**PHP**

**ლექცია 15. სესია, უსაფრთხოება**

**წვდომის ავტორიზაცია**

ინტერნეტ ქსელში არსებობს სხვადასხვა სახის მრავალი რესურსი. ამ რესურსებზე წვდომა სხვადასხვა სახისაა - ზოგჯერ, რომელიმე კონკრეტული რესურსი შეიძლება მიღებულ იქნეს თავისუფლად, ყოველივე წინაპირობის გარეშე. ასეთ რესურს თავისუფალი რესურსი ეწოდე­ბა. ხოლო მისი მიპივებას თავისუფალი წვდომა.

განსხვავებით თავისუფალი რესურსისა, არსებობენ ისეთი რესურსები, რომელთა გამოყე­ნება რომელიღაც წინაპირობითაა შესაძლებელი. ასეთ რესურსებს ჰყავთ მომსახურე პირები ან ჯგუფი, რომელთაც **ადმინისტრატორები** ჰქვია. ისინი ძირითადად რესურსის შიგთავსზე და მომხმარებლისთვის ამ რესურსის მიწოდებაზე არიან პასუხისმგებლები. ადმინისტრატირის ძირითადი დანიშნულებაა რესურსში ახალი ინფორმაციის დამატება, არსებული ინფორმაციის რედაქტირება, წაშლა; მომხმარებლისთვის ინფორმაციის მიწოდე­ბის საშულებების გაწყობა. **მომხმარებლის** ფუნქციას წარმოადგენს მხოლოდ ინფორმაციის მოძიება და წაკითხვა-დათვალიერება.

რადგანაც წვდომა რესურსებთან ინტერნეტის საშუალებით ხდება, ამდენად განსხვავება ადმინისტრატორისა და მომხმარებლისათვის უნდა მოხდეს სპეციფიკური საშუალებებით, რამეთუ მომხმარებელს არ ჰქონდეს საშუალება იმ რესურსებზე წვდომაზე, რომელიც დახუ­რულია და მხოლოდ ადმინისტრატორის უფლებამოსილებას წარმოადგენს. კერძოდ, როგო­რი უნდა იყოს რეაგირება იმ მოთხოვნაზე, რომელიც რომელიღაც კლიენტმა გაგზავნა საიტ­ზე - ამისთვის კი უნდა მოხდეს კლიენტის იდენტიფიცირება (მაგალითად განვასხვავოთ ადმინისტრატორია თუ მომხმარებელი) და მის შესახებ ინფორმაცია შენახულ იქნეს მანამ ის საიტზე იქნება.

იდენტიფიკაციის ერთ-ერთი საშუალებაა კლიენტის რეგისტრირება: კლიენტს მიენიჭება სახელი (ლოგინი) და პაროლი, ისინი შეინახება სერვერზე. ცხადია, რომ ადმინისტრატორის­თვის და მომხმარებლისთვის სხვადასხვა დონის რეგისტრაციაა გასავლელი - ანუ სისტემა გაარჩევს, თუ ვინ, როგორი უფლებამოსილებით შემოდის სისტემაში. შემდეგ, როცა რესურ­სის წვდომის მოთხოვნა შემოდის სერვერზე ხდება რეგისტრაციის შემოწმება, ანუ ინფორ­მაციის - ლოგინისა და პაროლი. თუ კი ისინი ემთხვევა სერვერზე შენახულ მონაცემებს, მა­შინ ხდება რესურსზე წვდომის უფლების მიცემა.

განვიხილოთ მარტივი სახის რეგისტრაცია კონკრეტულ მაგალითზე. ვთქვათ მოცემულია მარტივი ჰოუმ-ფეიჯი index.html

ლისტინგი 1. პერსონალირი გვერდი

<!DOCTYPE html>

<head>

<title>My home page</title>

</head>

<body>

გამარჯობა!

მე გიგო ვარ და ეს ჩემი პერსონალური გვერდია!

<a href="secret\_info.html">აქ კი ინფორმაციაა ნიკოსთვის</a>

</body>

</html>

გვერდიდან - index.html ხდება გადასცლა იმ გვერდზე, სადაც ნიკოსთვის უნდა იყოს კონფიდენციალური ინფორმაცია

ლისტინგი 2 secret\_info.html

<!DOCTYPE html>

<head><title>Secret info</title></head>

<body>

<p><b>აქ მე ჩემი მეგობარი ნიკოსთვის

მაქვს <i>კონფიდენციალური </i>ინფორმაცია</b>

</p>

</body>

</html>

თუ კი საიტის გვერდებს დავტოვებთ ისე, როგორცაა მოცემული ლისტინგ 1და 2-ში, მაშინ ნებისმიერ პირს შეუძლია დააწკაპუნის index.html გვერდის "აქ კი ინფორმაციაა ნიკოს­თვის” ლინკს და გადავიდეს secret\_info.html გვერდზე, რომელზეც მხოლოს ნიკოსთვისაა გათვალისწონებული. ეს კი რომ არ მოხდეს, ამისთვის საჭიროა, რომ "აქ კი ინფორმაციაა ნიკოსთვის” ლინკის დაწკაპუნების შემდეგ გადავიდეთ, რომელიმე შუალედურ სკრიპტზე, რომელიც გაარკვევს შეიძლება თუ არა ამ გადასვლის განხორციელება secret\_info.html გვერ­დზე. ანუ უნდა მოხდეს რეგისტრაციის მონაცემების შემოწმება. ამდენად index.html გვერ­დის კოდი შეიცვლება, ანუ ჯერ შუალედურ სკრიპტზე უნდა გადავიდეთ.

ლისტინგი 3 პერსონალირი გვერდი - ახალი ვარიანტი

<!DOCTYPE html>

<head><title>My home page</title></head>

<body>

გამარჯობა!

მე გიგო ვარ და ეს ჩემი პერსონალური გვერდია!

<a href="authorize.html">აქ კი ინფორმაცია ნიკოსთვის</a>

</body>

</html>

შემდეგ გადავდივართ რეგისტრაციის მონაცემების გადაგზავნაზე:

ლისტინგ 4 ფორმის მონაცემების გაგზავნა დამუშავებისთვი

<!DOCTYPE html>

<body>

<form action=handler.php method=post>

სახელი : <input type=text name=name><br>

პაროლი : <input type=password name=password><br>

<input type=submit value=გააგზავნეთ>

</form>

<body>

</html>

დამუშავება კი ხდება PHP-სკრიპტის საშუალებით:

ლისტინგ 5. შუალედური სკრიპტის კოდი

<!DOCTYPE html>

<?php

$pass = $\_POST['password']; $password=1234;

$nam = $\_POST['name']; $name="nick";

// თუ კოდები ემთხვევა გადავდივართ საიტზე

if ($pass == $password && $nam == $name) {

echo " შეყვანილია სწორი მონაცემები <br>";

}

else echo "შეყვანილია არასწორი მონაცემები";

?>

<a href="secret\_info.html">აქ კი ინფორმაცია ნიკოსთვის</a>

</html>

ლისტინგ 3-5 -ის შესრულებით ვღებულობთ თანამიმდევრულად:

გამარჯობა! მე გიგო ვარ და ეს ჩემი პერსონალური გვერდია! [აქ კი ინფორმაცია ნიკოსთვის](http://localhost/laboratoriuli_15/lab_15_2_registracia_new/authorize.html)

სახელი :   
პაროლი : 

შეყვანილია სწორი მონაცემები   
[აქ კი ინფორმაცია ნიკოსთვის](http://localhost/laboratoriuli_15/lab_15_2_registracia_new/secret_info.html)

**აქ მე ჩემი მეგობარი ნიკოსთვის მაქვს *კონფიდენციალური* ინფორმაცია**

სურათი 1**.** ლისტინგ 3-5 -ის შესრულების შედეგები

ასე მარტივად ხდება, რეგისტრაციის მონაცემების შემოწმება და კონფიდენციალობის დაცვა. მაგრამ თუ კი ბევრია საიდუმლო გვერდები და მათ შორის არსებობს ინტენსიური კავშირები, ყოველთვის შეყვანა ერთი და იგივე რეგისტრაციის მონაცემების მოუხერხებელია. შესაძლებელის ყოველ ამ საიდუმლო გვერდზრ იყოს სკრიპტი, რომელიც ერთი გვერდიდან გადასცემს მეორეში რეგისტრაციის მონაცემებს. მაგრამ ეს ხერხი მოწყვლადია, რადგანაც დიდია ალბათობა ამ საიდუმლო მონაცემების მიტაცების ან შეცვლისა. PHP-ში არსებობს უფრო უსაფრთხო და მოსახერხებელი მეთოდი ამ პრობლემის მოგვარებისა - ესაა კლიენტის მონცემების შენახვა, მანამ სანამ იგი ურთიერთობს დროის უწყვეტ მონაკვეთში საიტთან - ანუ სესიის პერიოდში. ასეთ მექანიზმს - სესიის მექანიზმი ეწოდება.

**სესიის მექანიზმი**

სესია ეს არის მექანიზმი, რომელიც იძლევა იმის საშუალებას, რომ შეიქმნას და გამოყენე­ბულ იქნეს ცვლადები, რომელნიც თავის მნიშვნელობებს ინარჩუნებენ კლიენტის საიტთან ერთჯერადი უწყვეტი ურთიერთობის პროცესში. ამ ცვლადს ყოველი მომხმარებლისთვის აქვს სხვადასხვა მნიშვნელობა და შეიძლება გამოყენებულ იქნან საიტის ნებისმიერ გვერდზე, მანამ სანამ კლიენტი - მომხმარებელი არ გავა სისტემიდან. ამასთან, ყოველჯერზე, როცა მომხმარებელი შედის საიტზე ის მიიღებს ცვლადების ახალ მნიშვნელობას, რომლის საშუ­ალებით ხდება მისი იდენტიფიცირება ამ სესიის, ან საიტთან მუშაობის სესიისას. აქედანაა ზუსტად სახელწოდება - სესია.

სესიის დროს მომხმარებლის იდენტიფიკაცია ხდება შემდეგნაირად - ყოველ მომხმარე­ბელს, როცა ის საიტთან იწყებს მუშაობას ენიჭება უნიკალური ნომერი, რომელსაც სესიის იდენტიფიკატორი ქვია (SID, Session IDentifier). მისი გენერირება ხდება PHP-ის მიერ მაშინ, როცა კლიენტი შედის საიტზე და უქმდება, მაშინ, როცა ის ტოვებს საიტს. SID არის 32 სიმ­ბოლებიანი სტრიქონი (მაგ.bc3a4a45bdc803434c95dcaffb1c1811), იგი გადაეცემა სერვერს კლი­ენტის ყოველ მოთხოვნასთან ერთად, და უბრუნდება სერვერისგან პასუხთან ერთად.

არსებობს სესიის იდენტიფიკატორის გადაცემის რამდენიმე გზა:

* ერთ-ერთი საშუალებაა cookies

Cookies - ეს პატარა ფაილია, რომელსაც ქმნის სერვერი კლიენტის ჰოსტზე და იგი შეიცავს ტექსტურ ინფორმაციას კლიენტის მონაცემების შესახებ. Cookies -ს წა­იკითხავს სერვერი. Cookies -ს არსებობა განპირობებულია იმით, რომ HTML პრო­ტოკოლი ვერ განასხვავებს თუ რომელი კლიენტი დაუკავშირდა სერვერს, ხოლო როცა კავშირი კლიენტისა-სერვერთან აქტიურია, და რამდენიმე სხვადასხვა გვერ­დზე ხდება შესვლა, ეს მოუხერხებელია და აქაც საჭიროა კლიენტების განსხვავების მექანიზმის განხორციელება. სწორედ ამისთვის იქმნება cookies, როგორც HTML-ის გაფართოება და დღეს თითქმის ყველა ინტერნეტ მონაწილე: კლიენტები, პროქსი-სერვერი, და სერვერი არიან ჩართული. საინტერესოა ტერმინ cookies წარმოშობა. Cookie ინგლისურიდან ითარგმნება როგორც ”კექსი” ან ტკბილი ბონუსი, რომელ­საც რესტორანი მიართმევს ხოლმე თავის კლიენტს, რათა მან დაიმახსოვროს ეს რესტორანი და ხელმეორედ ეწვიოს. ინტერნერის შემთხვევაში კი სესიის იდენტი­ფიკატორი შე­ინახება დროებით ფაილ - cookies-ში, რომელიც ინახება იმ კლიენტის მხარეს, რო­მელმაც მოთხოვნა გაგზავნა. მაგრამ ეს ხერხი სრულყოფილი არაა, რად­განაც ბევრი მომხმარებელი, უსაფრთხოების პრინციპებიდან გამომდინარე გამორ­თავს ხოლმე cookies მხარდაჭერას თავის კომპიუტერზე.

* მეორე საშუალებაა ბრძანების სტრიქონის კოდი.

ამ შემთხვევაში სესიის იდენტიფიკატორი ავტომატურად ჩაირთვება-მიიერთდება ყველა (URL) მოთხოვნას, რომელიც სერვერს გადაეცემა და ინახება სერვერის მხა­რეს. მაგალითად მისამართი: <http://localhost:3232/department/pl/plphp/12/2.html> გადაიქცევა მისამართად <http://localhost:3232/department/pl/plphp/12/2.php?PHPSESSID=bc3a4a45bdc803434c95dcaffb1c1811> .

ბრძანების სტრიქონის კოდის გამოყენება ხდება ავტომატურად, როცა იმ ბრაუზერს, რო­მელმაც გააგზავნა მოთხოვნა, გამორთული აქვს cookies . ბრძანების სტრიქონის კოდის გამო­ყენება საკმაოდ საიმედოა იმ თვალსაზრისით, რომ მისი გამოყენება ყოველთვის შეიძლება, გამორთული cookies დროსაც. მაგრამ არაა გამორიცხული, რომ ბრძანების სტრიქონში ვინმე უცხოს მიერ იქნა დამზერილი სესიის იდენტიფიკატორი, ანაც გამოყენებული, ანაც შეცვლი­ლი. მაგრამ ასეთი პრობლემაც შეიძლება დაძლეულ იქნეს, თუ კი სწორად იქნება ორგანიზე­ბული სესიები დამახსოვრებული სესიის იდენტიფიკატორს ვერ გაამოიყენებს უცხო, ამას­თან, მერეს მხრივ, ერთი შეხედვით დამახსოვრება 32 სიმბოლოიანი სტრინგისა არც ისე ად­ვილია. ეს უსაფრთხოების საკითხებია და მათ სათადადო ყურადღება უნდა დაეთმოს.

**სესიის მოწყობა და მომართვა**

სესიებთან მუშაობის დაწყებამდე, განვიხილოთ თუ როგორ შეიძლება კორექტულად მოვმართოთ PHP ინტერპრეტატორის მიერ მათი დამუშავება. ჩუმათობის პრინციპით PHP მხრიდან სესიის მუშაობა მხარდაჭერილია იმთავითვე. ამიტომ დამატებითი ელემენტების დაყენება - ჩართვა საჭირო არაა. მაგრამ, თუ რაა ჩაწერილი ამ მოდულის მომართვაში, ამის ცოდნა აუცილებელია, რომ არ დავუშვათ შეცდომები.

PHP -ს მომართვები, მათ შორის სესიებთან მუშაობა, გაწერილია ფაილში **php.ini** და განვი­ხილოთ იგი.

სესიის იდენტიფიკატორი შეინახება ან სერვერზე, ან კლიენტის ჰოსტზე, ან იქაც და იქაც. სწორედ პარამეტრი **session.save\_path** ფაილ php.ini-ში განსაზღვრავს, თუ სად იქნება სესიის მონაცემები შენახული. მაგალითად Windows-სერვერებისთვის, ჩუმათობით, განსაზღვრუ­ლია, რომ session.save\_path-მა მიუთითოს /tmp საქაღალდე, და თუ კი ძირ კატალოგში (ჩვენს შემთხვევაში c:\Program Files\Apache Group\Apache\htdocs) ასეთი საქაღალდე არ არის, სესია შეცდომებს გამოსცემს.

სერვერს შეუძლია ერთდროულად რამდენიმე სესიის მომსახურება, და ყველა ამ სესიის დროებითი ფაილები, თავიანთი სესიის მონაცემებით, მოთავსებული იქნება იმ დირექტორი­აში, რომელიც მითითებულია session.save\_path -ში. თუ კი სისტემა დიდი ზომის საქაღალდე­ებთან ცუდად მუშაობს, მაშინ შეიძლება ქვედირექტორიის გამოყენება და ამისთვის პარამეტ­რში დირექტორიის სახელის წინ ამატებენ რიცხვა, რომელიც ქვედირექტორიის ჩართვის დო­ნეს აღნიშნავს : N;/dir. ეს ქვედირექტორია აიღება უსათუოდ ბრჭყალებში, რადგანაც წერ­ტილმძიმე არის კომენტარიების ერთ-ერთი სიმბოლო PHP -ს მომართვის ფაილში. დამოუკი­დებლად იქმნება ყველა დირექტორია და ქვედირექტორია, რომელშიც ინახება სესიის მონა­ცემები.

მაგალითად: 2;/Temp განსაზღვრავს იმას, რომ სესიის ცვლადები მოთავსებულია საქაღალდეში, რომელსა შემდეგი სახე აქვს c:\Temp\0\d\ , c:\Temp\0\b\ და ა. შ.

კლიენტის მხარეს cookies-ში შენახული მონაცემების მართვა PHP-ის საშუალებით ხდება **session.use\_cookies**, **session.cookie\_lifetime** და სხვათა პარამეტრებით.

პარამეტრი session.use\_cookies განსაზღვრავს გამოიყენება თუ არა cookies სესიებთან მუშა­ობისას. ჩუმათობის პრინციპში ის ჩართულია - ანუ მისი მნიშვნელობაა ”1”.

პარამეტრი **session.cookie\_lifetime** მიუთითებს cookies-ის ”სიცოცხლეს” წამებში. ჩუმათო­ბით იგი ტოლია ”0”-ის და ნიშნავს იმას, რომ cookies მონაცემები სწორია ბრაუზერის ფანჯა­რის დახურვამდე.

არის კიდეცვ სხვა პარამეტრები: **session.name** განსაზღვრავს სესიის სახელს, ჩუმათობის პრინციპით მისი სახელია **PHPSESSID**; **session.auto\_start** - ავტომატურად გაუშვებს სესიას; session.serialize\_handler - მოცემული სესიის კოდირების განსაზღვრა; **session.cache\_expire** - განსაზღვრავს, თუ რამდენ წამში დაძველდება დოკუმენტი კეშში.

PHPSESSID ასევე გამოიყენება როგორც ცვლადის სახელი, რომელშიც მოთავსებულია სე­სიის იდენტიფიკატორი.

სესიის ავტომატურად გაშვება გამორთულია, ჩუმათობის პრინციპით, მაგრამ მისი ჩართვა შეიძლება, თუ მნიშვნელობად ავიღებთ ”1”; სესიის კოდირებისთვის გამოიყენება php. კეშში დოკუმენტის დაძველება ხდება 180 წუთის შემდეგ.

**სესიებთან მუშაობა**

უპირველეს ყოვლისა უნდა მოხდეს სესიის მექანიზმის გაშვება. თუ კი session.auto\_start -ის მნიშვნელობა ერთია ის ავტომატურად გაიშვება, ამითომ ჩავთვალოთ, რომ მისი მნიშვნე­ლობაა ”0”, ამ შემთხვევაში, თუ კი რომელიმე სკრიპტში გვჭირდება სესიის მონაცემების გა­მოყენება, მაშინ აუცილებელია ეს, სკრიპტი დაიწყოე შემდეგი ბრძანება:

**session\_start()**;

ამ ბრძანებით სერვერი შექმნის ახალ სესიას ან აღადგენს მიმდინარეს, იმ სესიის იდენ­ტიფიკატორის მეშვეობით, რომელიც მოთხოვნითაა გადაცემული. ეს ხდება შემდეგნაირად - PHP ინტერპრეტატორი ეძებს იმ ცვლადს, რომელშიც ჩაწერილია სესიის იდენტიფიკატორი (გვახსოვს, რომ ჩუმათობის პრინციპით ესაა PHPSESSID): თავდაპირველად ინტერპრეტატო­რი ამ ცვლადს ეძებს cookies-ში, შემდეგ ცვლადებში, რომლებიც POST და GET მოთხოვნით იქნენ გადმოცემულნი. თუ იდენტიფიკატორი ნაპოვნია, მაშინ ითვლება, რომ მომხმარებელი იდენტიფიცირებულია, ხდება ყველა URL-ის შეცვლა და cookies-ის დაყენება.

**session\_start()** ბრძანება უნდა გამოძახებულ იქნეს ყველა სკრიპტში, რომელშიც უნდა იქნეს გამოყენებული სესიის ცვლადები, მანამ სანამ იქნება ბრაუზერში გამოტანილი რაიმე მონაცე­მები. ეს იმასთანაა დაკავშირებული, რომ cookies-ის დაყენება ხდებამონიტორზე ინფორმაციის გამოტანამდე.

მიმდინარე სესიის იდენტიფიკატორის მიღება შეიძლება ფუნქციის session\_id() საშუალე­ბით. თვალსაჩინოებისთვის, სესიას შეიძლება მივანიჭოთ სახელი session\_name(sesiis\_sakheli) ფუნქციის საშუალებით და ეს უნდა მოხდეს მანამ, სანამ სესია ინიცირებულია. მიმდინარე სესიის სახელის მიღება შესაძლებელია ამავე ფუნქციით, ოღონდ არ უნდა იყოს მითითებუ­ლი პარამეტრი - **session\_name()**.

ლისტინგი 6. სესიის იდენტიფიკატორისა და სესიის სახელის გამოტანა

<?

session\_start();

// ვქმნით ახალ სესიას ან

// აღვადგენთ ძველს

echo session\_id();

// გამოგვაქვს სესიის იდენტიფიკატორს

?>

<html>

<head><title>My home page</title></head>

გამარჯობა!

მე გიგო ვარ და ეს ჩემი პერსონალური გვერდია!

<a href="authorize.php">აქ კი ინფორმაცია ნიკოსთვის</a>

</html>

<?

echo session\_name();

// მიმდინარე სსესის სახელის გამოტანა.

// ამ შემთხვევაში ეს PHPSESSID -იაA

?>

შედეგად მივიღებთ, რომ სესიის იდენტიფიკატორია (ამ კონკრეტულ შემთხვევაში) -

nqttdp25sa19ccvloqkmib7t45; ხოლო სესიის სახელია PHPSESSID

თუ ასევე მოვიქცევით authorize ფაილისთვის, აქაც მივიღებთ სესიის იდენტიფიკატორსა და სესიის სახელს.

**სესიის ცვლადების რეგისტრაცია**

სესიის იდენტიფიკატორისა და სესიის სახელის გარდა დგება აუცილებლობა სხვა პარა­მეტრების (კერძოდ, საკუთარის - ლოგინი, პასვორდი) გადაცემისა. ამ მიმართულებით, პირ­ველ ნაბიჯს წარმოადგენს საკუთარი ცვლადების დარეგისტრირება:

(cvladis\_sakheli\_1, cvladis\_sakheli\_2, …..);

საკმარისია ერთხელ დარეგისტრირება ნებისმიერ გვერდზე, სადაც გამოიყენება სესია. ცვლადების სახელი გადაეცემა session\_registrer() ფუნქციას $ ნიშნის გარეშე. ასეთნაირად დარეგისტრირებული ცვლადები საიტთან მუშაობის სესიისას ხდებიან გლობალურნი (ანუ მათდამი წვდომა შეიძლება ნებისმიერი გვერდიდან).

ცვლადების დარეგისტრირების კიდევ ერთი გზა არსებობს - უნდა ჩაიწეროს ცვლადების მნიშვნელობები მასივში - $\_SESSION, ანუ ჩავწეროთ

$\_SESSION[’cvladis\_sakheli’]=’cvladis\_mnishcneloba’ ;

ამ მასივში მოთავსებულია სესიის ყველა დარეგისტრირებული - გლობალური ცვლადი და მათთან წვდომა შესაძლებელია შემდეგნაირად $\_SESSION[’cvladis\_sakheli’] ( ან $HTML\_SESSION\_VARS[’cvladis\_sakheli’] PHP 4.0.6 და უფრო ადრინდელი ვერსიებისთვის). თუ კი PHP-ის მომართვაში ჩართულია ოპცია register\_globals, მაშინ სესიის ცვლადებზე წვდომა შესაძლებელია, როგორც უბრალო ცვლადების მიმართ - ანუ $cvladis\_sakheli. ხოლო, თუ კი register\_globals=off (გამორთულია), მაშინ session\_registrer()-ის გამოყენება არ ხდება რეგისტრაციისთვის იმ ცვლადებისთვის, რომელიც POST და GET მეთოდებით არიან გადა­ცემული. ის არ მუშაობას და საერთოდ, არაა რეკომენდებული ცვლადების რეგისტრა­ციის თვის ორივე მეთოდის: $\_SESSION და session\_registrer() გამოყენება ერთდროულად.

ლისტინგი 7. ლოგინისა და პაროლის რეგისტრაცია

<?php

session\_start();

// ვქმნით ახალ სესიას ან

// აღვადგენთ მიმდინარეს

if (!isset($\_GET['go'])){

echo "<form>

Login: <input type=text name=login>

Password: <input type=password

name=passwd>

<input type=submit name=go value=Go>

</form>";

}else {

$\_SESSION['login']=$\_GET['login'];

// დავარეგისტრირეთ ცვლადი login

$\_SESSION['passwd']=$\_GET['passwd'];

// დავარეგისტრირეთ ცვლადი passwd

// ამჟამად ლოგინი და პასვორდი გლობალურები

// ცვლადებია ამ სესიისთვის

if ($\_GET['login']=="niko" &&

$\_GET['passwd']=="1234") {

Header("Location: secret\_info.php");

// გადავამისამართებთ ამ გვერდზე

// secret\_info.php

}else echo "არასწორი შესვლაა,

სცადეთ თავიდან <br>";

}

print\_r($\_SESSION);

// გამოგვყავს სესიის ყველა ცვლადი

?>

ეხლა, როცა აღმოვჩდებით secret\_info.php გვერდზე, და არა მარტო საიტის ამ გვერდზე, არამედ სხვაზეც, ჩვენ ქვეძლევა საშუალება ვისარგებლოთ - ვიმუშაოთ სესიის ცვლადებზე, ჩვენთვის საინტერესო შემთხვევაში, კი ლოგინისა და პასვორდოს მნიშვნელობებთან. ეს მონაცემები ინახება მასივში $\_SESSION. ვთქვათ საიდუმლო გვერდის კოდია:

ლისტინგი 8. საიდუმლო გვერდი

<?php

session\_start();

// ვქმნით ახალ სესიას ან

// აღვადგენთ მიმდინარეს

print\_r($\_SESSION);

// გამოგვყავს სესიის ყველა ცვლადი

?>

<html>

<head><title>Secret info</title></head>

<body>

<p>აქ მე მსურს გავუზიარი კონფიდენციალური

ინფორმაცია ჩემს მეგობარ ნიკოს.

</body>

</html>

საიდუმლო დვერდზე ჩვენ ბრაუზერი გამოგვიტანს მონიტორზე:

**Array ( [login] => niko [passwd] => 1234 )**

**აქ მე მსურს გავუზიარი კონფიდენციალური ინფორმაცია ჩემს მეგობარ ნიკოს.**

როგორც ვხედავთ, თავად secret\_info.php საიდუმლო გვერდზე, არ ხდება ისევ, უშუალოდ ლოგინისა და პასვორდის შემოწმევა, არადა დავუშვათ, რომ რომელიმე ხაკერმა გაიგო საიდუმლო გვერდის მისამართი secret\_info.php და ბრაუზერის სამისამართო სტრიქონში ჩაწერა ის. შედეგად ის პრობლემების გარეშე გახსნის საიდუმლოების შემცველ გვერდა. მაგრამ ამ მდგომარეობის გამოსწორება შეიძლება, თუ კი ბრძანების secret\_info.php -ს კოდში შევამოწმებთ სესიის იმ ცვლადებს, რომლებიც პასვორდისა და ლოგინის მნიშვნელობებს შეიცავს. ცხადია, რომ ამ შემთხვევაში ჰაკერის სესიის ცვლადების მნიშვნელობები არ დაემთხვევა წესიერი მომხმარებლის სესიის შესაბამის მონაცემებს. მაშ ასე დავამატოთ ლისტინგ 8-ს

ლისტინგი 9

if (!($\_SESSION['login']=="niko" &&

$\_SESSION['passwd']==1234))

// ვამოწმებთ წყვილის ლოგინი-პასვორდის

// სისწორეს

Header("Location: authorize.php");

// თუ შეცდომაა ვაგზავნით ისევ

// ავტორიზაციის გვერდზე

და მივიღებთ ლისტინგ 9-ს საიდუნლო გვერდის კოდს - secret\_info.php

<?php

session\_start();

// ვქმნით ახალ სესიას ან

// აღვადგენთ მიმდინარეს

print\_r($\_SESSION);

// გამოგვყავს სესიის ყველა ცვლადი

if (!($\_SESSION['login']=="niko" &&

$\_SESSION['passwd']==1234))

// ვამოწმებთ წყვილის ლოგინი-პასვოედის

// სისწორეს

Header("Location: authorize.php");

// თუ შეცდომაა ვადზავნით ისევ

// ავტორიზავიის გვერდზე

?>

<html>

<head><title>საიდუმლო იმფორმაცია</title></head>

<body>

<p>აქ მე მსურს გავუზიარი კონფიდენციალური

ინფორმაცია ჩემს მეგობარ ნიკოს.

</body>

</html>

secret\_info.php გვერდის კოდს და ამით ამ საიდუმლო გვერდის წვდომა გართულდება ჰაკერისთვის

**სესიის ცვლადების წაშლა**

სესიის ცვლადების რეგისტრაციის გარდა სასარდებლოა სესიის სცლადებისა და თავად სესიის ანულირება - წაშლა.

session\_unregistrer(cvladis\_saxeli) ფუნქცია აუქმებს მიმდინარე სესიის ცვლადს. თუ კი რეგისტრაცია მოხდა $\_SESSION ( PHP 4.0.6 -ის შემთხვევაში $HTTP\_SESSION\_VARS -ით), მაშინ გამოიყენებს ენის კონსტრუქცია unset(), რომელიც არაფერს არ აბრუნებს და მხოლოდ ანადგურებს მითითებულ ცვლადს.

განვიხილოთ მაგალითად მომხმარებლის მონაცემების ( კერძოდ ლოგინისა და პასვორ­დის) განადგურება, რადგანაც ეს აქტუალურია, რამეთუ, თუ კი ეს მონაცემები არ არის წაშ­ლილი და ბრაუზერის ფანჯარა არ იქნა დახურული, მაშინ სხვა პირს შეუძლია წაიკითხოს ეს საიდუმლო ინფორმაცია. ამ მიზნით დავამატოთ secret\_info.php გვერდს, ანუ ლისტინგ 9-ს დავუმატოთ ნაწილი

ლისტინგი 10

<?php

session\_start();

// ვქმნით ახალ სესიას ან

// აღვადგენთ მიმდინარეს

print\_r($\_SESSION);

// გამოგვყავს სესიის ყველა ცვლადი

if (!($\_SESSION['login']=="niko" &&

$\_SESSION['passwd']==1234))

// ვამოწმებთ წყვილის ლოგინი-პასვორდის

// სისწორეს

Header("Location: index.php");

// თუ შეცდომაა ვაგზავნით ისევ

// ავტორიზაციის გვერდზე

?>

<html>

<head><title>საიდუმლო ინფორმაცია</title></head>

<body>

<p>აქ მე მსურს გავუზიარი კონფიდენციალური

ინფორმაცია ჩემს მეგობარ ნიკოს.

**<a href="index1.php">მთავარ გვერდზე გადასვლა</a>**

</body>

</html>

index.php ფაილში კი უნდა გავანადგუროთ ადრე შეყვანილი ლოგინი და პაროლი.

ლისტინგი 11

<?

session\_start();

session\_unregister('passwd');

// პაროლის განადგურება

unset($\_SESSION['login']);

// ლოგინის განადგურება

print\_r($\_SESSION);

// გამოგვყავს სესიის გლობალური ცვლადევი

?>

<html>

<head><title>My home page</title></head>

... // ჰოუმ ფეიჯი

</html>

ფუნქცია session\_unset() -ს შეუძლის სესიის ყველა ცვლადის მნიშვნელობების წაშლა. მიმდინარე სესიის გაუქმება შეიძლება ბრძანებით session\_destroy() - ეს ბრძანება კი არ შლის ცვლადების მნიშვნელობებს და არ ანადგურებს cookies -ს, არამედ ანადგურებს ყველა იმ მონაცემს, რომელიც ასოცირდება მიმდინარე სესიასთან.

გაგრძელება იქნება