

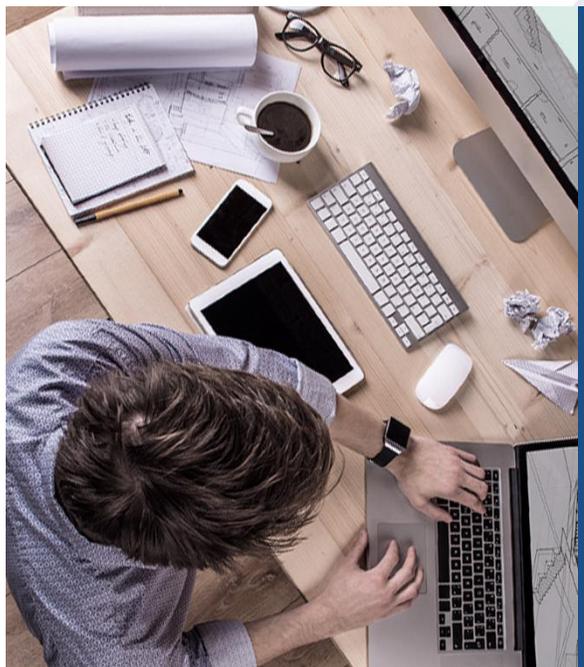


SANGFOR

SANGFOR_NGAF_V8.0.6_Professional

Attack Protection 2





- 1 Common Web Application Attack
- 2 SQL Injection
- 3 Whitelist

1. Common Web Application Attack



Common Web Application Attack

The OWASP Top 10 is a powerful awareness document for web application security. It represents a broad consensus about the most critical security risks to web applications.

From 2007–2017, the Injection and XSS attack is always at top 3 attack.

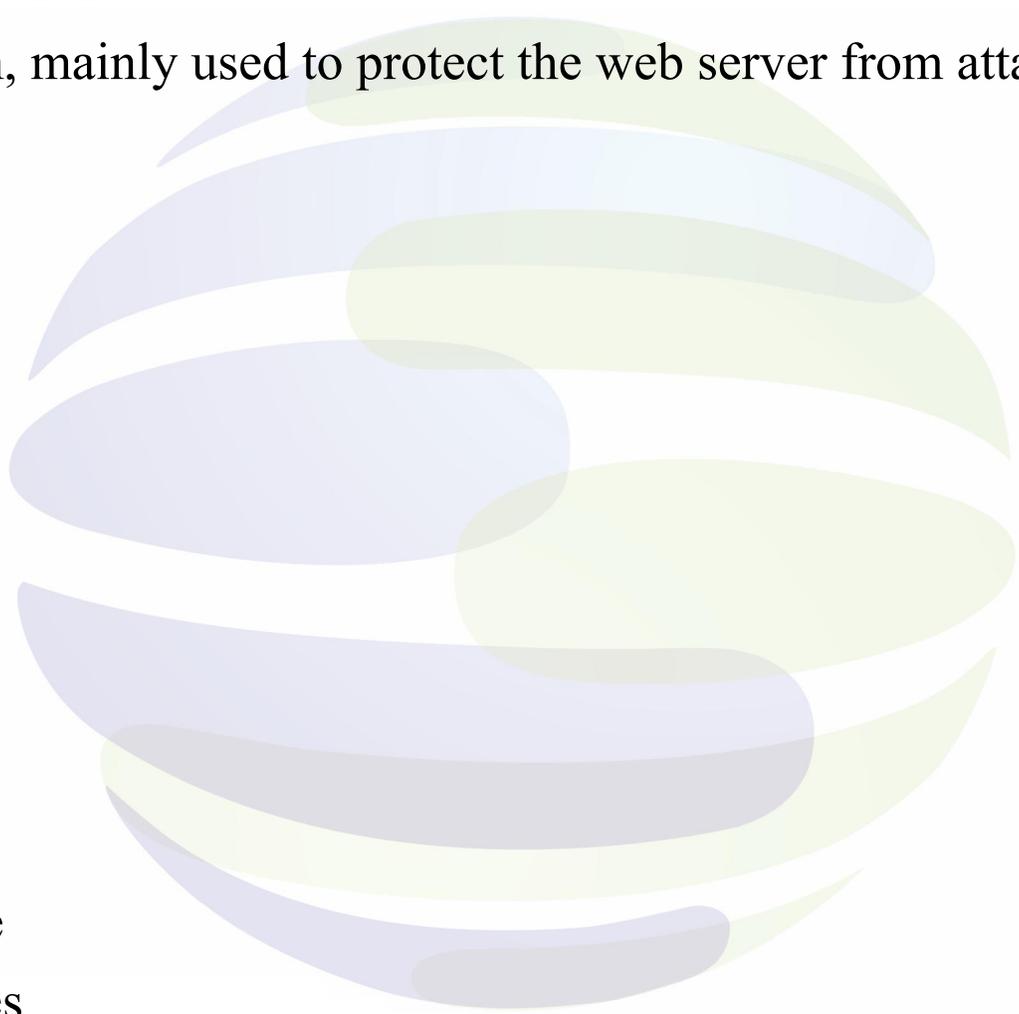
OWASP Top 10 – 2013 (Previous)	OWASP Top 10 – 2017 (New)
A1 – Injection	A1 – Injection
A2 – Broken Authentication and Session Management	A2 – Broken Authentication and Session Management
A3 – Cross-Site Scripting (XSS)	A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References - Merged with A7	A4 – Broken Access Control (Original category in 2003/2004)
A5 – Security Misconfiguration	A5 – Security Misconfiguration
A6 – Sensitive Data Exposure	A6 – Sensitive Data Exposure
A7 – Missing Function Level Access Control - Merged with A4	A7 – Insufficient Attack Protection (NEW)
A8 – Cross-Site Request Forgery (CSRF)	A8 – Cross-Site Request Forgery (CSRF)
A9 – Using Components with Known Vulnerabilities	A9 – Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards - Dropped	A10 – Underprotected APIs (NEW)

Common Web Application Attack

Web application protection, mainly used to protect the web server from attack. The common attack is as below:

1. SQL Injection
2. XSS Attack
3. Trojan
4. Website Scan
5. WebShell
6. CSRF
7. OS Command Injection
8. File Inclusion
9. Path Traversal
10. Information Disclosure
11. Web Site Vulnerabilities

So on...



2. SQL Injection



SQL Injection

SQL Injection refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server

SQL injection is actually in the web page to perform some SQL statements to operate the database, for its attack characteristics have a preliminary understanding, to further understanding, need to figure out the following questions:

1. Where does the attack data appear (where to submit, how to submit)
2. What is the SQL injection attack (what content is considered to be SQL injection attack)

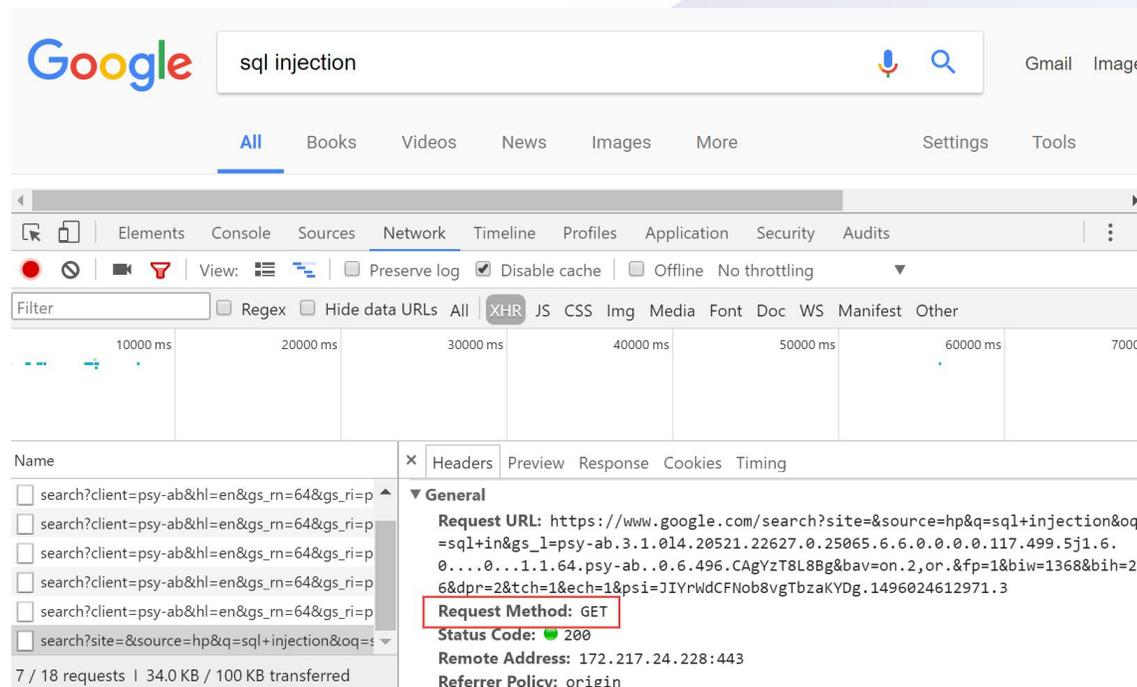
SQL Injection

Where does the attack data appear?

Two commonly used methods for a request-response between a client and server are: GET and POST.

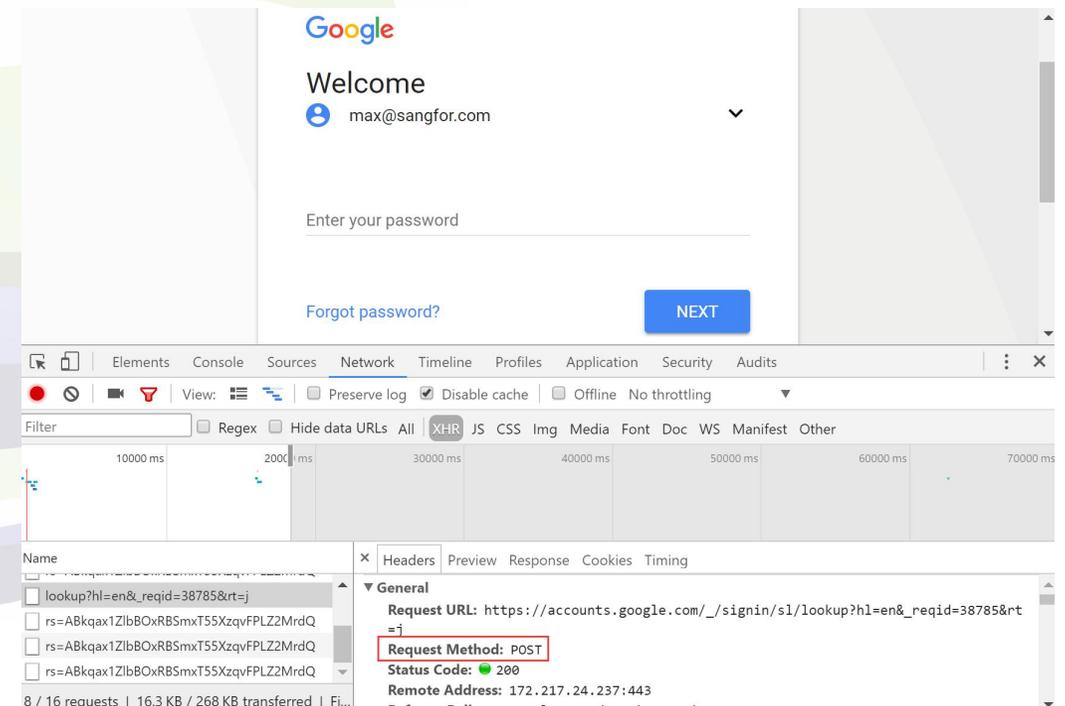
- **GET** - Requests data from a specified resource
- **POST** - Submits data to be processed to a specified resource

GET



The screenshot shows a Google search for "sql injection". The Network tab is open, displaying a list of requests. The selected request is a GET request to the URL: `https://www.google.com/search?site=&source=hp&q=sql+injection&oq=sql+in&gs_l=psy-ab.3.1.014.20521.22627.0.25065.6.6.0.0.0.117.499.5j1.6.0...0...1.1.64.psy-ab..0.6.496.CAGYzT8L8Bg&bav=on.2,or.&fp=1&biw=1368&bih=296&dpr=2&tch=1&ech=1&psi=JIYrWdCFNob8vgtBzaKYDg.1496024612971.3`. The request method is GET, the status code is 200, and the remote address is 172.217.24.228:443.

POST



The screenshot shows a Google account login page. The Network tab is open, displaying a list of requests. The selected request is a POST request to the URL: `https://accounts.google.com/_/signin/s1/lookup?hl=en&reqid=38785&rt=j`. The request method is POST, the status code is 200, and the remote address is 172.217.24.237:443.

SQL Injection

What is the SQL injection attack

SQL injection attacks can be classified according to their characteristics are weak features, strong features, injection tool features.

Weak attack: like “select * from test”, in this SQL statement, there are two key words: “select” and “from” perform a query, the risk is relatively low and strong attack;

Strong attack: such as “insert into test values (Max, 123)”, There are three SQL keywords: “insert”, “into”, “values” in this statement, and this statement may cause the user to add Max to the test table. This statement is considered dangerous. ;

Strong attack generally has the following characteristics:

1. contains three or more SQL keywords, and these three keywords together can become a legitimate SQL statement.
2. contains any SQL keyword conjunctions, these conjunctions include “union”, “;”, “and”, “or”, etc., and take the commonly used sql injection method to use these conjunctions, such as the existence of the packet “and 1 = 1” will be considered Strong features.

Injection tool attack: the use of a number of professional SQL injection tool to attack, these tools are attacks with fixed data flow characteristics.

SQL Injection

SQL injection test methods, generally can be divided into two types:

1. In the customer's real environment for scanning analysis to find the injection point and invasion,
The first approach requires the tester to have a high level of technical analysis and may affect the customer's actual business, the test success rate is low, but also the most convincing.
2. Build a virtual environment and a demonstration.
The second method is simple and easy to demonstrate, and can demonstrate the actual attack effect, the test success rate is higher, mainly play a demonstration effect.

SQL Injection

Take the virtual environment test as an example:

1. Build a test environment;
2. Get ready for testing tools or manually attack;
3. Configure NGAF WAF policy;
4. Check the NGAF WAF log.

SQL Injection

Test diagram:



Configure the basic network settings.

SQL Injection

We use the WebGoat as the attack object.



Thank you for using WebGoat! This program is a demonstration of common web application flaws. The exercises are intended to provide hands on experience with application penetration testing techniques.

The WebGoat project is led by Bruce Mayhew. Please send all comments to Bruce at WebGoat@owasp.org.



WebGoat Authors

Bruce Mayhew
Jeff Williams

WebGoat Design Team

David Anderson
Laurence Casey (Graphics)
Rogan Dawes
Bruce Mayhew

V5.4 Lesson Contributors

Sherif Koussa
Yiannis Pavlosoglou

Special Thanks for V5.4

Brian Ciomei (Multitude of bug fixes)
To all who have sent comments

Documentation Contributors

Erwin Geirnaert
Aung Khant
Sherif Koussa

Start WebGoat

You can learn more about WebGoat from

https://www.owasp.org/index.php/Category:OWASP_WebGoat_Project

SQL Injection

String SQL injection attack without WAF

 OWASP WebGoat v5.4

[Hints](#)
[Show Params](#)
[Show Cookies](#)
[Lesson Plan](#)
[Show Java](#)
[Solution](#)

Introduction

General

Access Control Flaws

AJAX Security

Authentication Flaws

Buffer Overflows

Code Quality

Concurrency

Cross-Site Scripting (XSS)

Improper Error Handling

Injection Flaws

Command Injection

Numeric SQL Injection

Log Spoofing

XPATH Injection

String SQL Injection

LAB: SQL Injection

Stage 1: String SQL Injection

Stage 2: Parameterized Query #1

Stage 3: Numeric SQL Injection

Stage 4: Parameterized Query #2

Modify Data with SQL Injection

Add Data with SQL Injection

Solution Videos
[Restart this Lesson](#)

SQL injection attacks represent a serious threat to any database-driven site. The methods behind an attack are easy to learn and the damage system compromise. Despite these risks, systems are susceptible to this form of attack.

Not only is it a threat easily instigated, but it can be easily prevented.

It is always good practice to sanitize all user inputs, and database queries, even if in a simple manner.

General Goal(s):

The form below allows a user to view their credit card numbers in all the credit card numbers by last name.

Enter your last name:

`SELECT * FROM user_data WHERE last_name = 'sangfor'`

No results matched. Try Again.

OWASP Foundation | Project WebGoat

Enter your last name:

`SELECT * FROM user_data WHERE last_name = 'sangfor' or '1' = '1'`

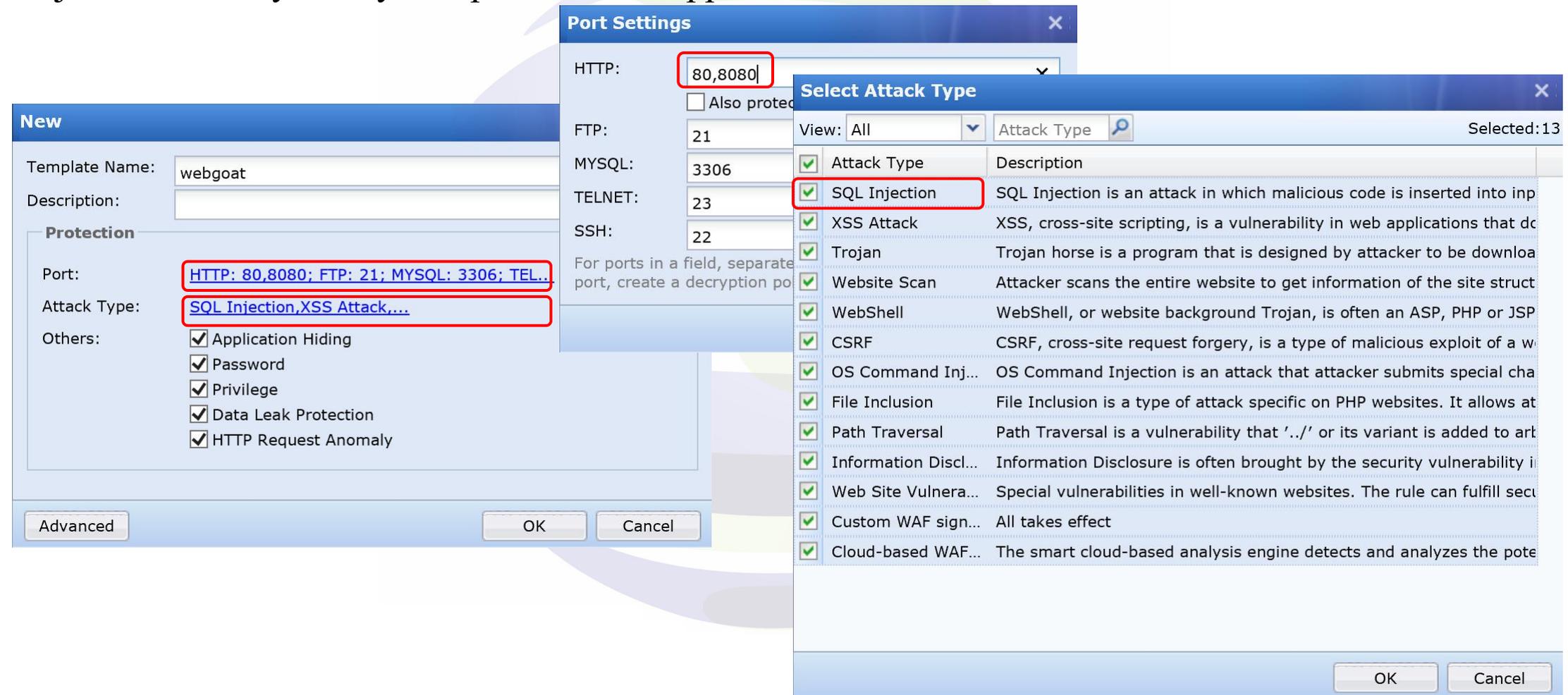
USERID	FIRST_NAME	LAST_NAME	CC_NUMBER	CC_TYPE	COOKIE	LOGIN_COUNT
101	Joe	Snow	987654321	VISA		0
101	Joe	Snow	2234200065411	MC		0
102	John	Smith	2435600002222	MC		0
102	John	Smith	4352209902222	AMEX		0
103	Jane	Plane	123456789	MC		0
103	Jane	Plane	333498703333	AMEX		0
10312	Jolly	Hershey	176896789	MC		0
10312	Jolly	Hershey	333300003333	AMEX		0
10323	Grumpy	youaretheweakestlink	673834489	MC		0
10323	Grumpy	youaretheweakestlink	33413003333	AMEX		0
15603	Peter	Sand	123609789	MC		0
15603	Peter	Sand	338893453333	AMEX		0
15613	Joeph	Something	33843453533	AMEX		0

All data is exposed.

SQL Injection

Create a template.

Objects > Security Policy Template > Web App Protection



The screenshot displays the configuration interface for a Security Policy Template. The main window is titled "New" and shows the following details:

- Template Name:** webgoat
- Description:** (empty)
- Protection:**
 - Port:** HTTP: 80,8080; FTP: 21; MYSQL: 3306; TELNET: 23
 - Attack Type:** SQL Injection, XSS Attack, ...
 - Others:**
 - Application Hiding
 - Password
 - Privilege
 - Data Leak Protection
 - HTTP Request Anomaly

Two dialog boxes are overlaid on the main window:

- Port Settings:** A dialog box with a list of protocols and their corresponding ports:

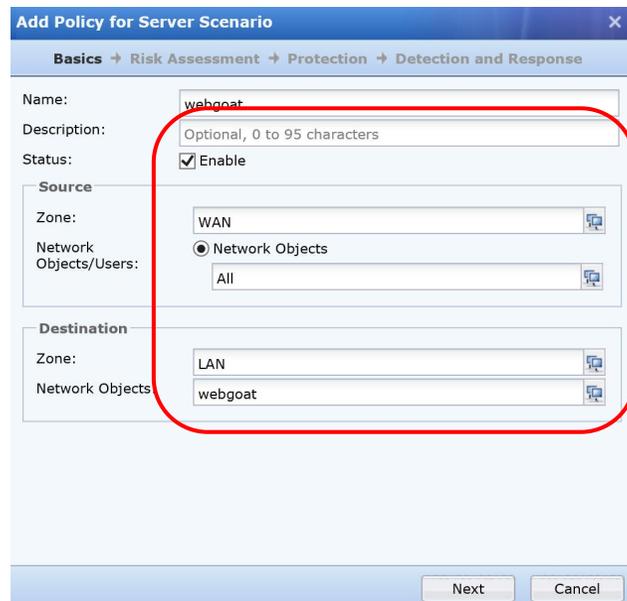
HTTP:	80,8080
FTP:	21
MYSQL:	3306
TELNET:	23
SSH:	22
- Select Attack Type:** A dialog box showing a list of attack types with checkboxes. The "SQL Injection" checkbox is highlighted with a red box.

Attack Type	Description
<input checked="" type="checkbox"/> Attack Type	
<input checked="" type="checkbox"/> SQL Injection	SQL Injection is an attack in which malicious code is inserted into input fields.
<input checked="" type="checkbox"/> XSS Attack	XSS, cross-site scripting, is a vulnerability in web applications that allows attackers to inject malicious code into the pages viewed by other users.
<input checked="" type="checkbox"/> Trojan	Trojan horse is a program that is designed by attacker to be downloaded and executed by victims without their consent.
<input checked="" type="checkbox"/> Website Scan	Attacker scans the entire website to get information of the site structure.
<input checked="" type="checkbox"/> WebShell	WebShell, or website background Trojan, is often an ASP, PHP or JSP script that allows the attacker to execute arbitrary commands on the host.
<input checked="" type="checkbox"/> CSRF	CSRF, cross-site request forgery, is a type of malicious exploit of a website where an attacker tricks the website into performing an action that the user is not expecting.
<input checked="" type="checkbox"/> OS Command Inj...	OS Command Injection is an attack that attacker submits special characters to the application to execute arbitrary OS commands.
<input checked="" type="checkbox"/> File Inclusion	File Inclusion is a type of attack specific on PHP websites. It allows the attacker to include files from the server.
<input checked="" type="checkbox"/> Path Traversal	Path Traversal is a vulnerability that '..' or its variant is added to the path to access files and directories outside the webroot.
<input checked="" type="checkbox"/> Information Discl...	Information Disclosure is often brought by the security vulnerability in the application.
<input checked="" type="checkbox"/> Web Site Vulnera...	Special vulnerabilities in well-known websites. The rule can fulfill security requirements.
<input checked="" type="checkbox"/> Custom WAF sign...	All takes effect.
<input checked="" type="checkbox"/> Cloud-based WAF...	The smart cloud-based analysis engine detects and analyzes the potential threats.

SQL Injection

Create a policy of Server Scenario.

Policies > Network Security > Policies



Add Policy for Server Scenario

Basics → Risk Assessment → Protection → Detection and Response

Name: webgoat

Description: Optional, 0 to 95 characters

Status: Enable

Source

Zone: WAN

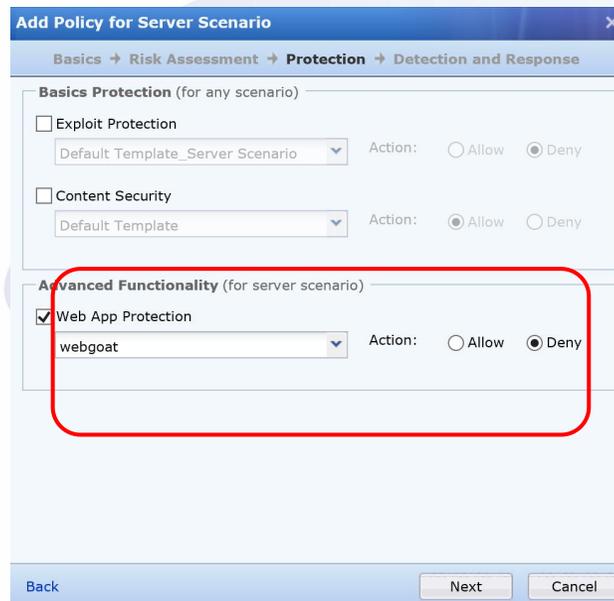
Network Objects/Users: Network Objects
All

Destination

Zone: LAN

Network Objects: webgoat

Next Cancel



Add Policy for Server Scenario

Basics → Risk Assessment → Protection → Detection and Response

Basics Protection (for any scenario)

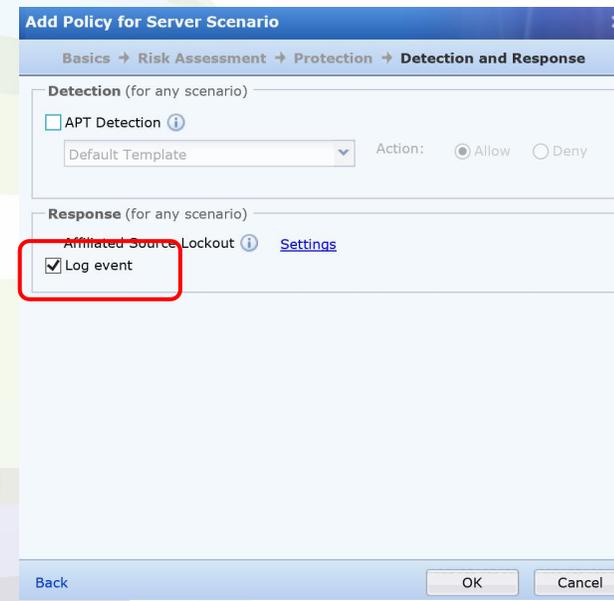
Exploit Protection
Default Template_Server Scenario Action: Allow Deny

Content Security
Default Template Action: Allow Deny

Advanced Functionality (for server scenario)

Web App Protection
webgoat Action: Allow Deny

Back Next Cancel



Add Policy for Server Scenario

Basics → Risk Assessment → Protection → Detection and Response

Detection (for any scenario)

APT Detection
Default Template Action: Allow Deny

Response (for any scenario)

Log event

Back OK Cancel

SQL Injection

If we input the sangfor' or '1' = '1, NGAF will detect it and block it and we can check the log in report center.

WAF

Filter | Export Logs

Filter: Period (2017-12-11 00:00~2017-12-11 23:59) | Src zone (All) | Src IP (All) | Dst zone (All) | Dst IP (All) | Rule ID (All) | State Code (All) | Domain/URL(All) | Type (All) | Threat level (High,Medium,Low) | /

Period (2017-12-11 00:00~2017-12-11 23:59) Src zone (All) Src IP (All) Dst zone (All) Dst IP (All) Rule ID (All) State Code (All) Domain/URL(All) Type (All) Threat level (High,Medium,Low) Action (Allow,Deny) Merge Logs(Enable)	Src Location	Dst IP	Rule ID	Description	Threat Level	Action	Details	Whitelist
United Stat...	172.17.1.10	13020036	Malicious Statemen...	High	Deny	View	Add	

Page of 1 | 2017-12-11 14:29:03 | Entry per Page: | Entry 1 - 1 of 1

1. SQL injection(High, Deny)
Previous(↑) | Next(↓)

Time: 2017-12-11 14:29:03 | Dst IP: 172.17.1.10
 Src IP: 192.200.19.83 | Dst URL: 192.168.1.105:8080/WebGoat/attack?Screen=492&menu=1100

Packet Content

T_U T_T o₁ i₀

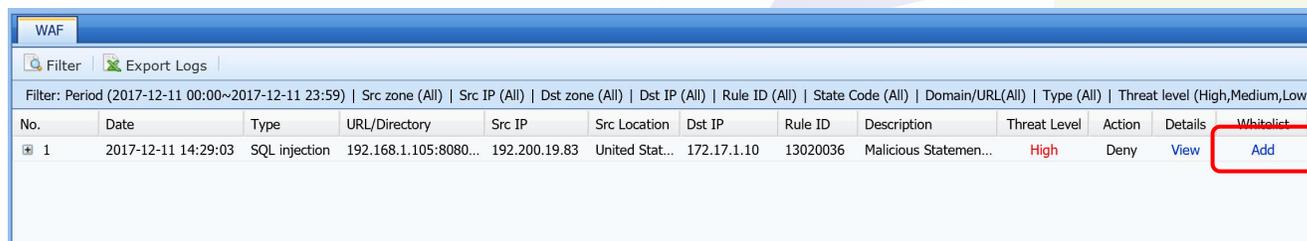
```

REQUEST:
POST /WebGoat/attack?Screen=492&menu=1100 HTTP/1.1
Accept: text/html, application/xhtml+xml, image/jxr, */*
Referer: http://192.168.1.105:8080/WebGoat/attack?Screen=492&menu=1100&Restart=492
Accept-Language: zh-Hans-CN,zh-Hans;q=0.8,en-US;q=0.5,en;q=0.3
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; Touch; rv:11.0) like Gecko
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
Host: 192.168.1.105:8080
Content-Length: 56
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: JSESSIONID=01BEBB525D9A48D81A969039A5D9D37F; td_cookie=18446744070506634699; td_cookie=18446744070013083518; PHPSESSID=nahg6325sfig2e3d0vlkqrl8r4
Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=

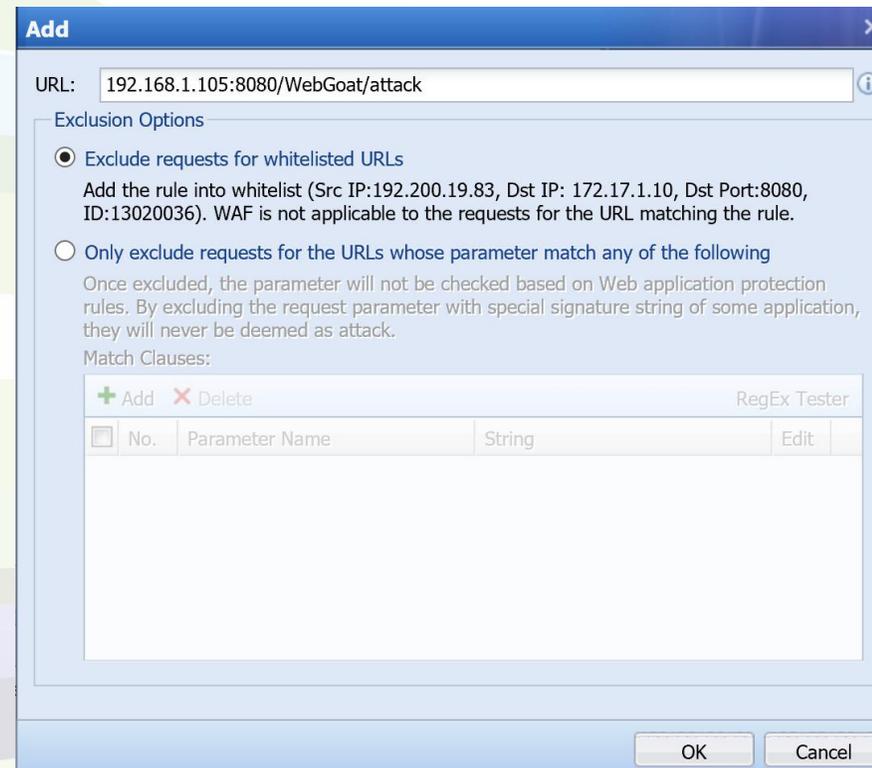
account_name=sangfor' or '1' = '1&SUBMIT=Go!
          
```

SQL Injection

How to add whitelist if there is a misjudgment?



No.	Date	Type	URL/Directory	Src IP	Src Location	Dst IP	Rule ID	Description	Threat Level	Action	Details	Whitelist
1	2017-12-11 14:29:03	SQL injection	192.168.1.105:8080...	192.200.19.83	United Stat...	172.17.1.10	13020036	Malicious Statemen...	High	Deny	View	Add



Add

URL: 192.168.1.105:8080/WebGoat/attack

Exclusion Options

Exclude requests for whitelisted URLs
Add the rule into whitelist (Src IP:192.200.19.83, Dst IP: 172.17.1.10, Dst Port:8080, ID:13020036). WAF is not applicable to the requests for the URL matching the rule.

Only exclude requests for the URLs whose parameter match any of the following
Once excluded, the parameter will not be checked based on Web application protection rules. By excluding the request parameter with special signature string of some application, they will never be deemed as attack.

Match Clauses:

+ Add		- Delete		RegEx Tester
No.	Parameter Name	String	Edit	

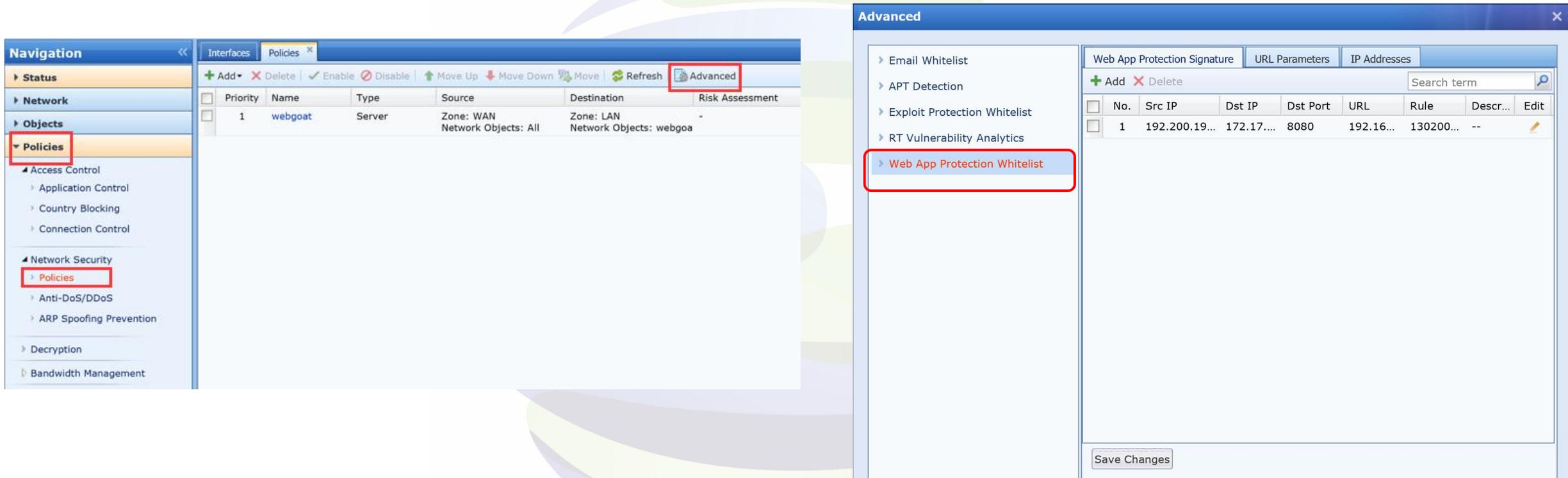
OK Cancel

3. Whitelist



Whitelist

We can check the whitelist added.



The screenshot displays the Sangfor management console interface. On the left, the 'Navigation' pane shows the 'Policies' section under 'Network Security' highlighted. The main area shows a table of policies with one entry:

Priority	Name	Type	Source	Destination	Risk Assessment
1	webgoat	Server	Zone: WAN Network Objects: All	Zone: LAN Network Objects: webgoa	-

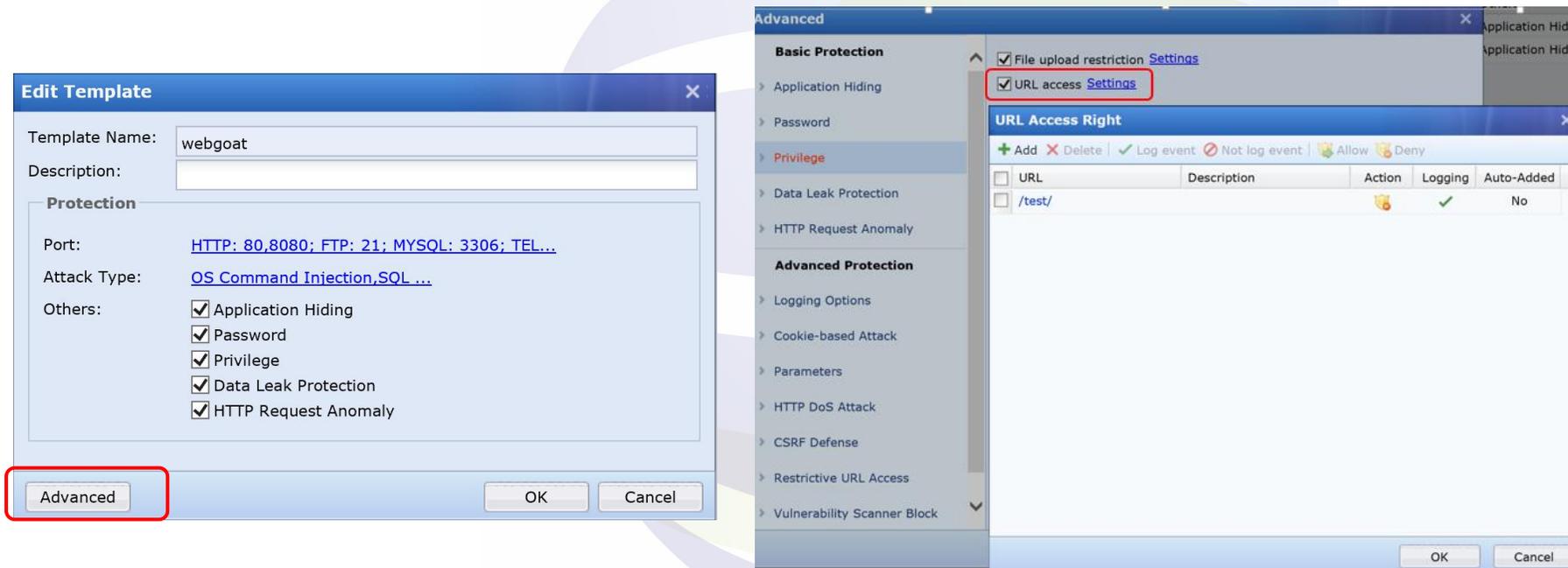
The 'Advanced' configuration window is open, showing the 'Web App Protection Whitelist' section. The 'IP Addresses' tab is active, displaying a table of whitelisted IP addresses:

No.	Src IP	Dst IP	Dst Port	URL	Rule	Descr...	Edit
1	192.200.19...	172.17....	8080	192.16...	130200...	--	

We can add manually here by these 3 ways.

Whitelist

We also can add the URL to **URL access**, NGAF will not detect the URL added as allow and log the logs in the report center as **URL access**.



The image shows two overlapping windows from the NGAF configuration interface. The 'Edit Template' window on the left is for a template named 'webgoat'. It has a 'Protection' section with several checked options: Application Hiding, Password, Privilege, Data Leak Protection, and HTTP Request Anomaly. The 'Advanced' button at the bottom left of this window is highlighted with a red box. The 'Advanced' window on the right shows a list of protection features. Under 'Basic Protection', 'URL access Settings' is checked and highlighted with a red box. Below this, the 'URL Access Right' window is open, showing a table with one entry: a URL of '/test/' with an 'Allow' action, logging enabled, and not auto-added.

URL	Description	Action	Logging	Auto-Added
<input type="checkbox"/>	/test/	Allow	✓	No

Thank you !

tech.support@sangfor.com

community.sangfor.com

Sangfor Technologies (Headquarters)

Block A1, Nanshan iPark, No.1001

Xueyuan Road, Nanshan District,

Shenzhen, Guangdong Province,

P. R. China (518055)

