****

**GÜVENLİ YAZILIM GELİŞTİRMEDE  
DOSYA YÜKLEME**

**Yazar:** Elif Bayrakdar  
**Baskı:** 21.02.2017

İçerik

Güvenli Yazılım Geliştirmede File Upload’ın Doğru Kullanım Yolları 3

Authentication ve Authorization 4

Yüklenen Dosyanın Maximum Boyutu 5

Uzantıların Kontrolü (BlackListing & WhiteListing) 6

Yüklenen Dosyada İsim Değişikliği 7

Dosya Content’ini Kontrol Etme 8

Client-Side Validation 11

IIS Request Filtering 12

# Güvenli Yazılım Geliştirmede File Upload’ın Doğru Kullanım Yolları

İşleyiş olarak son kullanıcıya oldukça yararlı ve günümüz dünyasında yaygınlığı olan uygulamalarda bile güvenlik açıklıkları çıkmaktadır. Bu durum yazılımcıların fonksiyonaliteyi arttırayım derken güvenliğe yeterince önem verememesinden kaynaklanabilmektedir. Bu açıklıklardan en fazla dikkat edilmesi gereken konulardan birisi olarak güvensiz upload’ı inceleyeceğiz.

File upload alanları istemci tarafından sunucu tarafına dosya gönderme olanağı sağlayan yapılardır. Dikkat edileceği üzere burası bir saldırgan için sunucuya açılan bir kapıdır. Bu nedenle yazılımcıların buradaki senaryoları iyi kodlaması gerekmektedir ki yüklenilen dosya üzerindeki yetersiz kontroller nedeniyle uygulamanın koştuğu sistem için güvenlik problemlerine yol açılmasın. Bu nedenlerdendir ki bir yazılımcı asla dışarıdan gelen kaynağa güvenmemelidir. Heleki bu kaynağın sunucu tarafında yorumlanıp çalıştırılabilecek bir dosya olabileceği düşünülürse.

Bu yazıda alınabilecek önlemler, best practise’ler ve tek başına yeterli olmayan kontroller ele alınacaktır.

# Authentication ve Authorization

Bir dosya yükleme alanı varsa öncelikle burayı bu uygulamaya giriş yapan ve dosya yüklemeye izni olan kullanıcılar kullanmalı. Aksi taktirde bir saldırganın uygulamaya girmeden (Authentication) ve de işlemi yapması için bir yetkiye ihtiyaç duymadan (Authorization) kolayca bu dosya yükleme alanında denemeler yapmasına ve hatta belki de başarılı olmasına neden olunabilir.

Önlem olarak projenin kendi yapısına uygun olacak şekilde bir kurgu oluşturulmalıdır. Dosya yükleme işlemini gerçekleştiren kullanıcı login durumunda mı, dosya yüklemeye veya yüklemek istediği dosyanın türünde bir dosyayı yüklemeye yetkili mi gibi kontroller yapan metodlar yazıp en azından herkesin bu önemli alana erişme imkanı ortadan kaldırılmalıdır.

# Yüklenen Dosyanın Maximum Boyutu

Yüklenen dosyalarda bir boyut sınırı olmazsa saldırgan sisteme oldukça büyük bir dosya yükleyip yer kaynaklarını tüketebilir ve sunucuyu hizmetdışı bırakabilir. Bunu engellemenin yolu yüklenmeye çalışılan dosyanın boyutunu kontrol etmektir. .Net’te yüklenen dosyanın boyutunu byte olarak döndüren bir metod vardır:

|  |
| --- |
| fileUploadControl.PostedFile.ContentLength |

Bu kod ile yüklenen dosyanın boyut kontrolünün yapılması gerekmektedir.

|  |
| --- |
| if(fileUploadControl.PostedFile.ContentLength > 0  && fileUploadControl.PostedFile.ContentLength < 1048576)  {;} |

Burada ancak 0 Byte’tan büyük, 1 MegaByte’tan küçük boyuttaki bir dosya ise if koşulunun içini gerçekleştir denmiş oluyor. İstenilen boyutun dışında bir dosya yüklenmeye çalışırsa da işlemi gerçekleştirmeyecektir.

# Uzantıların Kontrolü (BlackListing & WhiteListing)

Uygulamaya hangi uzantıya/uzantılara sahip olan dosyaların yüklenebilir olduğuna göre uzantı kontrolü yapılabilir. Bunun için blacklist veya whitelist gibi yöntemler uygulanabilir. Bir whitelist oluşturarak yüklenmesine izin verilen dosya uzantıları belirlenebilir ve bunların dışındaki dosyaların yüklenmesine izin verilmeyebilir. Ya da bir blacklist oluşturarak asla yüklenmemesi gereken dosya uzantıları kontrol edilebilir ve yüklenmesi engellenebilir.

Ancak zararlı bir içerik barındıran bir dosya için bu yöntem yeterli olmayacaktır çünkü saldırgan shell.png.aspx şeklinde bir dosya adı kullanarak bu yöntemi bypass edebilir. Örneğin png dosyası yüklemeye izin veren bir file upload alanında saldırgan içinde zararlı bir kod bulunan shell.png.aspx diye bir dosyayı sisteme yükleyebilir ve bunu çalıştırabilir.

Önlem olarak dosya adındaki ilk noktadan sonraki değil de en son noktadan sonraki uzantı kontrol edilmelidir. Bu dosyanın gerçek uzantısı aslında .aspx’tir.

ASP.NET’te dosyanın uzantısını bulan metod:

|  |
| --- |
| string extension = System.IO.Path.GetExtension(FileName); |

Bu kod dosyanın uzantısı olarak .aspx’i döndürecektir. Fakat kullanılan programlama diline göre bu işi yapan bir metod olmayabilir. O zaman da şu mantık ile dosya uzantısı bulunabilir:

|  |
| --- |
| string extension = fileUploadControl.FileName.Substring(fileUploadControl.FileName.LastIndexOf('.') + 1); |

# Yüklenen Dosyada İsim Değişikliği

Yüklenecek dosyaya yeni bir isim ve kabul edilen uzantı verilebilir. Bu yöntem XSS ve path traversal gibi dosya ismi yolu ile yapılabilecek saldırılara karşı alınabilecek bir önlem olarak kabul edilir.

# Dosya Content’ini Kontrol Etme

Her dosya için kendine özgü bir header bilgisi olur. Her dosya uzantısı için de ilk 4 byte sabittir. Bu ilk 4 byte’ı kontrol ederek de dosyaya güvenilebilir. Aşağıdaki örnek kodlar bu kontrolü yapmaktadır:

if (fileUploadControl.HasFile

&& fileUploadControl.PostedFile.ContentLength > 0

&& fileUploadControl.PostedFile.ContentLength < 1048576)

{

if (IsImage())

{

try

{

string fileName = Path.GetFileName(fileUploadControl.PostedFile.FileName);

string directoryPath = Server.MapPath("~/UploadedFiles");

string uploadPath = Path.Combine(directoryPath, fileName);

if (!Directory.Exists(directoryPath))

{

Directory.CreateDirectory(directoryPath);

}

fileUploadControl.SaveAs(uploadPath);

lblStatus.Text = "Success!";

}

catch (Exception ex)

{

lblStatus.Text = "Failure! " + ex.Message;

}

}

else

{

lblStatus.Text += "<br />This is not an image file!";

}

}

public bool IsImage()

{

string[] validFileTypes = { "JPG", "JPEG", "PNG", "TIF", "TIFF", "GIF", "BMP", "ICO" };

string ext = fileUploadControl.FileName.Substring(fileUploadControl.FileName.LastIndexOf('.') + 1).ToUpper(); //or string ext = Path.GetExtension(this.fileUploadControl.PostedFile.FileName).TrimStart('.').ToUpper();

bool isValidFileExtention = false;

for (int i = 0; i < validFileTypes.Length; i++)

{

if (ext == validFileTypes[i])

{

isValidFileExtention = true;

break;

}

}

if (!isValidFileExtention)

{

lblStatus.ForeColor = System.Drawing.Color.Red;

lblStatus.Text = string.Format("Invalid file. Please upload a file with extension {0}", string.Join(",", validFileTypes));

}

if (isValidFileExtention)

isValidFileExtention = CheckTrueImageType();

return isValidFileExtention;

}

public bool CheckTrueImageType()

{

// DICTIONARY OF ALL IMAGE FILE HEADER

Dictionary<string, byte[][]> imageHeader = new Dictionary<string, byte[][]>();

imageHeader.Add("JPG", new byte[][] { new byte[] { 0xFF, 0xD8, 0xFF, 0xE0 },

new byte[] { 0xFF, 0xD8, 0xFF, 0xE1 },

new byte[] { 0xFF, 0xD8, 0xFF, 0xE2 },

new byte[] { 0xFF, 0xD8, 0xFF, 0xE3 },

new byte[] { 0xFF, 0xD8, 0xFF, 0xE8 },

new byte[] { 0xFF, 0xD8, 0xFF, 0xDB } });

imageHeader.Add("JPEG", new byte[][] { new byte[] { 0xFF, 0xD8, 0xFF, 0xE0 },

new byte[] { 0xFF, 0xD8, 0xFF, 0xE1 },

new byte[] { 0xFF, 0xD8, 0xFF, 0xE2 },

new byte[] { 0xFF, 0xD8, 0xFF, 0xE3 },

new byte[] { 0xFF, 0xD8, 0xFF, 0xE8 },

new byte[] { 0xFF, 0xD8, 0xFF, 0xDB } });

imageHeader.Add("PNG", new byte[][] { new byte[] { 0x89, 0x50, 0x4E, 0x47 } });

imageHeader.Add("TIF", new byte[][] { new byte[] { 0x49, 0x49, 0x2A, 0x00 },

new byte[] { 0x49, 0x20, 0x49 },

new byte[] { 0x4D, 0x4D, 0x00, 0x2A },

new byte[] { 0x4D, 0x4D, 0x00, 0x2B } });

imageHeader.Add("TIFF", new byte[][] { new byte[] { 0x49, 0x49, 0x2A, 0x00 },

new byte[] { 0x49, 0x20, 0x49 },

new byte[] { 0x4D, 0x4D, 0x00, 0x2A },

new byte[] { 0x4D, 0x4D, 0x00, 0x2B } });

imageHeader.Add("GIF", new byte[][] { new byte[] { 0x47, 0x49, 0x46, 0x38, 0x37, 0x61 },

new byte[] { 0x47, 0x49, 0x46, 0x38, 0x39, 0x61 } });

imageHeader.Add("BMP", new byte[][] { new byte[] { 0x42, 0x4D } });

imageHeader.Add("ICO", new byte[][] { new byte[] { 0x00, 0x00, 0x01, 0x00 } });

bool isTrueImage = false;

if (fileUploadControl.HasFile)

{

// GET FILE EXTENSION

string fileExt = Path.GetExtension(this.fileUploadControl.PostedFile.FileName).TrimStart('.').ToUpper();

// CUSTOM VALIDATION GOES HERE BASED ON FILE EXTENSION IF ANY

byte[][] tmp = imageHeader[fileExt];

foreach (byte[] validHeader in tmp)

{

byte[] header = new byte[validHeader.Length];

// GET HEADER INFORMATION OF UPLOADED FILE

//fileUploadControl.FileContent.Seek(0, SeekOrigin.Begin);

fileUploadControl.FileContent.Read(header, 0, header.Length);

if (CompareArray(validHeader, header))

{

// VALID HEADER INFORMATION

isTrueImage = true;

break;

}

}

}

if (!isTrueImage)

{

lblStatus.ForeColor = System.Drawing.Color.Red;

lblStatus.Text += "<br />Invalid file header! ";

}

return isTrueImage;

}

private bool CompareArray(byte[] a1, byte[] a2)

{

if (a1.Length != a2.Length)

return false;

for (int i = 0; i < a1.Length; i++)

{

if (a1[i] != a2[i])

return false;

}

return true;

}

# Client-Side Validation

ASP.NET yazılımcılara daha dosyayı yükleme aşamasında kullanabilecekleri bir takım doğrulama yöntemleri sunmaktadır. Validation’lar… Validation’lar, yazılımcılara dosya yüklenip yüklenmediğini veya yüklenen dosyanın belirtilen regex’e (whitelist) uygun bir uzantıda olup olmadığını kontrol etmeye ve eğer istenmeyen bir durum varsa bu konuda uygun dönüşler yapması ve işlemi gerçekleştirmemesi konusunda yardımcı olur. ASP.NET’te validation’lar genel olarak sadece dosya yükleme için değil, diğer birçok doğrulama türü için kullanılabilmektedir.

Dosya yükleme esnasında client-side validation kullanımı daha çok .Net’te yaygındır ve sebebi de belirttiğimiz gibi .Net’in yazılımcıların işini kolaylaştıran bu imkanı basitçe sunmuş olmasından kaynaklanmaktadır.

ASP.NET’te örnek bir validation kullanımı:

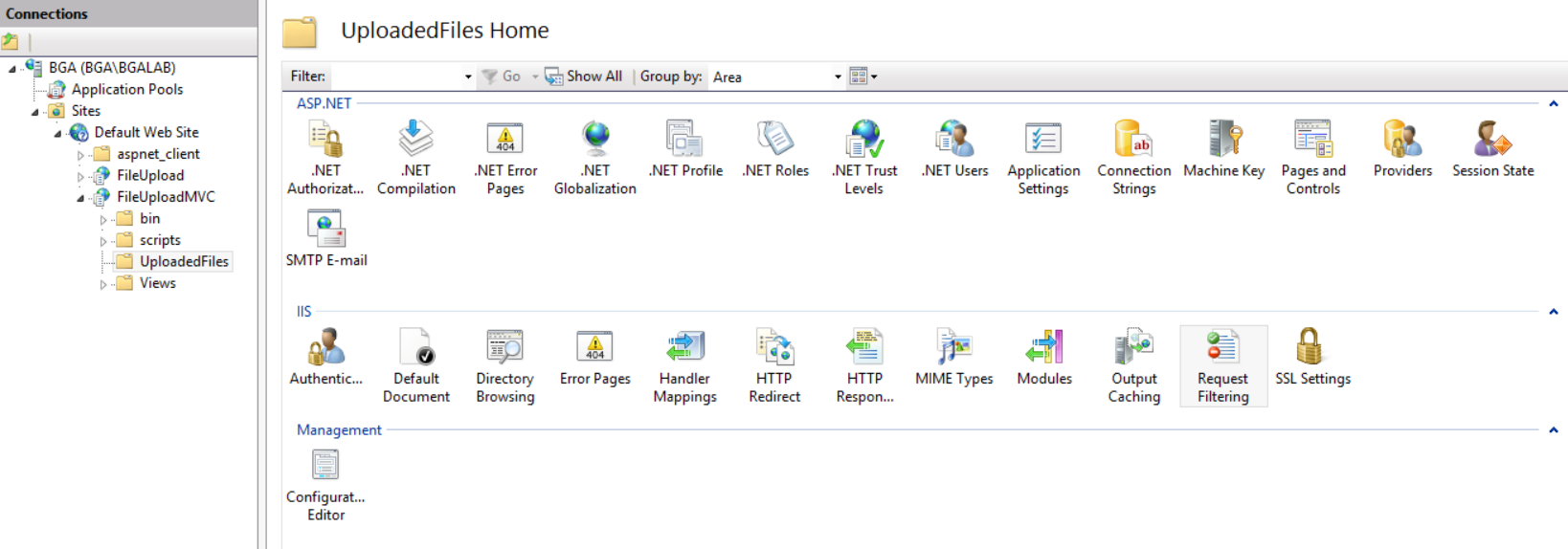
|  |
| --- |
| <asp:FileUpload ID="fileUploadControl" runat="server" />  <asp:RequiredFieldValidator ID="rfvFileUpload" runat="server"  ControlToValidate="fileUploadControl" ErrorMessage="File required." Display="Dynamic" ForeColor="Red">  </asp:RequiredFieldValidator>  <asp:RegularExpressionValidator ID="revFileUpload" ValidationExpression="([a-zA-Z0-9\s\_\\\-:])+(.jpg|.jpeg|.png|)$"  ControlToValidate="fileUploadControl" runat="server" ForeColor="Red" ErrorMessage="Please select a valid Word or PDF file."  Display="Dynamic" />  <asp:Button ID="btnUpload" runat="server" OnClick="btnUpload\_Click" Text="Save" /> |

ASP.NET, Jquery, HTML5, javascript veya diğer programlama dilleri ile yazılan client-side yöntemler saldırgan tarafından Burp veya Fiddler gibi araçlar ile kolayca bypass edilebileceği için yazılımcının kullanacağı bu yöntem güvensiz upload’lara karşı kesin alınmış bir önlem olarak kabul edilemeyecektir.

# IIS Request Filtering

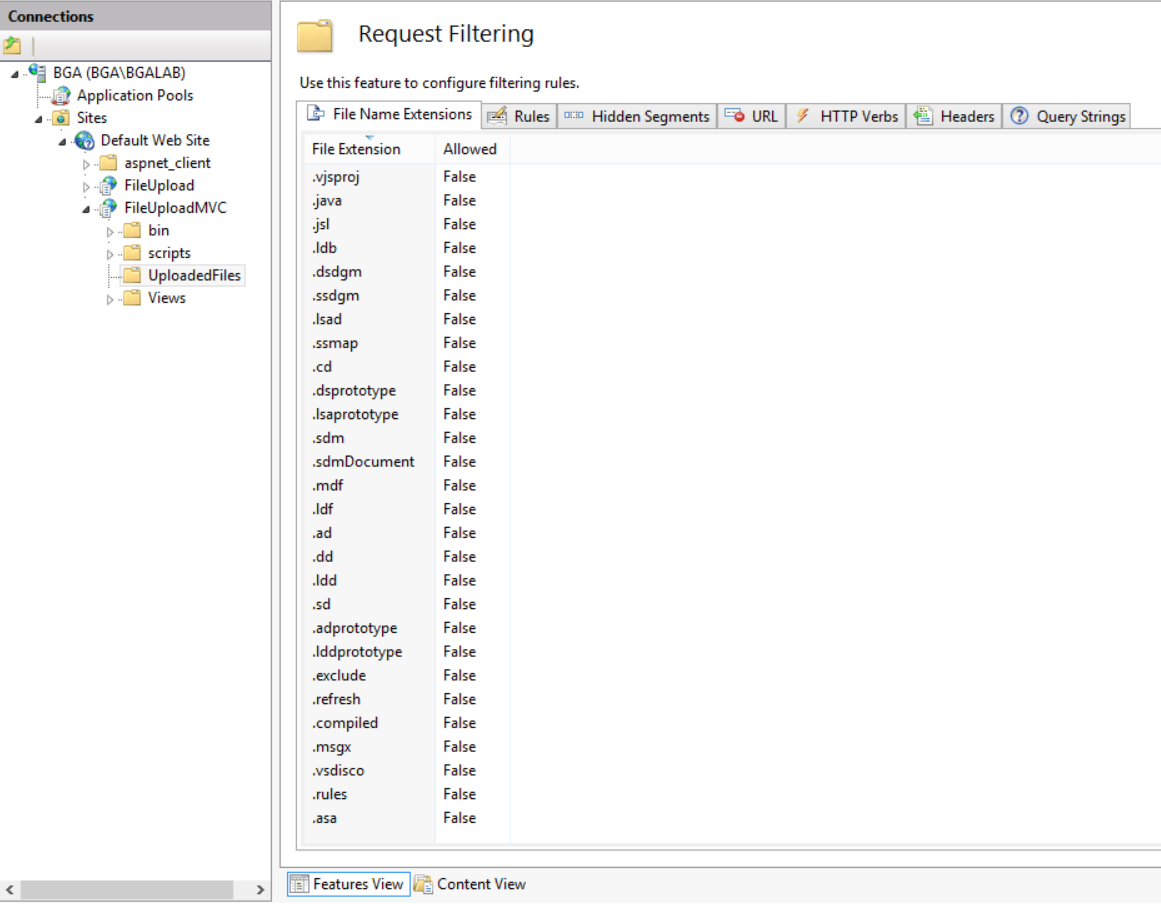
IIS (Internet Information Services) 7.0 ile gelen bir istek filtreleme mekanizması olan Request Filtering ile istenilen dizinde belirlenen türlerdeki dosyalar için gelen isteklere izin vermeyi ya da isteği reddetmeyi sağlamaktadır. Bu özelliği kullanmanın yolu ise şu şekilde:

IIS’te istek filtreleme yapılacak alana tıklanır:



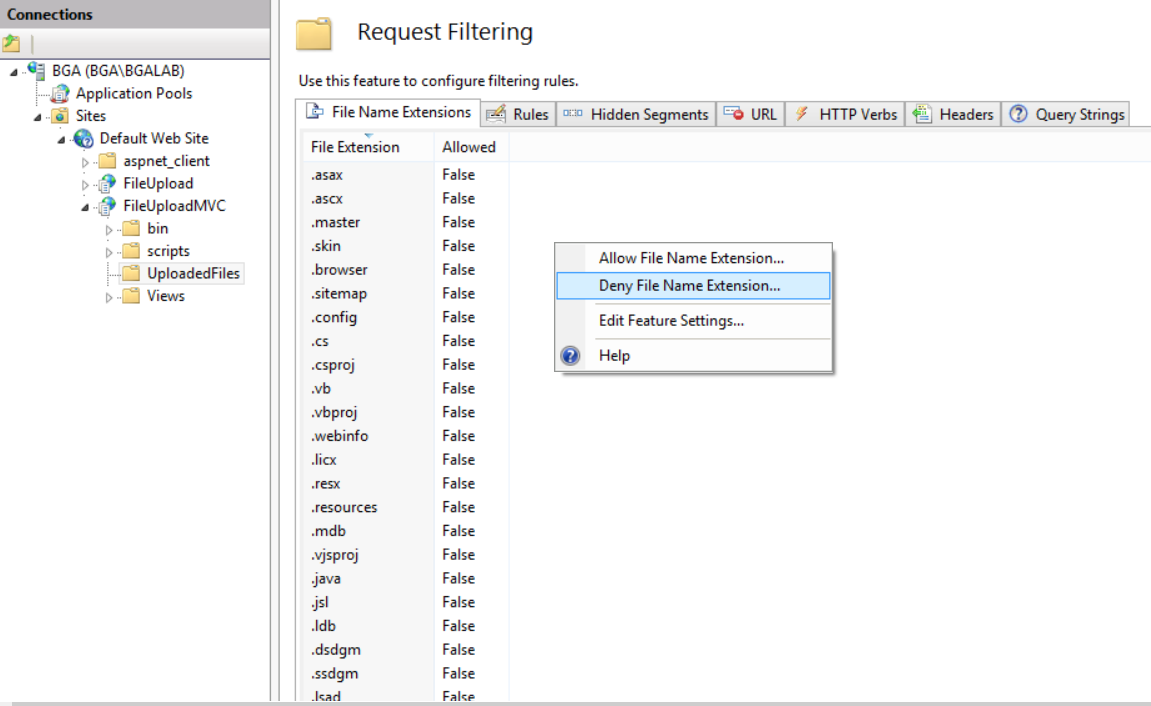
*Şekil 1 - IIS Görünüm*

Request Filtering yazan yere tıklandığında da aşağıdaki kısım açılacaktır:



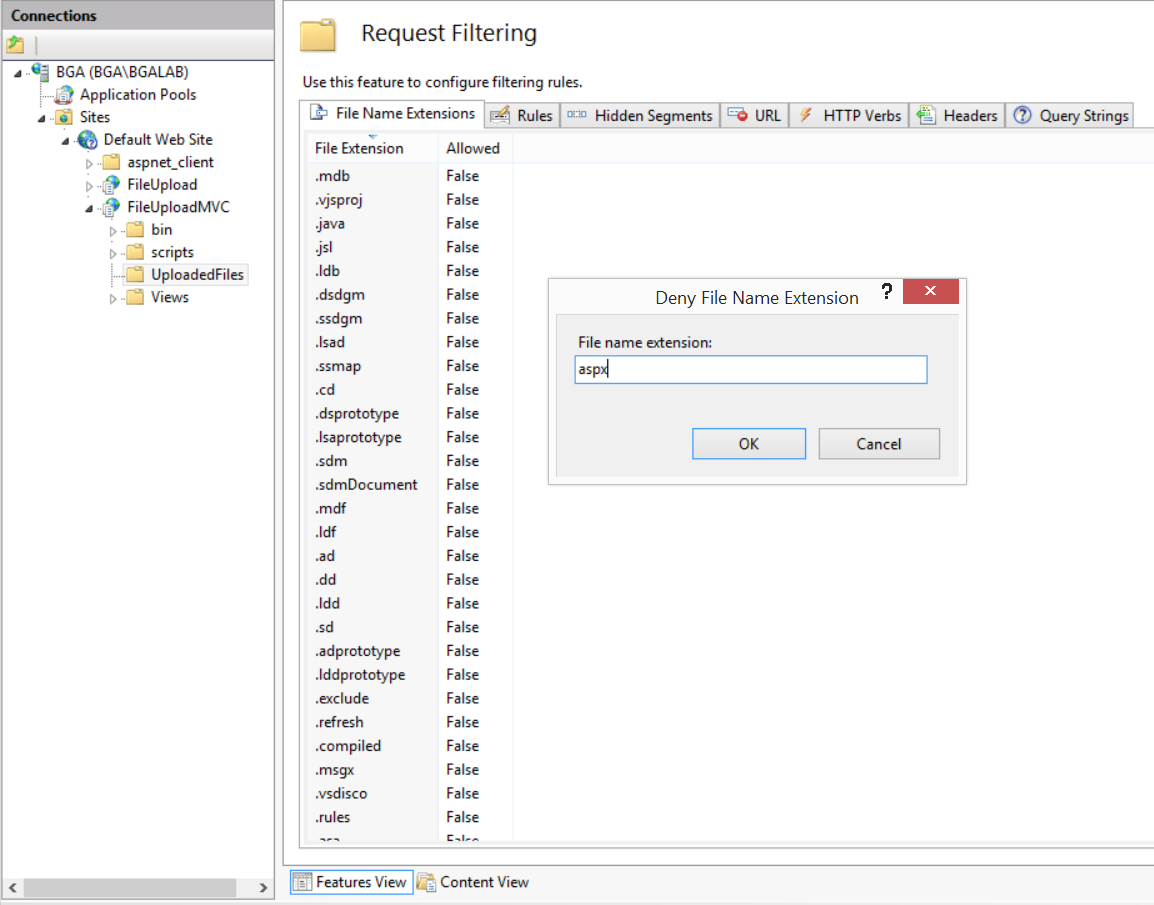
*Şekil 2 - File Name Extensions Alanından Çalıştırılmasının Engellendiği Dosya Uzantılarının Listesi*

Boş alana sağ tıklanınca açılacak küçük pencereden Deny File Name Extension seçeneği seçilir:



*Şekil 3 - File Name Extensions Alanından Çalıştırılması Engellenecek Dosya Uzantısının Eklenmesi*

Açılacak küçük pencereye istenilen uzantı yazılıp OK’a basılınca listeye eklendiği görülecektir:



*Şekil 4 - File Name Extensions Alanından Çalıştırılması Engellenmek İstenen Dosya Uzantılarının Belirlenmesi*

Bu yöntemi kullanarak da projedeki belli klasörlerde veya tamamında belirlenen türdeki dosyalar için gelen istekler kabul edilmeyebilir ve atılması olası herhangi bir shell ya da zararlı içerikli bir dosyanın çağrılıp çalıştırılması engellenecektir.

Buraya kadar anlatılanlardan da görüldüğü gibi bir saldırgan dosya yükleme alanlarında birçok bypass yöntemi kullanabilmektedir. Yazılımcıdan beklenen bu anlatılan hususlara dikkat ederek önlemler almak ve bu önlemlerin çalışır olduğundan emin olmaktır.

Bu yazı .Net file upload control’ü üzerinden yazılmıştır. Ancak alınacak önlemler diğer diller ile benzer başlıkları taşımaktadır. Faydalı olması dileğiyle.

**BGA Bilgi Güvenliği A.Ş. Hakkında**

BGA Bilgi Güvenliği A.Ş. 2008 yılından bu yana siber güvenlik alanında faaliyet göstermektedir. Ülkemizdeki bilgi güvenliği sektörüne profesyonel anlamda destek olmak amacı ile kurulan BGA Bilgi Güvenliği, stratejik siber güvenlik danışmanlığı ve güvenlik eğitimleri konularında kurumlara hizmet vermektedir.

Uluslararası geçerliliğe sahip sertifikalı 50 kişilik teknik ekibi ile, faaliyetlerini Ankara ve İstanbul ve USA’da sürdüren BGA Bilgi Güvenliği’nin ilgi alanlarını “*Sızma Testleri,* *Güvenlik Denetimi, SOME, SOC Danışmanlığı, Açık Kaynak Siber Güvenlik Çözümleri, Büyük Veri Güvenlik Analizi ve Yeni Nesil Güvenlik  Çözümleri*” oluşturmaktadır.

Gerçekleştirdiği başarılı danışmanlık projeleri ve eğitimlerle sektörde saygın bir yer edinen BGA Bilgi Güvenliği, kurulduğu günden bugüne alanında lider finans, enerji, telekom ve kamu kuruluşlarına 1.000'den fazla eğitim ve danışmanlık projeleri gerçekleştirmiştir.

BGA Bilgi Güvenliği, kurulduğu 2008 yılından beri ülkemizde bilgi güvenliği konusundaki bilgi ve paylaşımların artması amacı ile güvenlik e-posta listeleri oluşturulması, seminerler, güvenlik etkinlikleri düzenlenmesi, üniversite öğrencilerine kariyer ve bilgi sağlamak için siber güvenlik kampları düzenlenmesi ve sosyal sorumluluk projeleri gibi birçok konuda gönüllü faaliyetlerde bulunmuştur.

**BGA Bilgi Güvenliği AKADEMİSİ Hakkında**

BGA Bilgi Güvenliği A.Ş.’nin eğitim ve sosyal sorumluluk markası olarak çalışan Bilgi Güvenliği AKADEMİSİ, siber güvenlik konusunda ticari, gönüllü eğitimlerin düzenlenmesi ve  siber güvenlik farkındalığını arttırıcı gönüllü faaliyetleri yürütülmesinden sorumludur. Bilgi Güvenliği AKADEMİSİ markasıyla bugüne kadar “Siber Güvenlik  Kampları”, “Siber Güvenlik Staj Okulu”, “Siber Güvenlik Ar-Ge Destek Bursu” , ”Ethical Hacking yarışmaları” ve “Siber Güvenlik Kütüphanesi”  gibi birçok  gönüllü faaliyetin destekleyici olmuştur.