System

Main Responsibilities:

- Parse and analyze HTTP/HTTPS traffic for security threats
- Extract and validate HTTP headers, methods, URLs, and body content
- Detect malicious patterns in HTTP requests and responses
- Integrate TLS/SSL certificate analysis for encrypted traffic
- Provide comprehensive threat scoring and pattern matching

Security Detection Capabilities:

- Cross-Site Scripting (XSS) attacks
- SQL Injection (SQLi) attempts
- Local File Inclusion (LFI) and path traversal
- Remote Code Execution (RCE) patterns
- Malicious file uploads and binary content
- Base64-encoded payloads and evasion techniques
- Suspicious HTTP headers and user agents

Features:

- Complete HTTP protocol parsing with validation
- Multi-layer threat detection (text, binary, regex patterns)
- TLS/SSL certificate fingerprinting and analysis
- Content-type aware inspection optimization
- Comprehensive logging with forensic data
- Performance-optimized pattern matching

```
public class EnhancedHttpHandler : IHttpParserHandler, IDisposable
```

Inheritance

object ← EnhancedHttpHandler

Implements

IHttpParserHandler, IDisposable

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

EnhancedHttpHandler(HttpParser)

```
public EnhancedHttpHandler(HttpParser httpParser)
```

Parameters

`httpParser` HttpParser

Properties

Headers

```
public Dictionary<string, string> Headers { get; }
```

Property Value

Dictionary<string, string>

Host

```
public string Host { get; }
```

Property Value

string

Method

```
public string Method { get; }
```

Property Value

string

RequestBody

```
public byte[] RequestBody { get; }
```

Property Value

byte[]

ResponseBody

```
public byte[] ResponseBody { get; }
```

Property Value

byte[]

StatusCode

```
public int StatusCode { get; }
```

Property Value

int

TlsCipher

```
public string TlsCipher { get; set; }
```

Property Value

string

TlsFingerprint

```
public string TlsFingerprint { get; set; }
```

Property Value

string

TlsSni

```
public string TlsSni { get; set; }
```

Property Value

string

TlsVersion

```
public string TlsVersion { get; set; }
```

Property Value

string

Url

```
public string Url { get; }
```

Property Value

string

UserAgent

```
public string UserAgent { get; }
```

Property Value

string

Methods

Dispose()

Clean up resources

```
public void Dispose()
```

OnBody(HttpParser, ArraySegment<byte>)

```
public void OnBody(HttpParser parser, ArraySegment<byte> data)
```

Parameters

`parser` HttpParser

`data` ArraySegment<byte>

OnBody(HttpParser, ArraySegment<byte>)

```
public void OnBody(HttpParser parser, ArraySegment<byte> data)
```

Parameters

`parser` HttpParser

`data` ArraySegment<byte>

OnFragment(HttpParser, string)

```
public void OnFragment(HttpParser parser, string fragment)
```

Parameters

`parser` HttpParser

`fragment` string

OnHeaderName(HttpParser, string)

```
public void OnHeaderName(HttpParser parser, string name)
```

Parameters

`parser` HttpParser

`name` string

OnHeaderName(HttpParser, string)

```
public void OnHeaderName(HttpParser parser, string name)
```

Parameters

`parser` HttpParser

`name` string

OnHeaderValue(HttpParser, string)

```
public void OnHeaderValue(HttpParser parser, string value)
```

Parameters

`parser` HttpParser

`value` string

OnHeaderValue(HttpParser, string)

```
public void OnHeaderValue(HttpParser parser, string value)
```

Parameters

`parser` HttpParser

`value` string

OnHeadersEnd(HttpParser)

```
public void OnHeadersEnd(HttpParser parser)
```

Parameters

`parser` HttpParser

OnHeadersEnd(HttpParser)

```
public void OnHeadersEnd(HttpParser parser)
```

Parameters

`parser` HttpParser

OnMessageBegin(HttpParser)

```
public void OnMessageBegin(HttpParser parser)
```

Parameters

`parser` HttpParser

OnMessageBegin(HttpParser)

```
public void OnMessageBegin(HttpParser parser)
```

Parameters

`parser` HttpParser

OnMessageEnd(HttpParser)

```
public void OnMessageEnd(HttpParser parser)
```

Parameters

`parser` HttpParser

OnMessageEnd(HttpParser)

```
public void OnMessageEnd(HttpParser parser)
```

Parameters

`parser` HttpParser

OnMethod(HttpParser, string)

```
public void OnMethod(HttpParser parser, string method)
```

Parameters

`parser` HttpParser

`method` string

OnMethod(HttpParser, string)

```
public void OnMethod(HttpParser parser, string method)
```

Parameters

`parser` HttpParser

`method` string

OnQueryString(HttpParser, string)

```
public void OnQueryString(HttpParser parser, string queryString)
```

Parameters

`parser` HttpParser

`queryString` string

OnRequestUri(HttpParser, string)

```
public void OnRequestUri(HttpParser parser, string requestUri)
```

Parameters

`parser` HttpParser

`requestUri` string

OnRequestUri(HttpParser, string)

```
public void OnRequestUri(HttpParser parser, string requestUri)
```

Parameters

`parser` HttpParser

`requestUri` string

OnResponseCode(HttpParser, int, string)

```
public void OnResponseCode(HttpParser parser, int statusCode, string statusDescription)
```

Parameters

`parser` HttpParser

`statusCode` int

`statusDescription` string

OnResponseCode(HttpParser, int, string)

```
public void OnResponseCode(HttpParser parser, int statusCode, string statusDescription)
```

Parameters

`parser` HttpParser

`statusCode` int

`statusDescription` string

Reset()

Reset parser state for processing new HTTP message

```
public void Reset()
```

Class FtpParseResult

Namespace: IDSApp.ProtocolParsing

Assembly: IDSApp.dll

Comprehensive result container for FTP parsing analysis Contains parsed command information and security assessment flags

```
public class FtpParseResult
```

Inheritance

object ← FtpParseResult

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Properties

Command

FTP command extracted from payload (e.g., RETR, STOR, PASS)

```
public string Command { get; set; }
```

Property Value

string

ContainsCredentials

Flag indicating potential credential exposure

```
public bool ContainsCredentials { get; set; }
```

Property Value

bool

Filename

Extracted filename from file operations

```
public string Filename { get; set; }
```

Property Value

string

IsFileOperation

Indicates if this command operates on files

```
public bool IsFileOperation { get; set; }
```

Property Value

bool

IsSuspicious

Flag indicating suspicious activity detected

```
public bool IsSuspicious { get; set; }
```

Property Value

bool

NormalizedFilename

Normalized and sanitized file path for analysis

```
public string NormalizedFilename { get; set; }
```

Property Value

string

RawArgument

Raw argument string from the FTP command

```
public string RawArgument { get; set; }
```

Property Value

string

SuspicionReasons

List of reasons why the activity is considered suspicious

```
public List<string> SuspicionReasons { get; set; }
```

Property Value

List <string>

Methods

ToString()

Returns a string that represents the current object.

```
public override string ToString()
```

Returns

string

A string that represents the current object.

Class FtpParser

Namespace: IDSApp.ProtocolParsing

Assembly: IDSApp.dll

Advanced FTP protocol parser for Intrusion Detection System

Main Responsibilities:

- Parse and analyze FTP (File Transfer Protocol) commands and transactions
- Extract filenames, paths, and command arguments with security context
- Detect suspicious FTP activities and potential attack patterns
- Normalize paths and identify path traversal attempts
- Flag credential exposure in FTP sessions

Security Detection Capabilities:

- Path traversal attacks using "../" and absolute paths
- Suspicious file extensions indicating malware/scripts
- Base64-encoded payloads in filenames or arguments
- Suspicious keywords (shell, cmd, webshell, etc.)
- Credential exposure in PASS commands
- Long filenames potentially hiding malicious content
- Percent-encoding evasion techniques

Features:

- Comprehensive FTP command parsing with argument extraction
- Path normalization and traversal detection
- File operation classification (RETR, STOR, DELE, etc.)
- Multiple encoding detection and handling
- Suspicious pattern matching with detailed reasoning

```
public class FtpParser
```

Inheritance

object ← FtpParser

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

Parse(byte[])

Parse FTP protocol payload and extract security-relevant information

Processing Steps:

1. Decode payload using UTF-8 with ASCII fallback
2. Extract command and argument using regex
3. Handle quoted filenames and path normalization
4. Analyze for suspicious patterns and security threats
5. Return comprehensive parsing results with security flags

Supported Analysis:

- File operation detection and filename extraction
- Path traversal and directory climbing attempts
- Suspicious file extensions and encoded payloads
- Credential exposure in authentication commands
- Evasion technique detection

```
public FtpParseResult Parse(byte[] payload)
```

Parameters

`payload` byte[]

Returns

FtpParseResult

Class HttpParser

Namespace: IDSApp.ProtocolParsing

Assembly: IDSApp.dll

Advanced HTTP threat detection engine for Intrusion Detection System

Detection Methods:

- Text pattern matching for common attack signatures
- Regular expression analysis for complex attack patterns
- Binary pattern detection for executable content
- Base64 decoding and analysis for encoded payloads
- URL decoding for obfuscated attack vectors

Performance Features:

- Configurable inspection limits to prevent resource exhaustion
- Content-type filtering to skip non-relevant data
- Efficient Boyer-Moore algorithm for binary pattern matching
- Threat scoring with adjustable thresholds

```
public class HttpParser
```

Inheritance

object ← HttpParser

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

HttpParser(int, int, bool, bool)

```
public HttpParser(int threatThreshold = 1, int maxInspectBytes = 1048576, bool
enableBase64Decode = true, bool enableUrlDecode = true)
```

Parameters

`threatThreshold` int

`maxInspectBytes` int

`enableBase64Decode` bool

`enableUrlDecode` bool

Methods

AnalyzeBody(byte[], string, string, string, string)

Comprehensive HTTP body analysis for security threats

```
public HttpParser.ThreatResult AnalyzeBody(byte[] body, string srcIp, string dstIp, string
direction, string contentType = null)
```

Parameters

`body` byte[]

`srcIp` string

`dstIp` string

`direction` string

`contentType` string

Returns

HttpParser.ThreatResult

CheckBodyForThreats(byte[], string, string, string, string)

```
public bool CheckBodyForThreats(byte[] body, string srcIp, string dstIp, string direction,
string contentType = null)
```

Parameters

`body` byte[]

`srcIp` string

`dstIp` string

`direction` string

`contentType` string

Returns

bool

IsValidHttpPacket(byte[])

Validate if packet contains valid HTTP protocol data

```
public bool IsValidHttpPacket(byte[] payload)
```

Parameters

`payload` byte[]

Returns

bool

Class HttpParser.ThreatResult

Namespace: IDSApp.ProtocolParsing

Assembly: IDSApp.dll

Threat analysis result container

```
public class HttpParser.ThreatResult
```

Inheritance

object ← HttpParser.ThreatResult

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Description

```
public string Description { get; set; }
```

Property Value

string

IsMalicious

```
public bool IsMalicious { get; set; }
```

Property Value

bool

MatchedPatterns

```
public List<string> MatchedPatterns { get; }
```

Property Value

List <string>

Score

```
public int Score { get; set; }
```

Property Value

int

Threshold

```
public int Threshold { get; set; }
```

Property Value

int

Class HttpParserHelpers

Namespace: IDSApp.ProtocolParsing

Assembly: IDSApp.dll

HTTP parser utility functions

```
public static class HttpParserHelpers
```

Inheritance

object ← HttpParserHelpers

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

GetContentTypeFromHeaders(Dictionary<string, string>)

```
public static string GetContentTypeFromHeaders(Dictionary<string, string> headers)
```

Parameters

`headers` Dictionary<string, string>

Returns

string

Class ImapParser

Namespace: IDSApp.ProtocolParsing

Assembly: IDSApp.dll

IMAP protocol parser for Intrusion Detection System

Main Responsibilities:

- Parse and analyze IMAP (Internet Message Access Protocol) traffic
- Monitor email client-server communications for security threats
- Detect suspicious IMAP commands and authentication attempts
- Analyze email content and attachments for malicious patterns
- Track session activity and command sequences

IMAP Protocol Support:

- Standard IMAP commands (LOGIN, SELECT, FETCH, STORE, APPEND, LOGOUT)
- Email folder operations and message management
- Authentication and session management
- Email content retrieval and analysis

Security Detection Capabilities:

- Suspicious command patterns and sequences
- Authentication brute force attempts
- Malicious payloads in email content
- Sensitive information extraction attempts
- Protocol anomalies and error conditions

Features:

- Comprehensive IMAP command and response parsing
- Configurable suspicious command detection
- Payload fingerprinting for content correlation
- Session tracking and attempt counting
- Real-time threat detection and logging

```
public class ImapParser : IDisposable
```

Inheritance

object ← ImapParser

Implements
IDisposable

Inherited Members
object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

ImapParser(IEnumerable<string>, IEnumerable<string>)

```
public ImapParser(IEnumerable<string> suspiciousCommands = null, IEnumerable<string>
suspiciousPayloadPatterns = null)
```

Parameters

`suspiciousCommands` IEnumerable<string>

`suspiciousPayloadPatterns` IEnumerable<string>

Properties

Body

Access to raw payload data

```
public byte[] Body { get; }
```

Property Value

byte[]

Methods

Dispose()

Clean up resources

```
public void Dispose()
```

Parse(byte[], string, string, int, int, string)

Comprehensive IMAP packet parsing method Processes complete IMAP conversations including commands, responses, and body content

```
public void Parse(byte[] payload, string srcIp, string dstIp, int srcPort, int dstPort,
string sessionId = null)
```

Parameters

`payload` byte[]

`srcIp` string

`dstIp` string

`srcPort` int

`dstPort` int

`sessionId` string

ParseBody(byte[])

Analyze IMAP message body content for security threats

Security Checks:

- Payload size validation to prevent memory exhaustion
- ASCII content analysis for suspicious patterns
- SHA256 fingerprinting for content correlation
- Pattern matching against known attack signatures

```
public void ParseBody(byte[] bodyData)
```

Parameters

`bodyData` byte[]

ParseCommand(string, string, string, string, DateTime?, int)

Parse IMAP command line and extract security-relevant information

Processing Steps:

1. Validate input parameters and command format
2. Extract command type and arguments
3. Track authentication attempts and suspicious commands
4. Monitor folder operations and message access

Supported Commands:

- LOGIN: Authentication attempts tracking
- SELECT: Folder access monitoring
- FETCH: Email content retrieval
- APPEND: Email submission
- STORE: Email modification

```
public void ParseCommand(string commandLine, string srcIp, string dstIp, string sessionId,
DateTime? timestamp = null, int dstPort = 143)
```

Parameters

`commandLine` string

`srcIp` string

`dstIp` string

`sessionId` string

`timestamp` DateTime?

`dstPort` int

ParseResponse(string)

Parse IMAP server response for status and error conditions

Response Types:

- OK: Successful command completion
- NO: Command failure or error
- BAD: Protocol error or invalid command

```
public void ParseResponse(string responseLine)
```

Parameters

`responseLine` string

Reset()

Reset parser state for processing new IMAP session Clears all extracted information and payload buffer

```
public void Reset()
```

Class LdapParser

Namespace: IDSApp.ProtocolParsing

Assembly: IDSApp.dll

LDAP protocol parser for Intrusion Detection System

Main Responsibilities:

- Parse and analyze LDAP (Lightweight Directory Access Protocol) traffic
- Extract directory operations, authentication attempts, and search queries
- Detect suspicious LDAP activities and directory service attacks
- Monitor directory access patterns and privilege escalation attempts
- Analyze distinguished names and filter criteria for security threats

LDAP Protocol Support:

- Authentication operations (BindRequest, BindResponse)
- Directory search and query operations (SearchRequest, SearchResultEntry)
- Directory modification operations (Add, Modify, Delete)
- ASN.1 BER encoding parsing for LDAP packets
- Standard ports 389 (LDAP) and 636 (LDAPS)

Security Detection Capabilities:

- Suspicious distinguished name patterns
- Directory traversal and unauthorized access attempts
- LDAP injection and filter manipulation
- Privilege escalation through directory modifications
- Encrypted LDAPS traffic monitoring

Features:

- Complete ASN.1 BER protocol parsing
- Operation type extraction and classification
- Distinguished name analysis and validation
- Result code monitoring for failed operations
- Encrypted session detection and handling

```
public class LdapParser : IDisposable
```

Inheritance

object ← LdapParser

Implements
IDisposable

Inherited Members
object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

LdapParser()

```
public LdapParser()
```

Properties

Body

Access to raw payload data

```
public byte[] Body { get; }
```

Property Value

byte[]

Methods

Dispose()

Clean up resources

```
public void Dispose()
```

Parse(byte[], string, string, int, int, string)

Comprehensive LDAP packet parsing method Processes complete LDAP conversations including operations and responses

```
public void Parse(byte[] payload, string srcIp, string dstIp, int srcPort, int dstPort,
string sessionId = null)
```

Parameters

`payload` byte[]

`srcIp` string

`dstIp` string

`srcPort` int

`dstPort` int

`sessionId` string

Reset()

Reset parser state for processing new LDAP session Clears all extracted information and payload buffer

```
public void Reset()
```

Class NetbiosParser

Namespace: IDSApp.ProtocolParsing

Assembly: IDSApp.dll

```
public class NetbiosParser : IDisposable
```

Inheritance

object ← NetbiosParser

Implements

IDisposable

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

NetbiosParser()

Initializes a new instance of NetBIOS parser with default values Creates memory stream for payload storage and resets parser state

```
public NetbiosParser()
```

Properties

Body

Read-only property providing access to raw packet payload Returns copy of stored body stream as byte array

```
public byte[] Body { get; }
```

Property Value

byte[]

Methods

Dispose()

Implements IDisposable pattern for proper resource cleanup Disposes memory stream and marks instance as disposed Thread-safe disposal using lock synchronization

```
public void Dispose()
```

Parse(byte[], string, string, int, int, string)

Main packet parsing method - processes NetBIOS packet data Validates input parameters, checks packet size, and initiates parsing Handles errors gracefully and logs parsing results to database

```
public void Parse(byte[] payload, string srcIp, string dstIp, int srcPort, int dstPort,
string sessionId = null)
```

Parameters

`payload` byte[]

Raw packet bytes to parse

`srcIp` string

Source IP address

`dstIp` string

Destination IP address

`srcPort` int

Source port number

`dstPort` int

Destination port number

`sessionId` string

Optional session identifier

Reset()

Resets all parser fields to their default values Clears the body stream and prepares parser for new packet Thread-safe operation using lock synchronization

```
public void Reset()
```

Class NtpParser

Namespace: IDSApp.ProtocolParsing

Assembly: IDSApp.dll

```
public class NtpParser : IDisposable
```

Inheritance

object ← NtpParser

Implements

IDisposable

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

NtpParser()

Initializes a new instance of NTP parser with default values Creates memory stream for payload storage and resets parser state

```
public NtpParser()
```

Properties

Body

Read-only property providing access to raw packet payload Returns copy of stored body stream as byte array

```
public byte[] Body { get; }
```

Property Value

byte[]

Methods

Dispose()

Implements IDisposable pattern for proper resource cleanup Disposes memory stream and marks instance as disposed Thread-safe disposal using lock synchronization

```
public void Dispose()
```

Parse(byte[], string, string, int, int, string)

Main packet parsing method - processes NTP packet data Validates input parameters, checks packet size, and initiates parsing Handles errors gracefully and logs parsing results to database

```
public void Parse(byte[] payload, string srcIp, string dstIp, int srcPort, int dstPort,
string sessionId = null)
```

Parameters

`payload` byte[]

Raw packet bytes to parse

`srcIp` string

Source IP address

`dstIp` string

Destination IP address

`srcPort` int

Source port number

`dstPort` int

Destination port number

`sessionId` string

Optional session identifier

Reset()

Resets all parser fields to their default values Clears the body stream and prepares parser for new packet Thread-safe operation using lock synchronization

```
public void Reset()
```

Class Pop3Parser

Namespace: IDSApp.ProtocolParsing

Assembly: IDSApp.dll

```
public class Pop3Parser : IDisposable
```

Inheritance

object ← Pop3Parser

Implements

IDisposable

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

Pop3Parser(IEnumerable<string>, IEnumerable<string>)

Initializes POP3 parser with configurable security patterns Creates memory stream for payload storage and sets up detection rules

```
public Pop3Parser(IEnumerable<string> suspiciousCommands = null, IEnumerable<string>
suspiciousPayloadPatterns = null)
```

Parameters

`suspiciousCommands` IEnumerable<string>

Custom list of suspicious POP3 commands

`suspiciousPayloadPatterns` IEnumerable<string>

Custom regex patterns for payload analysis

Properties

Body

Read-only property providing access to stored email body content Returns copy of stored body stream as byte array

```
public byte[] Body { get; }
```

Property Value

byte[]

Methods

Dispose()

Implements IDisposable pattern for proper resource cleanup Disposes memory stream and marks instance as disposed Thread-safe disposal using lock synchronization

```
public void Dispose()
```

Parse(byte[], string, string, int, int, string)

Main packet parsing method - processes raw POP3 protocol data Automatically distinguishes between commands and responses Handles multi-line protocols and coordinates command/response/body parsing

```
public void Parse(byte[] payload, string srcIp, string dstIp, int srcPort, int dstPort,
string sessionId = null)
```

Parameters

`payload` byte[]

Raw packet bytes to parse

`srcIp` string

Source IP address

`dstIp` string

Destination IP address

`srcPort` int

Source port number

`dstPort` int

Destination port number

`sessionId` string

Optional session identifier

ParseBody(byte[])

Analyzes email body content for RETR and TOP commands Performs security scanning using regex patterns on ASCII content Generates payload fingerprints for tracking and correlation

```
public void ParseBody(byte[] bodyData)
```

Parameters

`bodyData` byte[]

Raw email body data bytes

ParseCommand(string, string, string, string, DateTime?, int)

Parses individual POP3 command lines from client to server Extracts commands, usernames, and tracks authentication attempts Detects suspicious commands based on configured patterns

```
public void ParseCommand(string commandLine, string srcIp, string dstIp, string sessionId,
DateTime? timestamp = null, int dstPort = 110)
```

Parameters

`commandLine` string

Raw command line string to parse

`srcIp` string

Source IP address (client)

`dstIp` string

Destination IP address (server)

`sessionId` string

Session identifier for correlation

`timestamp` DateTime?

Optional timestamp (uses current UTC if null)

`dstPort` int

Destination port (default 110)

ParseResponse(string)

Parses POP3 server response lines (+OK/-ERR) Extracts status codes, message sizes, and detects error conditions Automatically logs suspicious error responses

```
public void ParseResponse(string responseLine)
```

Parameters

`responseLine` string

Server response line to parse

Reset()

Resets all parser fields to default values for new session Clears body stream and resets counters while maintaining configuration Thread-safe operation using lock synchronization

```
public void Reset()
```

Class RdpParseResult

Namespace: IDSApp.ProtocolParsing

Assembly: IDSApp.dll

Result container for RDP parsing operations Stores extracted RDP session data, authentication information, and security analysis results Provides utility methods for result validation and formatting

```
public class RdpParseResult
```

Inheritance

object ← RdpParseResult

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Properties

AuthAttempts

Counter for authentication attempts detected in RDP traffic Used for brute-force attack detection and connection sequence analysis Default value: 0

```
public int AuthAttempts { get; set; }
```

Property Value

int

ClientIp

RDP client IP address (source) Used for connection tracking and client identification

```
public string ClientIp { get; set; }
```

Property Value

string

IsSuspicious

Flag indicating whether security heuristics detected suspicious activity Set to true when one or more security checks trigger Default value: false

```
public bool IsSuspicious { get; set; }
```

Property Value

bool

ServerIp

RDP server IP address (destination) Used for connection tracking and network path analysis

```
public string ServerIp { get; set; }
```

Property Value

string

SessionId

Extracted RDP session identifier or connection cookie Typically contains mstshash values or custom session identifiers Default value: "unknown"

```
public string SessionId { get; set; }
```

Property Value

string

SuspicionReasons

List of reasons why the RDP traffic was flagged as suspicious Contains specific security alerts and detection rationale

```
public List<string> SuspicionReasons { get; set; }
```

Property Value

List <string>

Methods

HasMeaningfulData()

Determines if the parse result contains meaningful RDP data Checks for valid session IDs and authentication activity Used to filter out incomplete or irrelevant parsing results

```
public bool HasMeaningfulData()
```

Returns

bool

True if result contains valid RDP session data, false otherwise

ToString()

Provides formatted string representation of RDP parse results Includes all key properties and security findings for logging and display

```
public override string ToString()
```

Returns

string

Formatted string containing RDP analysis results

Class RdpParser

Namespace: IDSApp.ProtocolParsing

Assembly: IDSApp.dll

```
public class RdpParser
```

Inheritance

object ← RdpParser

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

Parse(byte[], string, string, int)

Main RDP packet parsing method - processes RDP protocol traffic Analyzes TPKT headers, PDU types, and extracts session information Performs security checks and heuristic analysis for threat detection

```
public RdpParseResult Parse(byte[] payload, string srcIp, string dstIp, int srcPort)
```

Parameters

`payload` byte[]

Raw packet bytes to parse

`srcIp` string

Source IP address (client)

`dstIp` string

Destination IP address (RDP server)

`srcPort` int

Source port number

Returns

RdpParseResult

RdpParseResult containing parsed data and security analysis

Class SmbParseResult

Namespace: IDSApp.ProtocolParsing

Assembly: IDSApp.dll

Result container for SMB parsing operations Stores extracted SMB command data, file information, and security analysis results Provides formatted output for logging and monitoring systems

```
public class SmbParseResult
```

Inheritance

object ← SmbParseResult

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Properties

Command

Extracted SMB command name (Negotiate, SessionSetup, Create, Read, Write, etc.) Indicates the type of SMB operation being performed Default value: "unknown"

```
public string Command { get; set; }
```

Property Value

string

Filename

Extracted filename from SMB Create commands Contains the full path and filename being accessed Default value: "none"

```
public string Filename { get; set; }
```

Property Value

string

IsSuspicious

Flag indicating whether security checks detected suspicious activity Set to true when file extension or path analysis triggers alerts Default value: false

```
public bool IsSuspicious { get; set; }
```

Property Value

bool

Share

SMB share name being accessed (placeholder in current implementation) Future enhancement would extract actual share names from SMB packets Default value: "none"

```
public string Share { get; set; }
```

Property Value

string

SuspicionReasons

List of reasons why the SMB traffic was flagged as suspicious Contains specific security alerts and detection rationale

```
public List<string> SuspicionReasons { get; set; }
```

Property Value

List <string>

Methods

ToString()

Provides formatted string representation of SMB parse results Includes all key properties and security findings for logging and display

```
public override string ToString()
```

Returns

string

Formatted string containing SMB analysis results

Class SmbParser

Namespace: IDSApp.ProtocolParsing

Assembly: IDSApp.dll

Advanced SMB parser supporting SMB1/SMB2/SMB3, extracting command, filename, share, and performing IDS checks for suspicious activity.

```
public class SmbParser
```

Inheritance

object ← SmbParser

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

Parse(byte[])

Main SMB packet parsing method - processes SMB protocol traffic Validates SMB headers, extracts commands and filenames, performs security analysis Focuses on SMB1 protocol with basic command and file operation extraction

```
public SmbParseResult Parse(byte[] payload)
```

Parameters

`payload` byte[]

Raw packet bytes to parse

Returns

SmbParseResult

SmbParseResult containing parsed SMB data and security analysis

Class SmtpParseResult

Namespace: IDSApp.ProtocolParsing

Assembly: IDSApp.dll

Result container for SMTP parsing operations Stores extracted email metadata, security analysis results, and suspicion reasons Provides formatted output for logging and email security monitoring

```
public class SmtpParseResult
```

Inheritance

object ← SmtpParseResult

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Properties

FromAddress

Extracted sender email address from MAIL FROM command Contains the envelope sender address for the email Default value: "unknown"

```
public string FromAddress { get; set; }
```

Property Value

string

IsSuspicious

Flag indicating whether security checks detected suspicious email activity Set to true when keywords or dangerous attachments are detected Default value: false

```
public bool IsSuspicious { get; set; }
```

Property Value

bool

Subject

Extracted and decoded email subject line Includes MIME decoding for internationalized content Default value: "none"

```
public string Subject { get; set; }
```

Property Value

string

SuspicionReasons

List of reasons why the email was flagged as suspicious Contains specific security alerts and detection rationale

```
public List<string> SuspicionReasons { get; set; }
```

Property Value

List<string>

ToAddress

Extracted recipient email address from RCPT TO command Contains the envelope recipient address for the email Default value: "unknown"

```
public string ToAddress { get; set; }
```

Property Value

string

Methods

ToString()

Provides formatted string representation of SMTP parse results Includes all key properties and security findings for logging and display

```
public override string ToString()
```

Returns

string

Formatted string containing email analysis results

Class SmtpParser

Namespace: IDSApp.ProtocolParsing

Assembly: IDSApp.dll

Advanced SMTP parser that extracts From, To, Subject, decodes MIME headers, handles multi-line subjects, and performs basic IDS checks for suspicious content and attachments.

```
public class SmtpParser
```

Inheritance

object ← SmtpParser

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

Parse(byte[])

Main SMTP packet parsing method - processes email protocol data Extracts email headers, decodes MIME content, and performs security analysis Handles multi-line headers and various email encoding formats

```
public SmtpParseResult Parse(byte[] payload)
```

Parameters

`payload` byte[]

Raw SMTP packet bytes to parse

Returns

SmtpParseResult

SmtpParseResult containing extracted email data and security analysis

Class SnmpParser

Namespace: IDSApp.ProtocolParsing

Assembly: IDSApp.dll

```
public class SnmpParser : IDisposable
```

Inheritance

object ← SnmpParser

Implements

IDisposable

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

SnmpParser(IEnumerable<string>, IEnumerable<string>)

Initializes SNMP parser with configurable security patterns and thresholds Creates memory stream for payload storage and sets up detection rules

```
public SnmpParser(IEnumerable<string> suspiciousCommunities = null, IEnumerable<string>
suspiciousRequestTypes = null)
```

Parameters

`suspiciousCommunities` IEnumerable<string>

Custom list of suspicious SNMP community strings

`suspiciousRequestTypes` IEnumerable<string>

Custom list of suspicious SNMP operation types

Properties

Body

Read-only property providing access to raw packet payload Returns copy of stored body stream as byte array

```
public byte[] Body { get; }
```

Property Value

byte[]

Methods

Dispose()

Implements IDisposable pattern for proper resource cleanup Disposes memory stream and marks instance as disposed Thread-safe disposal using lock synchronization

```
public void Dispose()
```

Parse(byte[], string, string, int, int, string)

Main SNMP packet parsing method - processes SNMP protocol traffic Validates input parameters, checks packet size, and initiates ASN.1 parsing Handles both SNMP requests and responses with security analysis

```
public void Parse(byte[] payload, string srcIp, string dstIp, int srcPort, int dstPort,
string sessionId = null)
```

Parameters

`payload` byte[]

Raw packet bytes to parse

`srcIp` string

Source IP address

`dstIp` string

Destination IP address

`srcPort` int

Source port number

`dstPort` int

Destination port number

`sessionId` string

Optional session identifier

ParseRequest(string, string, string, IEnumerable<string>, string, string, string, DateTime?, byte[])

Processes SNMP request data and performs security analysis Validates community strings, request types, and OID patterns Detects bulk requests, suspicious operations, and weak authentication

```
public void ParseRequest(string version, string community, string requestType,
IEnumerable<string> oids, string srcIp, string dstIp, string sessionId, DateTime? timestamp
= null, byte[] rawPayload = null)
```

Parameters

`version` string

SNMP protocol version

`community` string

SNMP community string

`requestType` string

SNMP operation type

`oids` IEnumerable<string>

List of OIDs being queried

`srcIp` string

Source IP address

`dstIp` string

Destination IP address

`sessionId` string

Session identifier

`timestamp` DateTime?

Optional timestamp

`rawPayload` byte[]

Raw packet bytes for fingerprinting

ParseResponse(string, string, byte[])

Processes SNMP response data and performs content analysis Scans response values for sensitive information and attack indicators Monitors payload size and response patterns for anomalies

```
public void ParseResponse(string value, string responseCode = "OK", byte[] payload = null)
```

Parameters

`value` string

SNMP response value data

`responseCode` string

SNMP response status code

`payload` byte[]

Raw response payload for analysis

Reset()

Resets all parser fields to default values for new session Clears OID list, body stream, and resets state while maintaining configuration Thread-safe operation using lock synchronization

```
public void Reset()
```

Class SshParser

Namespace: IDSApp.ProtocolParsing

Assembly: IDSApp.dll

```
public class SshParser
```

Inheritance

object ← SshParser

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

Parse(byte[], string, string, int)

Main SSH packet parsing method - processes SSH protocol handshake data Extracts client and server version information from SSH protocol exchange Detects authentication attempts and analyzes protocol version patterns

```
public (string clientVersion, string serverVersion, int authAttempts) Parse(byte[] payload,
string srcIp, string dstIp, int srcPort)
```

Parameters

`payload` byte[]

Raw SSH packet bytes to parse

`srcIp` string

Source IP address of SSH traffic

`dstIp` string

Destination IP address (SSH server/client)

`srcPort` int

Source port number for direction analysis

Returns

(string clientVersion, string serverVersion, int authAttempts)

Tuple containing:

- clientVersion: Extracted SSH client version string
- serverVersion: Extracted SSH server version string
- authAttempts: Count of detected authentication attempts

Class TelnetParseResult

Namespace: IDSApp.ProtocolParsing

Assembly: IDSApp.dll

Result container for Telnet parsing operations Stores extracted Telnet commands, connection information, and security analysis results Provides utility methods for result validation and formatted output

```
public class TelnetParseResult
```

Inheritance

object ← TelnetParseResult

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Properties

AuthAttempts

Counter for authentication attempts and suspicious activity levels

- 0: No authentication detected
- 1: Authentication patterns detected (login/password)
- 2: Suspicious commands detected (shell access, file operations) Default value: 0

```
public int AuthAttempts { get; set; }
```

Property Value

int

ClientIp

Source IP address of Telnet client Used for connection tracking and client identification

```
public string ClientIp { get; set; }
```

Property Value

string

Command

Extracted Telnet command text after removing control sequences Contains clear-text user commands and data entered in Telnet session Default value: empty string

```
public string Command { get; set; }
```

Property Value

string

ServerIp

Destination IP address of Telnet server Used for connection tracking and server identification

```
public string ServerIp { get; set; }
```

Property Value

string

Methods

HasMeaningfulData()

Determines if the parse result contains meaningful Telnet data Checks for valid commands and filters out errors or empty results

```
public bool HasMeaningfulData()
```

Returns

bool

True if result contains valid Telnet command data, false otherwise

ToString()

Provides formatted string representation of Telnet parse results Includes all key properties for logging and display purposes

```
public override string ToString()
```

Returns

string

Formatted string containing Telnet analysis results

Class TelnetParser

Namespace: IDSApp.ProtocolParsing

Assembly: IDSApp.dll

```
public class TelnetParser
```

Inheritance

object ← TelnetParser

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

Parse(byte[], string, string, int)

Main Telnet packet parsing method - processes Telnet protocol traffic Filters out Telnet control sequences and extracts clear-text commands Detects authentication attempts and suspicious command patterns

```
public TelnetParseResult Parse(byte[] payload, string srcIp, string dstIp, int srcPort)
```

Parameters

`payload` byte[]

Raw Telnet packet bytes to parse

`srcIp` string

Source IP address of Telnet traffic

`dstIp` string

Destination IP address (Telnet server)

`srcPort` int

Source port number for direction analysis

Returns

TelnetParseResult

TelnetParseResult containing extracted commands and security analysis

Class TftpParser

Namespace: IDSApp.ProtocolParsing

Assembly: IDSApp.dll

```
public class TftpParser : IDisposable
```

Inheritance

object ← TftpParser

Implements

IDisposable

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

TftpParser()

Initializes a new instance of TFTP parser with default values Creates memory stream for payload storage and resets parser state

```
public TftpParser()
```

Properties

Body

Read-only property providing access to raw packet payload Returns copy of stored body stream as byte array

```
public byte[] Body { get; }
```

Property Value

byte[]

Methods

Dispose()

Implements IDisposable pattern for proper resource cleanup Disposes memory stream and marks instance as disposed Thread-safe disposal using lock synchronization

```
public void Dispose()
```

Parse(byte[], string, string, int, int, string)

Main TFTP packet parsing method - processes TFTP protocol traffic Validates input parameters, checks packet size, and initiates TFTP parsing Handles errors gracefully and logs parsing results to database

```
public void Parse(byte[] payload, string srcIp, string dstIp, int srcPort, int dstPort,
string sessionId = null)
```

Parameters

`payload` byte[]

Raw packet bytes to parse

`srcIp` string

Source IP address

`dstIp` string

Destination IP address

`srcPort` int

Source port number

`dstPort` int

Destination port number

`sessionId` string

Optional session identifier

Reset()

Resets all parser fields to their default values Clears the body stream and prepares parser for new packet Thread-safe operation using lock synchronization

```
public void Reset()
```

Class TlsParser

Namespace: IDSApp.ProtocolParsing

Assembly: IDSApp.dll

Production-ready TLS parser:

- Parses TLS records and handshake (ClientHello, Certificate)
- Extracts SNI, computes a best-effort JA3 fingerprint
- Extracts certificate(s) from Certificate handshake and inspects them (self-signed, expired, SHA256 fingerprint)
- Performs lightweight deep-inspection on ASCII content (configurable)
- Logs results to DB via LogBLL.Insert and saves TLS-ERROR rows also via LogBLL.Insert
- Loads known-malicious JA3s from Ja3Repository (DB-backed)

```
public class TlsParser
```

Inheritance

object ← TlsParser

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

InitializeJa3Store(IEnumerable<string>)

Initializes the JA3 fingerprint store with known malicious fingerprints Called at application startup to load threat intelligence data

```
public static void InitializeJa3Store(IEnumerable<string> initialJa3s)
```

Parameters

`initialJa3s` IEnumerable<string>

Collection of malicious JA3 fingerprints to load

Parse(byte[], string)

Main TLS packet parsing method - processes TLS/SSL protocol traffic Performs comprehensive TLS analysis including handshake parsing, fingerprinting, and security validation Implements multi-layer threat detection and extensive error handling

```
public TlsParser.TlsParseResult Parse(byte[] payload, string srcIp)
```

Parameters

`payload` byte[]

Raw TLS packet bytes to parse

`srcIp` string

Source IP address of TLS connection

Returns

TlsParser.TlsParseResult

TlsParseResult containing comprehensive TLS analysis and security findings

Class TlsParser.TlsParseResult

Namespace: IDSApp.ProtocolParsing

Assembly: IDSApp.dll

Comprehensive result container for TLS parsing operations Stores extracted TLS handshake data, security analysis results, and threat intelligence matches

```
public class TlsParser.TlsParseResult
```

Inheritance

object ← TlsParser.TlsParseResult

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

AlertReason

Reasons for security alerts and suspicious flags

```
public string AlertReason { get; set; }
```

Property Value

string

CertificateCN

Common Name from server certificate

```
public string CertificateCN { get; set; }
```

Property Value

string

CertificateFingerprintsSha256

List of SHA-256 certificate fingerprints in chain

```
public List<string> CertificateFingerprintsSha256 { get; set; }
```

Property Value

List <string>

CertificateIssuer

Issuer Common Name from server certificate

```
public string CertificateIssuer { get; set; }
```

Property Value

string

CertificateSummary

Summary of certificate chain validation results

```
public string CertificateSummary { get; set; }
```

Property Value

string

CipherSuiteSummary

Summary of cipher suites offered by client

```
public string CipherSuiteSummary { get; set; }
```

Property Value

string

IsSuspicious

Flag indicating suspicious activity detection

```
public bool IsSuspicious { get; set; }
```

Property Value

bool

Ja3Fingerprint

JA3 TLS fingerprint for client identification

```
public string Ja3Fingerprint { get; set; }
```

Property Value

string

MatchedKnownMaliciousJa3

Flag indicating match with known malicious JA3 fingerprint

```
public bool MatchedKnownMaliciousJa3 { get; set; }
```

Property Value

bool

Sni

Server Name Indication (SNI) from ClientHello

```
public string Sni { get; set; }
```

Property Value

string

SrcIp

Source IP address of TLS connection

```
public string SrcIp { get; set; }
```

Property Value

string

Version

TLS protocol version (SSL 3.0, TLS 1.0-1.3)

```
public string Version { get; set; }
```

Property Value

string