HTTP

Hyper Text Transfer Protocol olarak bilinen ve WWW üzerinde veri iletimi amacı ile 1989 dan bu yana kullanılan bir protokoldür.

HTTP KOMUTLAR

GET

Bir kaynağın talebi için kullanılır. Veri talebinden ve transferinden başka bir işlevi bulunmaz. Web tarayıcılarımızın adres çubuğunda gördüğümüz modeldir. Örneğin;

http://ogrenciportal.kku.edu.tr/test.php?index=5

Web tarayıcılarımız bu tür bir isteği karşı sunucuya iletebilmek için HTT Protokolüne uygun olarak iletmeleri gerekir; Şimdi yukarıdaki gibi bir isteğin yapılabilmesi için nasıl bir başlık oluşturmamız gerektiğine bakalım.

```
GET /test.php HTTP/1.1
Host: ogrenciportal.kku.edu.tr
Accept: image/gif, image/jpeg, */*
Accept-Language: tr-TR, en-us
Accept-Encoding: gzip,deflate
User-Agent: Opera 53 (compatible; MSIE 6.0; Windows NT 5.1)
Content-Length: 16
(boş satır)
test.php?index=5
```

Bu isteğimize binayen sunucu, aşağıdaki gibi bir cevap döner.

```
HTTP/1.1 200 OK
Date: Sun, 18 Oct 2009 08:56:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Sat, 20 Nov 2004 07:16:26 GMT
ETag: "10000000565a5-2c-3e94b66c2e680"
Accept-Ranges: bytes
Content-Length: 39
Connection: close
Content-Type: text/html
(boş satır)
<html><body><h1>cevap</h1></body></html>
```

POST

Karşıtaraftaki sunucya bir veri iletimi göndemek için kullanılan komuttur. Web tarayıcılarımızda form doldurup gönderme girişiminde bulunduğumuzda kullanılır. application/x-www-form-urlencoded ve multipart/form-data olamak üzere genel kullanılan iki biçimi vardır.

application/x-www-form-urlencoded: HTTP GET methodundan teknik olarak farkı bulunmaz. Gönderilmek istenen verilerin tamamı yine bir Query String formunda ve başlık bilgisinin takip eden 2 satır başı karakterinden sonra yazılır.

POST /login.aspx HTTP/1.1
Host: ogrenciportal.kku.edu.tr
Content-Type: application/x-www-form-urlencoded
Content-Length: 33
(boş satır)
username=140205054&passwrod=12345

Bu biçimdeki girdilerin parçalanabilmesi için hazırladığımız işlev aşağıdaki gibidir.

```
bool x_www_form_urlencoded(IHttpContext* context)
{
	if (context->GetRequest()->GetRemainingEntityBytes() < 1)
		return true;

	stringstream eb;

	while (context->GetRequest()->GetRemainingEntityBytes() != 0)
	{
		char* pchr = static_cast<char*>(context->AllocateRequestMemory(BS));
		memset(pchr, 0, BS);

		if (pchr == 0)
			return false;

		DWORD cbrec = 0;
		context->GetRequest()->ReadEntityBody(static_cast<void*>(pchr), BS - 2,
		false, &cbrec);

		eb << pchr;
	}

	if (eb.rdbuf()->in_avail() == 0)
		return true;

	string req(eb.str());

	std::vector<std::string> pairs = parse(req.c_str(), '&');
	std::map<std::string, std::string> Form;
```

```
		for (int i = 0; i < pairs.size(); i++)
		{
			size_t pos = pairs[i].find_first_of("=");

			if (pos == string::npos)
				continue;

			Form[pairs[i].substr(0, pos)] = pairs[i].substr(pos + 1);
		}

		for (std::map<std::string, std::string>::iterator it = Form.begin();
			it != Form.end(); it++)
		{
			if (it->first == "__VIEWSTATE" || it->first == "__EVENTVALIDATION")
				continue;


			match_results<std::string::const_iterator> mr;
			string in_text = decode(it->second);
			if (regex_search(in_text, mr, inj))
			{
				for (int x = 0; x < mr.size(); ++x)
				{
					if (mr[x].matched == true)
					{

						log(in_text, mr[x].str(),
inet_ntoa(((sockaddr_in*)context->GetRequest()->GetRemoteAddress())->sin_addr), context);

						return false;
					}
				}
				return false;
			}
		}

		return true;
	}
```

multipart/form-data: Bu yöntem sadece gönderilmek istenen veri bloğu binary bilgi yada büyük boyutta dosya içerdiğinde kullanılır. Burada dikkat edilmesi gereken; Her multipart/form-data post biçimi verilerde geçmeyen ve ayraç amacıyla kullanılan bir boundary belirlemek durumundadır.

```
POST /bin/upload.aspx HTTP/1.1
Host: ogrenciportal.kku.edu.tr
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Content-Length: 342
Connection: Keep-Alive
Cache-Control: no-cache
Content-Type: multipart/form-data; boundary=AaB03x


--AaB03x
Content-Disposition: form-data; name="isim"

Ahmet
--AaB03x
Content-Disposition: form-data; name="dosyalar"
Content-Type: multipart/mixed; boundary=BbC04y

--BbC04y
Content-Disposition: file; filename="file1.txt"
Content-Type: text/plain

dosya 1 içeriği
--BbC04y
Content-Disposition: file; filename="file2.gif"
Content-Type: image/gif
Content-Transfer-Encoding: binary

dosya 2 içeriği
--BbC04y--
--AaB03x--
```

Projemizde yine multipart/form-data biçimindeki girdikleri parçalayabilmek için kullandığımız işlev aşağıdaki gibidir.

```
bool multipart_formdata(IHttpContext* context)
	{
		if (context->GetRequest()->GetRemainingEntityBytes() < 1)
			return true;

		stringstream eb;

		while (context->GetRequest()->GetRemainingEntityBytes() != 0)
		{
			char* pchr = static_cast<char*>(context->AllocateRequestMemory(BS));
			memset(pchr, 0, BS);

			if (pchr == 0)
				return false;

			DWORD cbrec = 0;
			context->GetRequest()->ReadEntityBody(static_cast<void*>(pchr), BS, false,
&cbrec);


			eb << pchr;
		}

		if (eb.rdbuf()->in_avail() == 0)
			return true;

		string rt(eb.str());

		map<string, string> Form;
		string line;
		bool bpp = false;

		while (!eb.eof())
		{
			getline(eb, line);
			if (starts_with(line, "--"))
			{
				string subline;
				if (eb.eof())
					break;

				getline(eb, subline);
				if (!starts_with(subline, "Content-Disposition"))
				{
					eb.seekg(-subline.size(), ios::cur);
					continue;//fake start
				}

				vector<string> pairs = parse(subline.c_str(), ';');
				string form_elem_id;
				bool bfileupload = false;
				for (int i = 0; i < pairs.size(); i++)
				{
					size_t pos = pairs[i].find_first_of("=");
					if (pos == string::npos)
						continue;

					string p1 = pairs[i].substr(0, pos);
					trim(p1);

					if (p1 == "name")
						form_elem_id = pairs[i].substr(pos + 1);
```

```
                                else if (p1 == "filename")
                                {
                                        bfileupload = true;
                                        break;
                                }
                        }
                        if (bfileupload)
                                continue;//skip files
                        if (form_elem_id.size() < 1)
                                continue;//skip invalid ;
                        if (form_elem_id == "__VIEWSTATE" || form_elem_id ==
"__EVENTVALIDATION")
                                continue;//refuse asp.net driven values

                        Form[form_elem_id] = "";

                        while (!eb.eof())
                        {
                                if (eb.eof())
                                        break;

                                getline(eb, subline);

                                if (subline != "\r")
                                        continue;

                                //the following is the next fv
                                if (eb.eof())
                                        break;

                                getline(eb, subline);
                                Form[form_elem_id] = subline.substr(0, subline.size() - 1);
                                break;
                        }
                }
        }
```

SQL INJECTION

Web sistemleri üzerinde sıklıkla kullanılan bir programlama tekniğinden kaynaklanan bir güvenlik açığı türüdür. HTTP bilindiği üzere kaynakları birbirine kısayolar yoluyla bağlayan bir sistemdir. Bilhassa büyük sistemlerde kısayollar çok fazla oldukları için sistemler otomatize edilmiştir. Örneğin tipik bir haber sitesi için adres çubuğunda

http://www.habersitesi.com/haberdetay.aspx?haberid=754

şeklinde bir i fadeye rastlamak mümkündür. Hatta farklı haberler için sistematik olarak farklı ID ler verildiğide görülebilir.

Bu sistemler üzerinde tasarım genel manada şablonlar halinde olup, şablonların içine doldurulacak veriler veritabanlarında tutulmaktadır. Sayfaya yerleşecek olan kaydın ID nosu ise yukarıda görüldüğü gibi Query String içersinde gömülüdür ve genel izlenen yöntem bir sql sorgusuna bu ID yi gömerek sorguyu gerçekleştirmektir. Ör;

```
string sql = "SELECT * FOM haberdetay WHERE ID="+ Request.QueryString["haberid"];
```

Herhangi bir doğrulama (sanity check) yapmaksızın kullanıcıdan gelen veriyi doğrudan veritabanına çalıştırılabilir kod olarak göndermek naif bir yaklaşım biçimidir ve bahsettiğimiz Sql Injection güvenlik açığının temelini oluşturmaktadır.

Şimdi biz bu kulanıcının art niyetli olduğunu ve haberid kısmını değiştirdiğini düşünelim. Buradaki sayı yerine "; DROP TABLE haberdetay -- " yazdığını varsayalım bu halde iken bu naif kodumuzun veritabanına göndereceği komut tam olarak aşağıdaki biçimi alır.

```
SELECT * FROM haberdetay WHERE ID=';  DROP TABLE haberdetay --
```

Görülebileceği üzere burada iki komut bulunmakta birinci komut olan

```
SELECT * FROM haberdetay WHERE ID='
```

Hatalı kout olduğu için pas geçilir ikinci komut olan

```
DROP TABLE haberdetay
```

ise haberdetay tablosunu tamamen silecektir. İş bu halde iken bütün kullanıcı girişlerinin bir doğruluk kontrolünden geçmesi gerektiği ortadadır. Fakat, her bir giriş noktasına bunu yapmak pratik olmayabilir yada unutulabilir. Bütün bu problemleri kökten çzöümü için otomatize bir sistem gerektiği ortadadır.

FILE INCLUSION

Yine web siteleri üzerinde kullanılan farklı bir programlama alışkanlığının sonucu olarak ortaya çıkan bir güvenlik açığıdır ve şu an sadece PHP ile yapılan sistemleri etkilemektedir.

Bugün herhangi orta-büyük ölçekli web sitesini izlediğinizde sayfalar değişmesine rağmen başlık ve alt kısımların sabit orta kısmın ise içeriğe göre yeniden şekillendirildiği Master Page tarzı programlama tekniğini görebilirsiniz. Bu teknik PHP üzerinde oluşturulurken içerik teşkil eden sayafa include yoluyla taban sayfaya dahil edilir. Bugün tipik bir PHP siteminde adres çubuğunda

http://www.habersitem.com/index.aspx?sayfa=haberdetay&haberid=754

şeklinde bir ifadeye rastlamak mümkündür. İçeride ise kullanılan kod

<?php “include/”.include($_GET['haberdetay'].“.php”); ?>

Biçiminde naif bir yaklaşımdır. Çünkü Query String aşağıdaki biçimde değiştirildiğinde

http://www.habersitem.com/index.aspx?sayfa=http://www.benimsitem/c99&haberid=754

include edilecek sayfa http://www.benimsitem.com/ konumundaki c99 isimli dosyadır ki bu; PHP için hazırlanmış bir yönetim panelidir. Böylelikle saldırgan web sitesi üzerinde çalışan bir yönetim paneli ele geçirmiş ve istediği tüm kodları sunucu üzerinde çalıştırabilecek hale gelmiştir. Bu durumun engellenebilmesi için yine bütün girişlerde sanity check yapılmalıdır. Bu bazen pratik olmayabileceği gibi bazen de unutulabilir. Dolayısıyla yine bu durumda da insan hatalarından etkilenmeyen otomatize bir güvenlik sistemine ihtiyaç duyulmaktadır.

IIS ve IIS MODULLER

IIS bugün windows platformlarla birlikte gelen modüler, programlanabilir web sunucu yazılımıdır. IIS için yapılan ek modüllere ISAPI Extension denilmektedir. Gloabl ve Request Level olmak üzere iki biçimi vardır. Bizim projemiz Request Level ISAPI Extension sınıfıdır.

IIS 7.0 ile birlikte ortaya konulan standarta göre her ISAPI Extension belirli bir formatı takip etmelidir. Bunlar;

- RegisterModule işlevini EXPORT etmek
- CHttpModule sınıfından türemiş ve OnBeginRequest işlevini override etmiş bir sınıfın sağlamak
- IHttpModuleFactory sınıfından türemiş ve GetHttpModule işlevinin override etmiş bir sınıfı sağlamak

Dolayısıyla minimum yapı aşağıdaki şekilde olmalıdır.

```
class CHelloWorld : public CHttpModule
{
public:
    REQUEST_NOTIFICATION_STATUS OnBeginRequest(
        IN IHttpContext * pHttpContext, IN IHttpEventProvider * pProvider)
    {
      //...
    }
};


class CHelloWorldFactory : public IHttpModuleFactory
{
public:
    HRESULT GetHttpModule( OUT CHttpModule ** ppModule, IN IModuleAllocator * pAllocator)
    {
    }

    void Terminate()
    {
        delete this;
    }
};

HRESULT __stdcall RegisterModule( DWORD dwServerVersion,
IHttpModuleRegistrationInfo * pModuleInfo, IHttpServer * pGlobalInfo)
{
    return pModuleInfo->SetRequestNotifications(new CHelloWorldFactory,RQ_BEGIN_REQUEST,0);
}
```

REGULAR EXPRESSION

Regular expression sadece terminallerden oluşan belirli tip ifadelerin izahında kullanılan bir dildir. Takdir edileceği üzere sunucular üzerinde yapılan saldırıların tespitinde de kullanılabilir. İş bu gerekçe ile genel tip SQL Injection saldırılarının tespiti amacıyla bir regex ifadesini projemiz dahilinde hazırladık.

```
([\\'\\;]+[\\s\\(]*select\\s*([\\w\\(\\)\\,\\s]+|(\\*)+)\\s*from(\\s+|\\W+)\\w+)
|([\\'|\\;\\s]+or[\\s\\(]+[\\(\\s]*\\w+[\\)\\s]*=[\\(\\s]*\\w+[\\)\\s]*)
|(insert\W+(into[a-zA-Z0-9\\(\\)\\-\\_\\S])?[a-zA-Z0-9]+=)
|(delete\\W+(from)+\\W+\\w+)
|(drop\\W+(table)+\\w+)
|(update\\W+\\w+\\W+(set)+\\W+)
```

Yine benzer şekilde Query String üzerinde bulunan adresleri yakalamak içinde bir regex daha oluşturduk.

```
^([a-zA-Z]{3,9}:\\/\\/){0,1}([a-zA-Z0-9\\-]+\\.)*([a-zA-Z0-9\\-]{2,6}\\/){1}([a-zA-Z0-9\\-\\_]+\\/)*
([a-zA-Z0-9\\-\\_]+\\.[a-zA-Z0-9\\-\\_]+){1}
```