OWASP

The Open Web Application Security Project

May 2012

Table of Contents

Notes from the Editor

Deepak Subramanian

We thank you for the overwhelming response for the previous newsletter. The submission of articles has also improved greatly. We would however like to encourage a much higher number of submissions to the newsletter.

This is a call for papers and articles for the next quarterly issue due in July 2012.

The submission can be done as a complete article or in stages.

The preferred timeline for submission in stages is as follows:

1. Submission of abstract – 15th June 2012
2. Submission of First Draft – 30th June 2012
3. Review and submission of final draft – 20th July 2012

If you plan to submit as a complete article, the final deadline for submissions is set at 15th July 2012.

Any submission that is done beyond these deadlines will be taken into consideration for the September Newsletter.

The OWASP Newsletter aims to have many research publications and we welcome research articles with a great deal of enthusiasm.

Any suggestions and changes to making this Newsletter better are appreciated.

Email: deepak.subramanian@owasp.org

OWASP AppSec Asia 2011

Helen Gao, China AppSec 2011

The OWASP AppSec Asia 2011 was successfully held in Beijing, Nov 8-11, 2011. This four day event consists of two-day conference and two-day trainings. More than four hundred people from over ten countries attended the event. OWASP Board member Sebastien Deleersnyder kicked off the conference with a call for action to participate in OWASP and contribute to a safe computing ecosystem. The topics of discussion cover many areas of application security, including cloud security, database security, encryption, secure software development, RFID security, and mitigation of XSS and other threats. Dr. Liping Ding, Dr. Frank Fan, Cassio Goldschmidt, Tobias Gondrom, Mano Paul, Wei Zhang, Dr. Meng-Chow Kang are among the speakers and trainers. During the OWASP leader workshop led by Global Chapter Committee chairman Tin Zaw, leaders from China, Korea, Malasia and Indonasia shared their experience and ideas with those from the United States and countries from Europe and South American. This is the third year that the China chapters hosted a large scale OWASP event. For the first time, the event featured a formal product exhibition where fourteen vendors had participated. Seven media companies from inside and outside of China had covered the conference. This gathering also celebrated the tenth anniversary of OWASP, as well as the enormous growth of the OWASP network especially in Asia Pacific region. The registered members of the China chapters has increased four hundred percent last year to more than eight hundred people.

Conference Venue

Exhibition

Exhibition

Exhibition

Exhibition

Students Volunteers

Membership Committee

The main goal of 2012 is to increase membership by at least twenty percent. We will organize global OWASP membership drives. We will continue looking for ways to create incentives for people to join. In the past, we created a number of models tailored to different organizations and individuals. Over the years, some of them have become redundant. Some have proven confusing or inefficient. We will simplify Individual Supporters, clarify Local Chapter Supporters and Single Meeting Supporter. We will also define roles and responsibilities of Barter-in-Trade, University and Government Supporters.

If you think these goals are ambitious, then you are right. The Membership Committee is currently the smallest of the seven committees. If you have ever considered joining an OWASP committee, this is one in which you can make a difference. If you believe in a stable OWASP finance, if you are convinced of the importance of contributions from both individuals and organizations, and if you agree that the committee should better reflect the population of participants, then join us and make it a reality.

The Membership Committee and the Connection Committee have already had a successful meeting to look into new methods of reaching out to the mainstream and technical media. In the coming weeks, we will be contacting leaders of local chapters and other committees to create plans for Corporate Supporter membership drive and Individual Supporter membership drive.

Do you think the OWASP membership is worthy to you and your company? Have you or someone you know ever run a successful membership drive? We welcome your suggestions. The Membership Committee holds phone conference at noon EST on the 3rd Tuesday of each month. Meeting agenda will be polished to the mailing list prior to the meeting. You don't have to be a committee member to attend a meeting. To join the mailing list just go to OWASP home page and click on *Mailing List*, then select *Global_membership_committee*. Or simply go to http://tinyurl.com/OWASPMembership. You can also send an email directly to helen.gao@owasp.org.

Congratulations OWASP ZAP

Global Projects Committee Leader - Jason Li

The OWASP Newsletter in association with the Global Projects Committee congratulates the OWASP ZAP Project for winning the 2011 Toolsmith Tool of the Year Award. ZAP was featured in the November 2011 toolsmith article of the ISSA Journal. toolsmith highlights a different security-related project each month. ZAP is a great example of an OWASP Project - an open source project led by a passionate leader that helps improve the state of application security. Congrats to Simon Bennetts (aka Psiinon) on a job well done! If you haven't checked out the OWASP ZAP Project, read about it here: https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

State of Confusion
Security in State Management with ASP.NET

Tim Kulp

Imagine being handed a package and being told that it needs to go to the post office. You dutifully take the package to the post office and hand it to the clerk. The clerk looks at you and asks, "Did you package this with bubble wrap or foam peanuts?" You think for a minute…you were just handed the package, no one told you how it was packaged. You do not know anything that happened prior to the package being put in your hands. This is the world of HTTP, a stateless protocol where each request sent between the client and server are disconnected to any prior requests. HTTP does not know what happened prior to being handed the request, only that it has data to deliver to an address. As web developers we need to compensate for HTTP's stateless nature by building state management solutions. In this article, we will explore various state management solutions with ASP.NET and the security concerns of each. By the end of this article you will have a strong knowledge of the foibles of state management, how ASP.NET can be used for your solution and how to keep your state data safe.

Unique challenge/Opportunity to fail

Often developers think of state management as occurring on the client (through HTML fields, cookies, etc…) or on the server (Session variables). What they often fail to recognize is that Client-Server applications presents three environments for state management:

1. On the Client,

2. On the Server,
3. and between the Client and Server

Each of these environments presents the opportunity for security vulnerabilities to slip into your application. Attempting to build a custom state management solution that addresses all three environments is incredibly challenging and wrought with pitfalls that could snare even the most seasoned engineer. Questions like, how am I going to persist control settings between the client and server, or how can I prevent an attacker from replaying malicious state on a valid user's account can lead to a mine field of threats, counter measures and compensating controls. A common and high impact risk for application security, according to the OWASP Top Ten, is Broken Authentication and Session Management which directly involves developers trying to build custom session/state management solutions but in the end only contributes to their application's security issues (OWASP Top Ten Team, 2011). Custom session/state management solutions might address the immediate needs of an application but often are not comprehensive in covering topics such as associating session/state with authentication so that when a user logs out, their session information is disposed. In the end, with the constraints of project time and budget, custom state management solutions can cause more security issues than they resolve.

Fortunately there are numerous development frameworks with state management solutions that have been widely tested and deeply examined. ASP.NET provides a robust state management solution. Depending on your understanding, state management in ASP.NET can be complex and confusing or simple and reliable. This article will help you to understand View State and the plethora of other tools at your fingertips for remembering information about your users. As we explore each state management tool, we will examine some of the vulnerabilities it introduces as well as counter-measures to protect your system.

Client Side State Management

With Client Side State Management, the client maintains state information sending it to the server with each request. Figure 1: Sample client side state management model (Northrup & Snell, 2010, p. 121) illustrates a sample client side state management solution for a corporate reporting application. The user has subscribed to two reports and set their personalized design theme to "SPRING". This is transmitted to the server with each request to be consumed and processed.

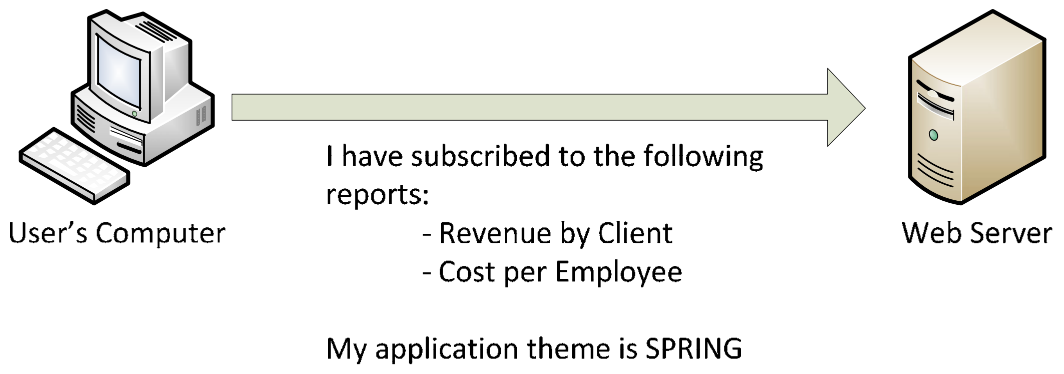

User's Computer
I have subscribed to the following reports:
- Revenue by Client
- Cost per Employee
My application theme is SPRING
Web Server

Figure 1: Sample client side state management model (Northrup & Snell, 2010, p. 121)

Using client side state management allows for scalability in your applications. As the data resides on the client system, server memory can be used for processing instead of data storage. While this solution can provide increased scalability, you pay by exposing state data to the client machine. This introduces the first attack we will explore that is common to all client side state management solutions: *Parameter Manipulation*. Malicious users can modify parameters stored on the client to abuse the trust the system has with the client (Meier, et al., 2003). Examining Figure 1: Sample client side state management model (Northrup & Snell, 2010, p. 121), consider if a malicious user modified what report they are to receive and the server system did not validate the user had permission to the resource. The malicious user would then have access to a resource that they do not have permission to, violating the confidentiality of the resource. Parameter Manipulation will be a reoccurring topic when examining the client side state management tools offered by ASP.NET. Each control offers unique concerns and solutions for this attack.

ASP.NET offers a variety of client side state management tools including:

- Query String parameters
- Hidden Fields
- Cookies
- View State
- Control State

Regardless of the tool that you use, the golden rule of Client Side State Management is to NEVER store sensitive information on the client (Meier, et al., 2003). Storing information like the font-family to use for a control is fine, but storing credit card information, personal health records, or anything else that is considered sensitive should remain on the server. On this same note, do not store information that is needed for making security decision on the client. Using Parameter Manipulation an attacker can elevate their privileges or perform unauthorized actions if the application relies solely on parameters stored on the client.

Query String Parameters

Using HTTP GET parameters, ASP.NET can pass data from the client to the server through the query string. As an example, in the following URL I can pass my product ID to the product details page:

http://some-e-com-site.com/product-details.aspx?id=9

The product-details.aspx page can then load the id parameter using the following code:

```
int prodId = Convert.ToInt32(Request.QueryString["id"]);
```

Query String parameters are ideal for small amounts of data as many browsers restrict the URL length to just under 2,100 characters (Northrup & Snell, 2010). Passing identifiers, abbreviations, etc… Another use for Query String parameters is to permit users to bookmark requests for specific requests. For the above example, a user can send the URL to a friend to directly view product id nine or a user can quickly return to this address to see the product details again.

A malicious user can easily modify query string parameters through their browser (or HTTP request transmission tool). Query String parameters are the first state management solution explored as they are the easiest example of Parameter Manipulation. When using a Query String parameter, start with the following questions:

1. What are the boundaries of this value? What is the max possible value? What is the minimum possible value?
2. What if someone passes in something I'm not expecting?

These are standard input validation questions. The first represents boundary testing. For our URL example:

http://some-e-com-site.com/product-details.aspx?id=9

What will happen if id=-1? Will the system crash, find nothing or display a message saying "id" is not a recognized value? Query String parameters make it very easy to manipulate parameters using tools like Fiddler (http://fiddler2.org) or Hackbar (an excellent Firefox extension by Johan Adriaans). Establish acceptable boundaries around data points and ensure that incoming data conforms to those boundaries. As an example, if some-e-com-site.com only has ten products (id 1 through 10) your application does not need to support an id number greater than ten or less than one.

Question #2 above is an example of Equivalence Partitioning. This testing technique groups input into blocks of data that attempts to minimize the number of tests that need to be done. As an example, one Partition could be alpha characters. If I pass in id=A, and the system crashes due to the alpha character, I do not have to test each letter of the alphabet to verify that the system will crash. Partitions can be data outside of known boundaries (such as id=11 or id=0 when we only have id 1 through 10). By entering in data for other partitions, as a security test, we want to make sure that the system maintains expected behavior and does not leak information or show a Yellow Screen of Death (YSoD).

To protect our system, we can leverage the defense in depth by passing the data through various checks prior to actually using it. Specific to ASP.NET and C# (or VB.NET) you can use the `tryparse` method to validate the provided data conforms to the data type to which the value will be cast. If the value provided can be cast as the expected data type (in this case int), the method returns true. Unlike the `parse` method, `tryparse` does not throw an exception if the value cannot be cast. Many data types support `tryparse` in C# (such as bool, DateTime, decimal, etc…) allowing developers to check data types and maintain the flow of their application without an exception. Here is a simple validation of data using `tryparse`:

```
if (Request.QueryString[«id»].Length < 3)
{
      int prodId = 0;
      if (!int.TryParse(Request.QueryString["id"], out prodId))
               displayMessageToScreen("Product Id must be a recognized value.");


     if (prodId < 1 || prodId > 10)
          displayMessageToScreen("Product Id is not in a valid range.");
}
```

This code first checks to validate that the value of Request.QueryString["id"] is no more than three characters long. This will avoid receiving values larger than a two digit number. Next we check to see if Request.QueryString["id"] can be cast as the int data type for C# (which is an Int32). If the value can be cast as an int, the value of prodId is set to be that of the Request.QueryString["id"]. If the value cannot be cast, the displayMessageToScreen method executes which shows a friendly error message to the user (avoid the YSoD and throwing errors whenever possible, throwing is expensive from a memory perspective and the YSoD makes your site appear broken). Next we confirm that the value is in the expected boundary, again displaying an error if the value is less than 1 or greater than 10. This is a simple example but conveys the idea of how to use tryparse to check whether a value can be cast and then applying simple input validation to maintain our boundaries.

Hidden Fields

A long time ago I was talking to a developer who said that their data was secured because they stored it in hidden fields. Since the user could not see them, the data was secure. Unfortunately this is extremely incorrect. The data might be safe from a non-snooping, non-malicious, non-curious user but always remember that software is a tool and people like to tinker with their tools. To this end, believing that Hidden Fields provide security is subscribing to the "Security through Obscurity" fallacy.

Hidden Fields are HTML input tags with the type of "hidden". This prevents the tag from rendering as part of the web page layout but can easily be found by viewing the source for the html page. ASP.NET Hidden fields are built using the HiddenField control, which renders as an input type="hidden" tag:

```
<asp:HiddenField ID="hdn1" runat="server" Value="Some String Value" />
```

Renders as:

```
<input type="hidden" id="hdn1" name="hdn1" value="Some String Value" />
```

The strength of Hidden Fields can be found in passing non-user friendly data back to the web page (such as a GUID) or as a data container to be used for an Ajax application that will eventually be passed back to the server in a Postback. Hidden fields can be accessed (read/write) via JavaScript making them an ideal bridge to pass data back to ASP.NET from the client for an Ajax enabled page.

Besides being aware of the parameter manipulation attack, developers need to know that hidden fields do not offer any protection for the data stored in them. The value of the field can be exposed simply by viewing the page source. Using an HTML hidden field places the responsibility on the developer to ensure the data provided is managed properly. Some developers have built very interesting client side encryption systems (jCrypt by Daniel Griesser is very interesting and easy to implement) which could be used to encrypt the data in the hidden field. You can build some regular expression checks on the data, but again, you are on your own for this.

One of the great strengths of ASP.NET are the Validation controls. Using the RequiredField control you can ensure that a value is provided, or using the CompareValidator you can ensure that the provided value conforms to parameters of other controls. Unfortunately, the ASP.NET validation controls do not work with the HiddenField control. All validation of hidden fields needs to be done manually. When dealing with hidden fields ensure that you are using proper input validation to keep the data safe while moving through the system. Avoid making security decisions based on any value in the hidden field and always replicate any security checks on the server against hidden field data.

Cookies

Cookies have been around for a long time on the web. Their implementation is almost ubiquitous on the web as sites leverage them for storing advertising campaign information, customer id, favorite color, the possibilities are as unique and endless as the many applications that spot the web. Unlike Query String parameters and Hidden Fields, Cookies can live long after the page is closed. A cookie's lifetime is defined during the creation process.

```
HttpCookie cookie = new HttpCookie("roles");
cookie.Value = "Access Maps, Access Reports, Access Reports";
cookie.Expires = DateTime.Now.AddMinutes(30);
Response.Cookies.Add(cookie);
```

In the code, line three dictates that the cookie will expire in thirty minutes from now (the current date and time). The value of the cookie can be any string value (which could be a serialized XML object, JSON object or string representation of some other data type). Being that a cookie can store anything, you are only limited by your imagination and a very small file size (4kb). Cookies are saved to the client system as a small text file that is only available to the website that created it. HTTP Headers transfer cookies from the client to the server making their data available to the server side C# as well as client side JavaScript.

Like all client side state management solutions, cookies are susceptible to parameter manipulation. As the data is stored only in a text file, unencrypted by default, the content can be viewed, manipulated and saved for the next connection. With their transmission in the HTTP header, Cookie values can be manipulated in transmission with a tool like Fiddler or OWASP's ZAP. A specific manifestation of parameter manipulation is manipulating a Role Cache. Sometimes a developer will cache a list of Roles to which the user belongs in an effort to reduce database communication for each time the system needs to authorize access. While on the surface this might seem like a great caching solution, a modification to the cookie can modify the user's Role membership. As an example, in our sample cookie above imagine changing the value of the roles cookie to be "Admin", "Administrator" or various other synonyms for administrator access. If the application recognizes one of these roles and does not validate the value, the user can elevate their privileges in the system.

Another vulnerability facing cookies is Hijacking Cookies. This is when a malicious user steals the cookies from a legitimate user. Hijacking Cookies is often the de facto illustration of a Cross Site Scripting (XSS) attack with more impact than flashing an alert message. With just a few lines of code you can return the cookie content as output to malicious code:

```
$(document).ready(function () {
$("#btn").click(function (e) {
              _stealTheCookie(document.cookie);
       });
});

function _stealTheCookie(val) {
$("#output").html(val);
}
```

While cookie theft might not matter when storing simplistic information like the user's color preferences, it is critical when the developer stores sensitive information in cookies. As mentioned previously, never store sensitive information in any client side state management solution. Even when encrypting the data, anything stored on the client's machine is in a hostile environment and open to exposure.

Finally, look again at the malicious code for _stealTheCookie. It displays the value of the cookie as HTML to the browser. We know the cookie will save to the system as:

```
roles=Access Maps, Access Reports, Access Reports
```

What if I alter the cookie.value to be:

```
cookie.Value = "<iframe src='malsite.com'></iframe>";
```

This would render an iframe to the browser and through some creative CSS can lead to a very convincing site replacement. While the user believes they are interacting with the legitimate site, they are in fact interacting with malsite.com. Again, always validate input in your application and sanitize data going out to the user.

View State

Of all the State Management solutions used in ASP.NET, View State might be the most prevalent…and least understood. By default, every ASP.NET web page carries with it a hidden field called __VIEWSTATE. This field contains a base-64 encoded value that represents the state of all the controls on the web page. View State can be very small or very large depending on the complexity of the controls on the ASP.NET page. Here is a sample of the __VIEWSTATE hidden field:

```
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE" value="/
wEPDwULLTEyMTE0MTI3OTJkZPmGU1lm0YsgU53Toeaoeeoajf92zZLCje8JDdoOSpwDe1dD/" />
```

Many developers see the value and assume a level of "encryption" due to a lack of knowledge about View State. Just like any other hidden field, __VIEWSTATE does not provide any default protection to prevent users from reading the contents. View State's purpose is to store information about web controls such as the value provided by a user between postbacks. This reduces the code a developer needs to write to repopulate all fields after a trip to the server and is more a tool of convenience than a security control. In many code samples I have seen custom controls (ascx controls or custom control libraries) using View State as a dumping ground for all types of data including entire datasets. The golden rule of client side state management applies to View State and must be remembered when building ASP.NET controls: Never store sensitive information on the client (Meier, et al., 2003). In the case where you encounter someone's code that has not followed the golden rule, ASP.NET can encrypt View State by setting the Page.ViewStateEncryptionMode to true. Using this property, ASP.NET will handle all the encryption and decryption of View State data. Like all encryption, the complexity of data can impact performance and so this option should only be used when you cannot keep sensitive data out of View State.

While data is not encrypted it is signed to ensure that View State is not modified on the client computer. By default, ASP.NET signs View State with a machine authentication code (MAC) using the MachineKey algorithm and provided in the web.config file. If the MAC value computed on the server does not match what came from the client, an exception is thrown preventing the application from processing the View State further. While the MAC can prevent modification of the View State value, it does not prevent a malicious user re-submitting the View State at a later date. This is known as a *View State Replay Attack* (Baier, 2006). By capturing a valid View State through properly using your web application, attackers can craft a malicious View State for another user. The crafted View State can then be submitted by the victim through another attack such as Cross Site Scripting (XSS [https://www.owasp.org/index.php/Top_10_2010-A2]) or Cross Site Request Forgery (CSRF [https://www.owasp.org/index.php/Top_10_2010-A5]). Using a View State Replay attack, victims can submit maliciously crafted requests to a web application that will appear as legitimate traffic. Fortunately, this is very easy to mitigate using the Page.ViewStateUserKey property. All ASP.NET pages have a property called ViewStateUserKey which allows the developer to place a unique seed into the View State that ties the value to a unique piece of data. Often, Session Id (from the Session object that we will examine further in the Server State Management section) is used to tie View State to a specific user session. Whether you use the Session Id, Username or hash of both, the key to ViewStateUserKey is to use some data that is available to the server and is unique to the user. Using the following line of code, you can assign the ViewStateUserKey to the Session Id of the user:

```
Page.ViewStateUserKey = Session.SessionID;
```

Later we will examine how Session Id can be hijacked which could lead to an attack circumventing this control. Consider the level of security your application needs when building your ViewStateUserKey. Is Session Hijacking a likely attack? Can you use another piece of data such as the username or a value in the user's Profile (we will examine Profiles later as well)?

Another common mistake that developers make is to assume View State is safe. View State should be treated as another input field just like a text field when dealing with the values and contents. While the View State MAC can provide a level of protection against malicious external users as developers we need to build our components to protect against the insider threat as well. A disgruntled developer could build malicious components that inject vulnerabilities into a web application. By simply processing the contents of View State, your web application could permit an angry developer the opportune attack vector to "get back" at their employer. Always validate input into your application and sanitize anything that is returned to the user.

Control State

View State's little brother, Control State is View State that cannot be disabled. Using Page.ViewStateEnabled, developers can turn off View State for the page or the entire application (at the configuration level). Control State is used by developers to persist values in the case that View State is disabled. Control State is stored in the __VIEWSTATE hidden field and thus, from a security perspective, is an extension of View State. By properly securing View State and validating input, Control State will be secured as well.

Server Side State Management

While the client provides numerous state management options, the server offers different options with their own unique challenges. In server side state management, (depending on the technique used) some information is provided to identify the user to the server. The server stores and manages the state information. This solution avoids the Parameter Manipulation risks that client side state management carries but has its own challenges and opportunities for your application. State can be maintained on the server with ASP.NET using the following objects:

- Application
- Session
- Profile
- View State (View State can exist on the server)

Each object carries its own security baggage usually related to access and scope. For access you need to be aware of how your application will access the various state objects. Scope refers to properly scoping the data so that you do not provide content outside of the necessary scope. We will examine each of these as they apply to the various server side state objects.

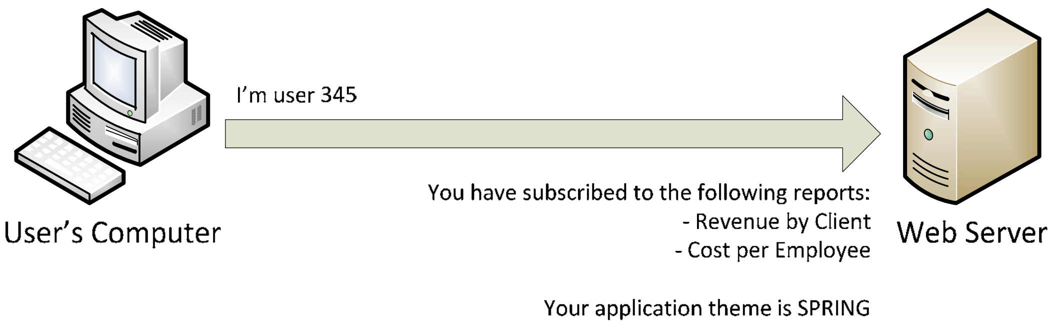

I'm user 345
User's Computer
You have subscribed to the following reports:
- Revenue by Client
- Cost per Employee
Web Server
Your application theme is SPRING

Figure 2

Application

Application is used to store information that is needed for all users of an ASP.NET application. The data is stored in memory allowing quick storage and retrieval. This object is ideal for storing small amounts of data that does not change from user to user (such as a default ID value). The Application object is a key value pair that is instantiated when the IIS Web Application starts and is lost when the application stops.

Use of the Application object should be limited to small amounts of data that are global in nature. Improperly scoping data can lead to an Information Disclosure vulnerability. In the instance where user specific information is stored in the Applications object (which it should never be because the Application object is specifically not for user specific information) it would be available to any user of the web application. While this is an example of the improper use of an object leading to a security vulnerability, it is an excellent example of how a simple mistake can lead to exposing your user's data to anyone else using the application. When using the Application object, ensure that the data being stored is not specific to a user.

Session

The Session object is a key value pair that is associated with an individual user. Unlike the Application object, Sessions are not global in scope. Users are identified by the server using a SessionID value that is passed to the client. This ID is transmitted with each request and then used by the server to determine what (if any) data is stored for the user in the Session data store. Sessions are active as long as requests are sent to the server within the specified Session lifetime. This allows Sessions to expire after a specified time period.

By default, SessionID is stored in a cookie on the client system. ASP.NET does support another option for SessionID communication between the client and server: through the URL. This transmits the SessionID as a part of the URL (using URL rewriting). By adding the following element to your web.config you can configure your ASP.NET application to pass SessionID through the URL:

```
<sessionState cookieless="true"/>
```

This will yield a URL that appears as follows:

http://localhost /**(S(tmuwrs2ubkjgnxi4ulrznncy))**/default.aspx

Notice the **tmuwrs2ubkjgnxi4ulrznncy** this is the SessionID. ASP.NET will add the session ID in to each URL that is processed by the application without any effort by the developer. This is a great solution for client systems that do not accept cookies but leads to numerous security challenges.

Once the SessionID is presented to the client computer, it is susceptible to Parameter Manipulation attacks. This is true for the cookie as well as the URL delivery method. While the SessionID is randomly generated (MSDN, 2011) and not easily guessed, it can be altered to another valid session ID that an attacker harvests through a capturing the clear text transmission of the SessionID. Using a network monitoring system like WireShark, attackers can collect SessionIDs that are transmitted in clear text over the network. After collecting the SessionID, the attacker can simply modify their SessionID to reflect that of another user exposing any sensitive data that is stored in the Session object. To avoid exposing the SessionID, avoid clear text transmission using encrypted communication like SSL/TLS.

Another security challenge opened by using cookieless Sessions is the recycling of SessionIDs. By default, if a SessionID is submitted via a URL ASP.NET will create a session with the provided ID. This could lead to two users having the same SessionID and result in an Information

Disclosure vulnerability in the system. By sending the URL with the SessionID embedded in it via email or search engine, another user can capture any data stored in another user's Session. To replicate this vulnerability, open a website that uses cookieless sessions (or create one yourself [http://seccode.blogspot.com/2012/03/cookieless-sessions-with-aspnet.html]). Copy the URL with the Session ID to another browser (this is essence should create a new Session) and load the page. Notice that any preset Session variables will follow you to the new browser.

ASP.NET Session state provides numerous challenges to maintain security in your application and while no solution can be 100% secure, we can add layers of defense to make session compromise more difficult. First and foremost: **Housekeeping**. Make sure that you abandon sessions (using the Session.Abandon() method) when the user logs out or leaves (if possible) your application. Combining the abandon method with the regenerateExpiredSessionId configuration setting will drastically reduce your exposure to session hijacking through sharing a Session ID. The regenerateExpiredSessionId attribute of the sessionState configuration element is set to true by default. This setting ensures that when a client attempts to use an old Session Id, a new one is generated instead of the supplied one. Using this setting with the practice of abandoning unused sessions reduces the time when a Session Id can be compromised to when the Session Id is in active use. While not bullet proof, this combination of configuration and best practice does reduce your possible attack surface (from a temporal point of view).

Next, consider signing the Session Id with information that is specific to individual users. By accompanying the Session Id with a machine authentication code (MAC) you can add an extra layer of verification that the user supplying the Session Id value, is the one who it was originally assigned to. In his article Foiling Session Hijacking Attempts [http://technet.microsoft.com/en-us/query/cc300500], Jeff Prosise examines adding a MAC to the Session Id composed of User Agent, network address of the user's IP and the Session Id. While noted that each of these values could be spoofed, the concept is sound in that using a MAC as another validity check on the incoming session information can make a session more difficult to hijack. In the article, Jeff uses an ASP.NET Module to capture all incoming requests to validate the supplied session MAC. If the MAC is not valid, the application logs the attempted hijack and denies access.

Finally, consider a secure transmission of Session Id values (via SSL). Exposing the Session Id through the URL or Cookie can lead to numerous hijacking vulnerabilities discussed above. By encrypting the data in transmission you can reduce the possibility of capture a Session Id in transmission.

Profile

So far all the state management solutions we have examined have been temporary. Session state expires, client side methods depend on a specified life span, and even Application state can be lost with an IIS restart. Profile provides a persistent and enduring state management tool that allows users to store state information in your application that will be waiting for them on their next visit.

Profiles are associated with individual users and stored according to the Profile Provider. By default, ASP.NET uses the SqlProfileProvider to store profile information into a Microsoft SQL Server or SQL Express database. Examination of the ASP.NET Provider Model is beyond the scope of this article but more information can be found on Microsoft's Developer Network (MSDN [http://msdn.microsoft.com/en-us/library/014bec1k.aspx]). The core concept of the Provider Model is to be able to easily manage and configure commonly reused functionality in web applications. For the SqlProfileProvider, this means defining how to store the Profile information in a SQL Server or SQL Express database. You can build your own custom Profile Provider for storing data in Oracle, SQLLite, XML or any data format that you want. This introduces the first vulnerability in ASP.NET Profiles, improperly built Providers. Ambitious developers hear about the Provider Model and instantly want to get their hands dirty in building their own. Just like Broken Authentication Schemes, a broken Provider can open numerous vulnerabilities in your system. Use the existing Providers from ASP.NET when possible to reduce the risk of an insecure, custom developed Provider. If you need to build your own, review the produced code carefully to ensure that it follows secure code best practices, avoids known attack patterns (like SQL Injection) and fails to a safe state.

Profiles use an extremely flexible implementation scheme. In the web.config file, the developer specifies what Profile fields are available by adding elements to the Profile.Properties object:

```
<profile>
   <providers>
      <clear/>
         <add …/>
   </providers>
    <properties>
       <add name="FavoriteColor" allowAnonymous="false" type="System.String" defaultValue="Blue"/>
    </properties>
</profile>
```

In this example, the Profile object will have a property name "FavoriteColor" that is a string, is "Blue" by default and is not available to Anonymous users (meaning that users must be authenticated to access this Profile Property). Assigning values to the Profile Property is as easy as:

```
Profile.FavoriteColor = TextBox1.Text;
Profile.Save();
```

This will set FavoriteColor to equal whatever was provided in the Textbox1 control's (ASP.NET Textbox) Text property. Following with the Save method will write the Profile to the data store (as defined in the Provider) for later use. While working with the Profile can be very simple, you must remember to validate data going in to it. Profile can provide an attack vector for storing malicious input for later use/display in the application. Remember to always validate user input on the server side.

Some sites use Profile data constantly to drive the application. FavoriteColor in this example might be used to define a CSS string that modifies the background of the web site to be whatever the user enters. To ease the network traffic, sometimes Profiles will be stored on the client as a Cookie. While FavoriteColor does not exposure critical information, other common Profile data points like First Name, Last Name, Date of Birth, etc… can expose the user's data on the client. When using client side storage to cache Profile data, be conscience of what data is being stored and how you are storing them. By default, Profiles stay between the server and database (rendering occurs through some other ASP.NET control). Keep sensitive data away from the client as this is a general state management best practice.

Viewstate (again?)

Yes, View State can be held on the server. Where ASP.NET stores View State is defined by inheriting an abstract class called the PageStatePersister. This class is used by a host of other classes such as HiddenFieldPageStatePersister which implement how View State is stored on the Page. The default PageStatePersister is the HiddenFieldPageStatePersister which stores View State as a hidden field on the client using base-64 encoding for the output. Other PageStatePersister include the SessionPageStatePersister that stores View State in the Session object. MSDN provides an excellent article on how to build a PageStatePersister using a System.IO.Stream [http://msdn.microsoft.com/en-us/library/system.web.ui.pagestatepersister.aspx]. Building your own Persister opens your code up to the same security challenges of building a custom Provider. False assumptions and insecure code will leave your application vulnerable and possibly, leave your View State exposed. When building a custom Persister, review the code frequently to ensure that best practices are being followed and that you do not accidently create a vulnerability while trying to move View State off the client.

As you can imagine, removing View State from the client holds numerous benefits with security and performance of the system. From a security perspective, you are removing the exposure of View State to the client which can mitigate Parameter Manipulation and View State Replay attacks. By taking the View State away from the client, you remove the control an attacker can have on the data stored in it. When considering where to store the View State information, think of scope, access and persistence. For scope, storing View State as a value in the Application object would scope the data to everyone using the application. This is clearly a bad idea as it opens a single View State to many users. How will you store View State so that user has access to it and replay attacks are not possible? Access follows a similar line of thinking in that you want to ensure that only the user has access to their View State. If you write all View State information to a text file that is available to anyone who knows the address, you are exposing your View State data to others. Does your solution only grant access to the View State for the user who owns it? Finally, you must consider how long you want to keep the View State information in your system. If you persist View State to a database is there a cleaning process that removes old View States? How will you ensure that only active View State information can be processed by your solution? Answering these questions can assist you in building a secure custom PageStatePersister object and avoid possible pitfalls in implementation that could lead to unintended Information Disclosure.

Conclusion

ASP.NET offers many state management tools, each with a purpose and intended use. With so many options developers can become confused as to when to use which. Using the wrong state management tool can make HTTP run malicious input to your server or lead to attackers stealing your user's data. Building secure state management requires an understanding of the challenges and opportunities each tool brings to your application. ASP.NET has done most of the heavy lifting for you related to state management. As a developer you must now piece these together and lock them down to keep your user's data safe.

In this article we explored numerous ASP.NET state management tools and the security concerns of each. We focused on Client Side and Server Side State Management, their strengths and weaknesses and how to layer your defenses to allow each to do their job. By applying the practices and asking the questions presented within the article, you can build a powerful and more secure state management system for your next ASP.NET application.

Works Cited

Baier, D. (2006). *Developing More-Secure Microsoft ASP.NET 2.0 Applications*. Redmond, WA: Microsoft Press.

Ballad, T., & Ballad, W. (2009). *Securing PHP Web Applications*. Boston, MA: Addison Wesley.

Hickson, I. (2011, 11 28). *Web Storage*. Retrieved from W3C: http://dev.w3.org/html5/webstorage/

Howard, M., & Lipner, S. (2003). *Writing Secure Code*. Redmond, WA: Microsoft Press.

inferno...@gmail.com. (2010, 1 30). *Issue 33876: Security: LocalStorage Cross Domain Denial of Service Attack*. Retrieved from chromium project: http://code.google.com/p/chromium/issues/detail?id=33876

Meier, J., Mackman, A., Vasireddy, S., Dunner, M., Escamilla, R., & Murukan, A. (2003). *Improving Web Application Security: Threats and Countermeasures*. Redmond, WA: Microsoft Press.

Microsoft. (2005, 1 1). *Control State vs. View State Example*. Retrieved from Microsoft Developer Network (MSDN): http://msdn.microsoft.com/en-us/library/1whwt1k7.aspx

Microsoft. (2010, 6 22). *ASP.NET Session State Overview*. Retrieved from Microsoft Developer Network: http://msdn.microsoft.com/en-us/library/ms178581.aspx

Microsoft. (2011, 9 8). *ASP.NET Cookies Overview*. Retrieved from Microsoft Developer Network: http://msdn.microsoft.com/en-us/library/ms178194.aspx

Microsoft. (2011, 9 5). *ASP.NET State Management Overview*. Retrieved from Microsoft Developer Network (MSDN): http://msdn.microsoft.com/en-us/library/75x4ha6s.aspx

Mitchell, S. (2004, 5 1). *Understanding ASP.NET View State*. Retrieved from Microsoft Developer Network (MSDN): http://msdn.microsoft.com/en-us/library/ms972976.aspx

Mozilla. (2011, 11 26). *DOM Storage*. Retrieved from Mozilla Developer Network: https://developer.mozilla.org/en/DOM/Storage

MSDN. (2011, 12 31). *HttpSessionState.SessionID Property*. Retrieved from MSDN: Microsoft Developer Network: http://msdn.microsoft.com/en-us/library/system.web.sessionstate.httpsessionstate.sessionid.aspx

Northrup, T., & Snell, M. (2010). *Web Application Development with Microsoft .NET Framework 4.0.* Redmond, WA: Microsoft Press.

OWASP Top Ten. (2010). *OWASP Top 10 - 2010.* Internet: OWASP.

OWASP Top Ten Team. (2011, 12 15). *Top 10 2010-A3-Broken Authentication and Session Management*. Retrieved from OWASP: The Open Web Application Security Project: https://www.owasp.org/index.php/Top_10_2010-A3

http://msdn.microsoft.com/en-us/library/ms178594.aspx

For all Enquiries on this article please contact:

Tim Kulp timkulp@live.com

If there were any error you would like to see rectified and/or republish the contents of this article, kindly contact:

Deepak Subramanian deepak.subramanian@owasp.org
Kate Hartmann kate.hartmann@owasp.org

Projects Reboot 2012

What is the OWASP Project ReBoot initiative?

OWASP needs to refresh, revitalize & update its projects. We need to make the software development community more aware of our efforts and demonstrate the foundations library of solutions & guidance designed to help with the secure application development lifecycle.

The proposal for this initiative is here:

Project Re-Boot Proposal

Project Lead: Eoin Keary
Proposal Approval Team: Jim Manico, Rahim Jina, Tom Brennan

To that end we have a budget to fund various project related activities. The expected outcome of this initiative is to deliver some great high quality material which can be used to support software developers and testers for years to come:

Current Submissions

OWASP Application Security Guide For CISOs

OWASP Development Guide

OWASP Testing Guide - Agreement reached. Awaiting proposal.

Zed Attack Proxy

OWASP Cheat Sheets

OWASP AppSensor

OWASP Mobile Project

Key Dates

Submission closing date: July 30th 2012

First round of proposal selection: 15 June 2012

Second round of proposal selection: 10 Aug 2012

Activity types

Type 1: Update, rewrite & complete guides or tools.
This "type" is aimed at both existing and new tools or guides which require development effort to update, augment, rewrite, develop in order to achieve a high quality release quality product.

Examples:

1. "Mini" Project based summits: Expenses associated with getting global workshops, with the aim of releasing a new version of a project.
2. Paying contributors for their time and effort.
3. Paying for user guides etc to be professionally developed (technical writing etc).

Type 2: Market, Training, Awareness, increase adoption.
Existing, healthy robust tools and guides can utilise Type 2 activities to help with creating awareness and increasing adoption of that project.

Examples:

1. Assisting with expenses associated with marketing a project.
2. Costs facilitating OWASP project focused training and awareness events

Donate to help save a current or future software application

Donate

Can I apply for this Reboot?

You certainly can, assuming you are an OWASP member.
If you feel your project is ready or has potential you can apply for the reboot programme.

How does funding work?

Type 1: Funding can be applied for as required if travel/mini summit etc is to be expensed as part of the reboot. Development activities; payment to contributors shall be at 50% and 100% milestones.

Milestones are agreed prior to project reboot initiation.

Once the 50% milestone is reached the work done to date shall be reviewed by a member of the - GPC and also another nominated OWASP reviewer (generally an OWASP leader).

Type 2: Funding is supplied as required. Items to be funded are agreed prior to reboot initiation.

Invoices for the required services are sent directly to the foundation for payment.

How do I apply? Send in a proposal with the following information:

1. Project name and description. Including reboot project lead and any team members.
2. Re boot type (Type 1 or Type 2)
3. Goals of the reboot
4. Timeline for the 50% milestone and the 100% milestone. Suggested milestone reviewers (Generally OWASP Leaders or other industry experts)
5. Budget required and how you shall spend it.

Want to support this initiative or learn more? Contact Eoin Keary

OWASP Podcast

Hosted by Jim Manico

OWASP Podcast series is hosted by Mr. Jim Manico and features a wide variety of security experts. This week we feature Troy Hunt, a Microsoft MVP involved in.Net Security. .

Podcast Link: https://www.owasp.org/download/jmanico/owasp_podcast_91.mp3

OWASP TOP 10 with Hacking-Lab

Martin Knobloch

Introduction

OWASP Global Education Committee (GEC) and Hacking-Lab have embarked in a joint educational project: Academy Portal and Hacking-Lab's remote security lab. While passive learning methods are generally acceptable to achieve lower levels of performance, but an interactive learning environment will allow the learner to achieve higher levels of performance *(i)*.

OWASP Academy Portal https://www.owasp.org/index.php/OWASP_Academy_Portal_Project

When it started…

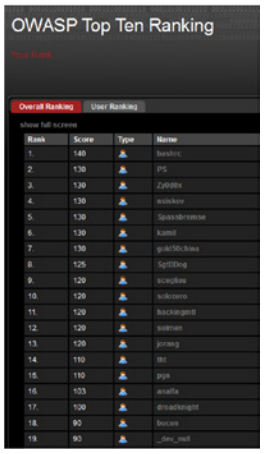
Since its launch at AppSec US in Minneapolis 2011, the portal has seen more than 6000 active global users and more than 1072 individuals have assigned to the free OWASP TOP 10 challenges.

Scoring System

Currently, the user with the nickname "bashrc" is leading the scoring of the OWASP TOP 10 event. Within the last couple of months, 167 users have successfully solved the OWASP challenges.

OWASP GEC team is checking submitted solutions day and night. This effort is driven through the support of the following key individuals: Martin Knobloch, Cecil Su, Steven van der Baan and Zaki Akhmad.

OWASP Online Competition



OWASP Top Ten Ranking

Rank	Score	Type	Name
1.	140	👤	bashrc
2.	130	👤	PS
3.	130	👤	Zy0d0x
4.	130	👤	asiskov
5.	130	👤	Spassbremse
6.	130	👤	kamil
7.	130	👤	gold50china
8.	125	👤	SgtDDog
9.	120	👤	sceplaw
10.	120	👤	solozero
11.	120	👤	hackingmtl
12.	120	👤	seimen
13.	120	👤	jorang
14.	110	👤	tht
15.	110	👤	pgs
16.	103	👤	anaifa
17.	100	👤	dreadknight
18.	90	👤	bucen
19.	90	👤	_dev_null

OWASP is planning to add additional challenges. Thanks to the Greece Hackademics project, additional challenges are now ready to be used for the planned OWASP online security competition in 2012. The winner will receive a free ticket to one of the OWASP international conferences.

WebGoat Integration

Through the efforts of volunteers, WebGoat has been integrated into the Hacking-Lab framework during the last couple of weeks. Thanks to Nicolas Hochart from Helsinki, the major work is done and we are in the quality assurance process before making them public. The addition of the Hackademics and the WebGoat projects, will introduce more than 20 new and free challenges available to everyone looking to gain some hands-on experience.

Advanced Web Security

The OWASP TOP 10 is only one important area of focus. Many additional, critical security aspects needspecial attention. In response to some recent media discussions, OWASP now has additional security challenges for the Apache Struts2 security vulnerability plus the commonly unknown XML external entity attack (XXE).

Apache Struts2 Tutorial: http://media.hacking-lab.com/movies/struts2/

LiveCD

Don't hesitate and start exploring the hands-on exercises. Joining the OWASP TOP 10 challenges is easy. Sign-Up for a Hacking-Lab account, register for the free OWASP TOP 10 challenges and get your free xUbuntu based LiveCD that provides everything you need to get started.

Sign-Up and register for FREE OWASP TOP 10 challenges https://www.hacking-lab.com/events/registerform.html?eventid=245&uk=

Download LiveCD http://media.hacking-lab.com/largefiles/livecd/

(i) http://www.nwlink.com/~donclark/hrd/strategy.html

OWASP News

Michael Coates

security101@lists.owasp.org

OWASP has created a new mailing list that is focused on bringing security information to anyone new to the security space. Have a question on a security topic? Wonder what best practices are recommended for a particular topic? Join the security101 mailing list and ask a question or help answer others!

Join at the following link: https://lists.owasp.org/mailman/listinfo/security101

Monthly Security Blitz

OWASP is starting a monthly security blitz where we will rally the security community around a particular topic. The topic may be a vulnerability, defensive design approach, technology or even a methodology. All members of the security community are encouraged to write blog posts, articles, patches to tools, videos etc in the spirit of the current monthly topic. Our goal is to show a variety of perspectives on the topic from the different perspectives of builders, breakers and defenders.

https://www.owasp.org/index.php/OWASP_Security_Blitz

OWASP Confirmed Member Linkedin Group

Curious to network with other OWASP members? Want to promote to the world that you support OWASP? If you're an OWASP member then join the confirmed member linkedin group. Note: This is a virtual badge/membership card. There aren't any resources or discussions at this linkedin group.

http://www.linkedin.com/groups?viewMembers=&gid=4342746&sik=1336166179573

https://www.owasp.org/index.php/Membership

Upcoming Events

Global AppSec Events

Global AppSec Events	Date	Location	GCC Rep	OWASP Introduction/ Keynote
Global AppSec Research 2012 (Wiki)	July 10, 2012 - July 13, 2012	Athens, Greece	John Wilander	Tom Brennan
Global AppSec North America 2012	Oct. 22, 2012 - Oct. 26, 2012	Austin, TX	Lorna Alamri	Michael Coats, Matt Tesauro, Tom Brennan, Eoin Keary
Global AppSec Latin America 2012	Q4 2012	Montevideo, Uruguay	TBD	Tom Brennan
OWASP AppSec ASIAPAC 2013	Feb. 21, 2013 - Feb. 22, 2013	Jeju	TBD	TBD

Regional and Local Events

AppSec India 2012	Regional Event	Aug. 24, 2012 - Aug. 25, 2012	India	Tom Brennan
OWASP Ireland	Regional Event	Sept. 4, 2012 - Sept. 6, 2012	Dublin, Ireland	Eoin Kearny, Tom Brennan

Partner and Promotional Events

Want to get your event listed here? Be sure to work with the Global Conferences Committee

Event	Date	Location	OWASP Participation
BHack Conference	June 14, 2012 - June 17, 2012	Belo Horizonte/MG, Brazil	TBD
Cyber Security, Cyber Warfare and Digital Forencis (CyberSec12)	June 26, 2012 - June 28, 2012	Kuala Lumpur	TBD
BlackHat USA	July 25, 2012 - July 26, 2012	Las Vegas, NV	TBD

Global Committees

Global Chapter Committee

Mission

To provide the support required by the local chapters to thrive and contribute to the overall mission and goals of OWASP.

Committee Chair: Josh Sokol

Global Conference Committee

Mission

The OWASP Global Conferences Committee (GCC) exists to coordinate and facilitate OWASP conferences and events worldwide.

Committee Chair: Mark Bristow

Global Connections Committee

Mission

To help the OWASP foundation communicate to the outside world in a unified and coherent way. We also assist with internal communication between different OWASP projects and committees.

Committee Chair: Jim Manico

Global Education Committee

Mission

Provide awareness, training and educational services to corporate, government and educational institutions on application security.

Committee Chair: Martin Knobloch

Global Industry Committee

Mission

The OWASP Global Industry Committee (GIC) shall expand awareness of and promote the inclusion of software security best practices in Industry, Government, Academia and regulatory agencies and be a voice for industry. This will be accomplished through outreach; including presentations, development of position papers and collaborative efforts with other entities.

Committee Chair: Rex Booth

Global Membership Committee

Mission

The Membership Committee recommends policies, procedures, and strategies for enhancing the membership in OWASP both numerically and qualitatively. The committee provides a written plan and recommends policies, procedures, and initiatives to assure a growing and vital membership organization.

Committee Chair: Dan Cornell

Global Projects Committee

Mission

To foster an active OWASP developer community, facilitate contributions from OWASP community members, provide support and direction for new projects, and encourage adoption of OWASP Projects by the global community at large.

Committee Chair: Jason Li

ARTICLE I - OWASP Bylaws

Section 1.01. Offices

The principal office of the Foundation in the State of Maryland, shall be located in County of Howard. The Foundation may have such other offices, either within or without the State of Maryland, as the Board of Directors may designate or as the business of the Foundation may require from time to time.

Section 1.02. Purpose

The OWASP Foundation will be the thriving global community that drives visibility and evolution in the safety and security of the world's software.

Section 1.03. Values

OPEN: Everything at OW ASP is radically transparent from our finances to our code. **INNOVATION**: OW ASP encourages and supports innovation/experiments for solutions to software security challenges. **GLOBAL**: Anyone around the world is encouraged to participate in the OWASP community. **INTEGRITY**: OWASP is an honest and truthful, vendor agnostic, global community.

Organization and Barter In Trade Supporters

acunetix
ACCUVANT LABS
Adobe
ADP
CERT ae
Akamai
amazon
AppliedTrust
ascure
ASPECT SECURITY Application Security Experts
ASTECH security
art of defence
BEST BUY
Black Hat
Cargill
cigital
CORE SECURITY TECHNOLOGIES
DENIM GROUP
DIGITAL DEFENSE INCORPORATED
DREAMLAB TECHNOLOGIES
ENGINEERING
fishnet SECURITY
FUJITSU
gemalto security to be free
GOTHAM DIGITAL SCIENCE
HARRIS CONNECT
HIGH-TECH BRIDGE INFORMATION SECURITY SOLUTIONS
HUAWEI
IBM
(ISC)²
IMPERVA
Information Builders
INTREPIDUS GROUP
IOActive COMPREHENSIVE COMPUTER SECURITY SERVICES
mozilla Firefox
mnemonic -securing your business
nixu
NetIQ
NOKIA
ORACLE
Penta SECURITY
PRAETORIAN
protiviti Independent Risk Consulting
QUALYS ON DEMAND SECURITY
Quotium
rackspace
Redspin
salesforce.com Success On Demand
SD ELEMENTS
Security Innovation
SECURITY PS
Seeker
Symantec
TENABLE Network Security
Trustwave
ups
WhiteHat SECURITY

Academic Supporters

ADELPHI UNIVERSITY

Anglia Ruskin University

AUM

CIT Institiúid Teicneolaíochta Chorcaí Cork Institute of Technology

DARTMOUTH VOX CLAMANTIS IN DESERTO 1769

DAKOTA STATE dsu UNIVERSITY

DCU

foi UNIVERSITY OF ZAGREB FACULTY OF ORGANIZATION AND INFORMATICS VARAŽDIN

KATHOLIEKE UNIVERSITEIT LEUVEN

hochschule mannheim

heig-vd Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

HOCHSCHULE FURTWANGEN UNIVERSITY HFU

IDC HERZLIYA Efi Arazi School of Computer Science

ISCTE IUL Instituto Universitário de Lisboa Lisbon University Institute

TECNOLÓGICO DE MONTERREY

isep

itb Institute of Technology Blanchardstown Institiúid Teicneolaíochta Baile Bhlainséir

NYP Nanyang Polytechnic

ORT UNIVERSIDAD ORT Uruguay

RUTGERS THE STATE UNIVERSITY OF NEW JERSEY

TRINITY COLLEGE DUBLIN

SGU

STEVENS INSTITUTE of TECHNOLOGY THE INNOVATION UNIVERSITY

TEMASEK POLYTECHNIC

UCIRVINE

NKU NORTHERN KENTUCKY UNIVERSITY

northumbria UNIVERSITY

NORTH TEXAS

THE GEORGE WASHINGTON UNIVERSITY WASHINGTON DC

NYU·poly POLYTECHNIC INSTITUTE OF NYU

TEI ΠΑΡΙΣΑΣ

FACULTAD DE INGENIERIA Universidad de Buenos Aires

UCLA

UdeMM Universidad de la Marina Mercante Tu futuro es nuestra meta.

UNIVERSITI KUALA LUMPUR

UT DALLAS

The OWASP Foundation

The Open Web Application Security Project (OWASP) is an international community of security professionals dedicated to enabling organizations to conceive, develop, acquire, operate, and maintain applications that can be trusted. All of the OWASP tools, documents, forums, and chapters are free and open to anyone interested in improving application security. We advocate approaching application security as a people, process, and technology problem because the most effective approaches to application security includes improvements in all of these areas. OWASP is a new kind of organization. OWASP is not affiliated with any technology company, although we support the informed use of commercial security technology. Our freedom from commercial pressures allows us to provide unbiased, practical, cost-effective information about application security. Similar to many open-source software projects, OWASP produces many types of materials in a collaborative, open way.

Core Values

- OPEN – Everything at OWASP is radically transparent from our finances to our code
- INNOVATION – OWASP encourages and supports innovation/experiments for solutions to software security challenges
- GLOBAL – Anyone around the world is encouraged to participate in the OWASP community
- INTEGRITY – OWASP is an honest and truthful, vendor neutral, global community

OWASP is an Open community of Application Security Professionals.
The opportunities to participate in the organization are limitless

Donate

OWASP Membership

The professional association of OWASP Foundation is a not-for-profit 501c3 charitable organization not associated with any commercial product or service. To be successful we need your support. OWASP individuals, supporting educational and commercial organizations form an application security community that works together to create articles, methodologies, documentation, tools, and technologies

A complete list of all OWASP members can be found here: https://www.owasp.org/index.php/Membership

Individual Supporter -$50 USD/year

- Underscore your awareness of web application software security
- Receive Discounts to attend OWASP Conferences
- Expand your personal network of contacts
- Obtain an owasp.org email address
- Allocate 40% of your membership dues to directly support your local chapter
- Participate in Global Elections and vote on issues that shape the direction of the community

Corporate Supporter - $5,000 USD/year

- Tax deductible donation
- Receive Discounts at OWASP Conferences to exhibit products/services
- Opportunity to post a rotating banner ad on the owasp.org website for 30 days at no additional cost ($2,500 value)
- Be recognized as a supporter by posting your company logo on the OWASP website
- Be listed as a sponsor in the quarterly newsletter distributed to over 10,000 individuals
- Have a collective voice via the Global Industry Committee
- Participate in Global Elections and vote on issues that shape the direction of the community
- Allocate 40% of your annual donation to directly support your choice of chapter and/or projects

For More information on sponsorship opportunities, contact Kelly Santalucia at Kelly.santalucia@owasp.org

JOIN NOW

OWASP Membership Categories

	Voice During Elections	Recognition on OWASP.org Website	Discounts on Conferences	Complimentary Advertising	Recognition in Newsletter	owasp.org email address	Directly Support local chapter or project
Corporate Member	X	X	X	X	X	X	X
Individual Member	X	X	X		X	X	X
Government Supporter		X	X		X		X
Academic Supporter		X	X		X	X	
Organizational Supporter		X	X	X	X		X

Other ways to Support OWASP

Local Chapter Supporter

Organizations that are not yet interested in becoming a full Corporate Member but who have a desire to direct their support in a more regional manner may prefer to become a Local Chapter Supporter. Check with your local Chapter Leader to learn more about specific price levels for Chapter Supporters. The funds donated are divided with 90% directly supporting the OWASP local chapter and 10% to the OWASP Foundation. Local chapter pages

Single Meeting Supporter

Organizations that wish to support OWASP local chapter with a 100% tax deductible donation to enable OWASP Foundation to continue the mission. The fees are set by local chapter, so contact the chapter leader of the chapter that you want to work with. Local chapter pages

Event Sponsorship

Participate in one of our Global or Regional events by sponsoring the expo or providing tangibles to the conference attendees. View Sponsorship Opportunities

Tax Deductible Donation

The OWASP Foundation is a registered 501(c)3 in the US as well as a Not for Profit entity in Europe. As a result, your direct donation is eligible to be deducted as a charitible donation. Please contact your tax advisor for complete information.

Individual Participation

With over 140 active chapters globally and hundreds of OWASP Projects and millions of great ideas waiting to become projects, it would be difficult to NOT find a way to participate. All it takes to participate is a willingness to share ideas and collaborate with the key minds in the industry. Please reach out to your local chapter leader, a current project leader, or start your own!

Newsletter Advertising:

- 1/4 page advertisement $2000
- 1/2 page advertisement $2500
- 1/2 page advertisement + either a 30 rotating banner on the OWASP site or 10 copies of the Top 10 Books $3000
- full page advertisement $5000
- Year subscription (1 newsletter every quarter with the 1/2 page advertisement posted) $9000.

Please contact Kelly.Santalucia@owasp.org or Kate.Hartmann@owasp.org for details.