Security Best Practice: Host Naming & URL Conventions

Security considerations for web-based applications

Abstract

A consideration often neglected by many organisations when rolling out new servers or developing web-based applications that will be accessible by Internet clients and customers is that of host and URL naming conventions. There are a number of simple steps that can be taken to strengthen the security of an environment or application making it more resilient to several popular attack vectors. By understanding how an attacker can abuse poorly thought out naming conventions, and by instigating a few minor changes, it is possible to positively increase the defence-in-depth stature of an environment.

Author

Gunter Ollmann, Professional Services Director – email: gunter [at] ngssoftware.com

Section 1: **Background**

From an attacker's perspective, the method by which an organisation names their Internet visible hosts or references web-application URL's can often be abused to make for a more successful attack. Due to a lack of insight or understanding of current attack vectors, many organisations are failing to follow best security practices in their host naming and linking conventions – thereby unwittingly aiding their attackers.

In the last 5 years, organisations have seen a phenomenal year-on-year increase in the number and sophistication of the vectors used by malicious attackers to target their customers or clients. Ranging from social engineering through to URL obfuscation and domain hijacking, attackers are abusing poorly thought out and implemented host naming and URL referencing conventions. For example, attacks such as Phishing often make use of confusing host names to dupe customers by directing them to web applications designed to impersonate a legitimate site – once the customer hits the fake site their authentication credentials are recorded for later use in financial fraud or identity theft.

In addition, many organisations who have developed web-based applications to service the requirements of their customers have given negligible thought to the increasing sophistication and length of URL's being used, and how confusing these can be to their customers. Consequently, it has become easier for attackers to disguise their attack within a URL – making it increasingly difficult for customers or third-party tools to detect any embedded malicious payloads.

By following a few simple best practices, organisations can easily strengthen the security of their environments against many of these attacks and make it much more difficult for an attacker to confuse customers or clients.

Section 2: Understanding the Threat

Attackers have an ever increasing number of vectors in which they can manipulate poorly thought-out and implemented online services. The consequences of this ranges from the erosion of customer confidence in the online offering, through to the manipulation and eventual compromise of the hosting environment.

To understand the necessity of improving the processes in which an organisation selects host names for their Internet services or references URL's within a web-based application, a study of key threats and the attack vectors that abuse them is required. This section focuses upon the techniques currently used by attackers to construct their attack.

2.1. Which Threats?

Depending upon an attacker's motivation and the sophistication of the online service, there are a large number of threats which an organisation may be exposed to. However, by focusing upon the threats that can make use of poorly implemented host naming procedures or web-application URL referencing, the number becomes more manageable.

Threats that traditionally make use of poor host naming and URL referencing include:

- Phishing – use of an electronic message (e.g. email, web banner advertising, instant messaging) to socially engineer a customer into following a disguised or obfuscated URL. The URL leads to a host controlled by the attacker in which they seek to harvest customer authentication details. See "The Phishing Guide" by the author for a comprehensive analysis of this threat.
- Cross-site Scripting – manipulation of a web-application's URL designed to cause an attackers code (hosted at an alternative site) to be executed within the customers web-browser. The attacker may choose to inject malicious content with the purpose of discrediting an organisation, or seek to actually compromise the customer's host.
- Preset Session Hijacking – the hijacking of a customer's interactive session after they have authenticated themselves using a SessionID specified by an attacker within an insecure URL. The attacker subsequently gains interactive access to the logged in session and may carryout application functions as if they were the real customer.
- 'Bot-Net Building – similar to Phishing however, the attacker's purpose is to compromise the customers host and install a remotely controllable agent rather than merely harvest authentication details. Depending upon the nature of the 'bot installed, the attacker may also monitor all network traffic and subsequently capture customer authentication details used for multiple online services.
- Mistyped Names – many customers mistype host names and registered domains. An attacker may register permutations of an organisations domain to capture these mistypes and direct them to an application of their choice. This alternative application may be used to discredit the organisation or seek to impersonate it with the aim of capturing customer authentication details.
- SQL Injection – abuse of poor data handling processes that causes an attackers code submitted through a URL to be executed by the applications backend database server. Through this vector, an attacker may choose to steal or corrupt the data contained in the database, or seek to compromise the database host.

2.2. The Attackers Armoury

Most attackers, whether they are malicious users or professional criminals, have a bag of 'tricks' from which they construct their attack. Many common attack vectors initially depend upon the manipulation of the host name and/or application URL to deceive the customer in order to be successful.

To conduct an attack comprised of any of the threats previously discussed, the attacker has a finite pool of techniques and vectors that he can use. The most important and successful techniques are:

- Registration of similarly named domains
- Manipulation of complex URL's

2.2.1. Registration of Similarly Named Domains

It is a simple process for an attacker to register a domain name through any international domain registrar. Consequently there are many routes and opportunities for third parties to register domain names that may infringe upon an organisation's trademark or be used to trick customers into believing that they have reached a legitimate host.

To understand the many permutations to this attack vector, let us assume that our example organisation's name is "Global Widgets" and that their normal web-application is available through www.globalwidgets.com. The most common techniques available for this attack include the following.

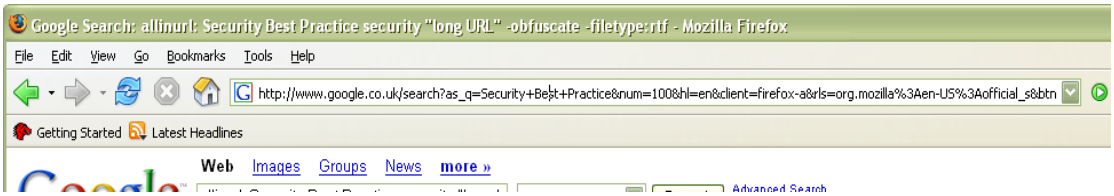
Attack Technique	Example
Hyphenated Names	http://www.global**-**widgets.com
Country Specific Registration	http://www.globalwidgets**.com.au**
Legitimate Possibilities	http://www.**secure-**globalwidgets.com
Mixed Wording	http://www.**widgetglobal**.com
Long Host Names	http://www.global.**widgets.com**
Mixed-case Ambiguities	http://www.g**i**oba**i**widgets.com
Common Misspellings	http://www.globalwidget.com (missing 's') http://www.globa**l**lwidgets.com (extra 'l')

This problem is compounded by the fact that many organisations also register a myriad of top level domains themselves, and then use them inconsistently when interacting with their customers. Due to this, it is almost impossible for a customer to differentiate between a 'new' legitimate domain (and the message using it) and one created by a malicious attacker.

2.2.2. Manipulation of Complex URLs

Due to poor application development processes and an over-reliance on third-party suite integration tools, organisations increasingly find themselves implementing long and complex URL's in order to provide access to specific components of their online service. Their customers are subsequently barraged with undecipherable URL's and links. Consequently, these customers are unlikely and often unable to identify URL's that may contain an attacker's malicious code.

Injecting malicious code (obfuscated or not) is a trivial process with many such applications and is unlikely to be detected at the time of clicking. As an example of how complex common application's URL can become, consider the following Google search request within a client browser.



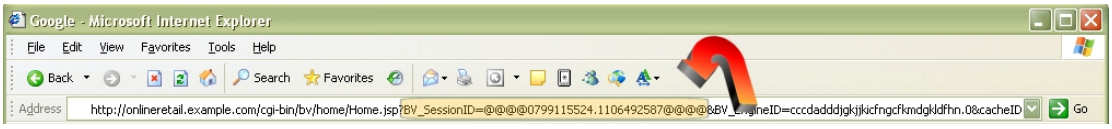
```
http://www.google.co.uk/search?as_q=Security+Best+Practice&num=100&hl=en&client=firefox-a&rls=org.mozilla%3Aen-US%3Aofficial_s&btnG=Google+Search&as_epq=long+URL&as_oq=security&as_eq=obfuscate&lr=lang_en&as_ft=e&as_filetype=rtf&as_qdr=y&as_occt=url&as_dt=i&as_sitesearch=&safe=active
```

Note how much of the URL is not actually visible within the browser window. Applications that make use of similarly long and complex URL's make for excellent attack delivery platforms since much of the payload can be hidden from view.

Session Information

The use of URL's that contain SessionID information can help an attacker carryout a number of sophisticated attacks – ranging from brute-forcing of access controls through to preset session hijacking. For many poorly constructed applications, if an attacker creates their own unique SessionID and passes the URL on to unsuspecting customers, they may be able to hijack a customer's account after they have successfully authenticated themselves using the attackers SessionID (this attack is commonly referred to as a 'Preset Session Attack').

Consider the following example containing SessionID information (by simply examining the URL below it is clear, to an experienced security expert, that the web application makes use of the BroadVision content delivery platform):




```
http://onlineretail.example.com/cgi-bin/bv/home/Home.jsp?BV_SessionID=@@@@0799115524.1106492587@@@@&BV_EngineID=cccdadddjgkjjkicfngcfkmdgkldfhn.0&cacheID=uknetscape&3283945388=3283945388&11064925878560.6464825117748412=11064925878560.44817548475695823&com.broadvision.session.new=Yes
```

Third-party Shortened URL's

Due to the length and complexity of many web-based application URLs – combined with the way URL's may be represented and displayed within various email systems (e.g. extra spaces and line feeds may be inadvertently inserted into the URL) – numerous third-party organisations have sprung up offering free services designed to provide shorter and more memorable URL's. The most popular web sites providing this functionality for free are http://smallurl.com and http://tinyurl.com.

An organisation that constantly uses very long or complex URL's will often find that customers expect not to understand a URL provided to them in any electronic communication. Subsequently,, the attacker can abuse this "trust" through a combination of social engineering and deliberately broken long or incorrect URL's, to cause the customer to follow a shortened URL. This short URL would be supplied by a third-party site and is used to obfuscate the true destination.

For example, a customer may be used to following complex links such as:

```
https://privatebanking.mybank.com/privatebanking/ebankver2/secure/customersupport.aspx?messageID=3324341&Sess=asp04&passwordvalidate=true&changepassword=true
```

The attacker could easily create a fake web site at a URL such as http://www.attackersite.com/fake/mybank/support and register it as http://tinyurl.com/4outd

Section 3: **Best Practices**

The secret to protecting against all of the threats and attack vectors explained in the previous section is by adopting a robust and comprehensive defence-in-depth posture. While there are no 'silver bullets' in information security, the inclusion of well thought out and implemented best practices can significantly contribute to an organisations ability to thwart many aspects of these attacks. In many cases, it is often the adoption of the simplest and most basic security best practices that have the greatest impact in helping to secure an organisation and the multiple Internet-based services it offers.

At a fundamental level, the process of keeping host names as simple and recognisable as possible – combined with the use of short URL's for referencing application components – can appreciably contribute to the overall security of an organisation's online service. Customers and clients must be able to tell at a glance exactly which service offering they are connecting to, and have confidence that they are not succumbing to a fraudulent link.

3.1. Domain Names and Host Services

All organisations with an Internet presence will have registered a domain name and are likely to own a number of very similar domain names. For the majority of web-based applications, the selected domain name forms an integral part of the online offering – providing information about the business unit or service being accessed by the customer, or the particular application sub-component being referenced.

Care should be taken when considering how domain names are to be used when delivering host services. Regardless of any particular attack vector, most customers are non-technical and are easily overwhelmed with the long and complex information presented in "follow this link" URLs.

Best practices in domain naming and host service referencing include:

- Use of the same top level domain
- Redirection of regional domains
- Representative service naming
- Use of the simplest and least confusing host name
- Avoiding host numbering

3.1.1. Use the Same Top-Level Domain

There has been a trend for organisations to register unique domains (or to direct multiple domain permutations on a theme to a single 'approved' new domain) for each new service the organisation creates. The result is an ever increasing menagerie of custom domains and loosely associated URL's which customers are required to trust in order to access new services. Consequently, an attacker who chooses to register and use a similarly sounding or affiliated domain name can easily deceive customers. To prevent this abuse, it is important that customers can always gain access to services (new or existing) through a well known and trusted domain.

Wherever possible, organisations should always use the same top-level domain for all Internet applications. The following table provides some best practice examples.

Use:	Instead of:
http://www.mybank.com**/ebank**	http://www.**mybank-ebank**.com
http://www.mybank.com**/UK**	http://www.mybank**uk**.com
https://**secure**.mybank.com	https://www.**secure-mybank**.com

3.1.2. Redirect Regional Domains

For organisations with an international presence, or those just wishing to safeguard certain country-specific domains, it is recommended that any registered regional domains are redirected to easily identifiable links from the root domain. In this way customers attempting to access any regional domain (e.g. www.mybank.co.za) would be automatically redirected to a single root domain (e.g. www.mybank.com) which would provide appropriate information or redirection to the country specific customer service. This process not only aids security, but also helps to promote a scalable environment capable of managing global load-balancing.

The following table provides some best practice examples.

Use:	Instead of:
http://www.mybank.com/**UK**	http://www.mybank**.co.uk**
https://**secure**.mybank.com/**AU**	https://www.**secure**.mybank.com.**au**
http://www.mybank.com/**DE/Investor**	http://www.**mybank-investor.de**

3.1.3. Representative Naming

It is recommended that host names be representative of the web-based application or service offered to the customer, and that it be clear or easily identifiable. This means that, instead of registering complex domains that include the service and organisations name (e.g. CustomerPlus-mybank.com), the root domain should be used and provide an appropriate URL instead (e.g. www.mybank.com/customerplus).

Additionally, when using secure services (such as SSL-based HTTPS), it is generally recommended that this added protection service be reflected in the host name. For instance, instead of referring to the standard host over HTTPS (e.g. https://www.mybank.com) the organisation should redirect customers to the dedicated secure host (e.g. https://secure.mybank.com). Similarly, double-barrelled domains that reflect additional security mechanisms (e.g. www.secure-mybank.com) should instead be replaced with an appropriate host name which is part of the same root domain (e.g. secure.mybank.com).

The following table provides some best practice examples.

Use:	Instead of:
https://secure.mybank.com	**https://www**.mybank.com
http://**invest**.mybank.com	http://www.**InvestorAtMyBank**.com
http://www.mybank.com/**invest**	http://**investment**.mybank.com

3.1.4. Simple URL Paths and Host Names

To prevent unnecessary confusion for customers or clients, the shortest and least complex combination of host name and URL should be used. Time and effort should be made when building an online web-application to ensure that each customer service can be referenced through a shortened URL. When selecting a host name, ensure that the name reflects the key aspects of the service.

The following table provides some best practice examples.

Use:	Instead of:
https://**secure**.mybank.com/**investor**	**https://www**.mybank.com/**secureinvestor**
http://**news**.mybank.com/**UK**	http://www.mybank.**co.uk/onlinebanking/cha**

	nges/news
https://**secure**.mybank.com/**ZA/personal**	**https://personal**.mybank**investor.co.za/securelogin**

3.1.5. Avoid Host Numbering

Organisations should use an appropriate mix of address translation and load-balancing technologies to prevent sequentially numbered hosts being visible on the Internet. Failure to do so will not only cause confusion to customers, but also aid potential attackers in their discovery of additional insecure hosts that could be individually targeted for later abuse.

The following table provides a best practice example.

Use:	Instead of:
http://**www**.mybank.com	http://**www3**.mybank.com

3.2. URL Referencing

Linked closely with domain names and host services, the URL's used within a web-based application must similarly be handled with care. To defend against many of the most malicious application-focused threats, organisations must review how URL's are used to navigate and access application functionality. Through a careful combination of application design and use of appropriate host naming conventions, URL complexity can be markedly reduced – thereby reducing the window of opportunity for an attacker.

Best security practices in URL referencing for web-based applications include:

- Keeping URLs as small as possible
- Never placing session information in a URL
- Removing application query variables from the URL

3.2.1. Small URL's are Best

Long URL's introduce unnecessary complexity to a customer's experience of the web-application as well as introducing numerous vectors for attack (of both the client and server). While traditional development techniques have made extensive use of the HTTP GET request, all this functionality can be easily migrated to HTTP POST requests instead. In addition to reducing the complexity of the URL's presented to the customer, this modification makes it more difficult for attackers to conduct certain attacks (such as Phishing, cross-site scripting and SQL injection).

Organisations should only use URL's to direct customers to key application components or services – ideally binding all environment details to their unique SessionID (e.g. customer identity, application preferences, etc.). All other details and data submissions can be handled through the use of HTTP POST requests through HTML forms. A combination of HTML forms and client-side scripting can enable any possible request that would traditionally be managed through HTTP GET requests alone.

With foresight, it is possible to develop applications that make use of short and very simple URL's. An advantage to this process is that the URL's become more memorable to the customer – consequently becoming easily transportable between systems, and easy for the customer to detect any obfuscated attacks. Application developers should strive to remove any reliance on individual page references, and instead use a well designed and implemented session management solution.

The following table provides some best practice examples.

Use:	Instead of:
http://www.mybank.com/**paybills**	http://www.mybank.com/**ebanking/customerportal/transactions/paybill/payment.aspx?country=UK&account=667996302pay=true**
https://secure.mybank.com/**transfers**	https://secure.mybank.com/**transfers.html**
http://www.mybank.com/**joining** …and track progress using HTTP POST variables which stage of the account creation process the customer is up to.	http://www.mybank.com/**ebanking/accountcreation/join.aspx?currentpage=3** followed by http://www.mybank.com/**ebanking/accountcreation/join.aspx?currentpage=4**

Developers should ensure that the application is configured in such a way that it is not possible for an attacker to successfully submit HTTP POST data through an alternative carefully designed URL.

However, it is important to note that the use of HTTP POST by itself does not provide a robust security mechanism for protection against all types of data manipulation – but it does provide some protection against the threats previously discussed. Through the use of personal proxies and other hacking tools, an attacker can easily manipulate data sent via the POST method.

3.2.2. Remove Session Information from URL's

Session information must never be stored or referenced directly through the applications URL. Instead, application developers must use session cookies (i.e. cookies which are temporarily recorded in the client browsers dynamic memory and are automatically purged when the browser is closed) to temporarily store SessionID information and use them instead for access control or application tracking.

If application SessionID's must be passed to the application (or associated/affiliated sites) it can be achieved through a combination of client-side scripting and the HTTP POST command.

For example, instead of using the following HTTP GET request:

```
http://www.mybank.com/ebanking/transfers/doit.aspx?funds=34000&agent=kelly02&sessionid
=898939289834
```

Application developers should use the more secure and robust HTTP POST method to make application requests. For example, the code behind the customer's page request may look like:

```
<FORM METHOD=POST ACTION="ebanking/transfers/doit.aspx">
<INPUT TYPE="hidden" NAME="funds" VALUE="34000">
<INPUT TYPE="hidden" NAME="agent" VALUE="kelly02">
<INPUT TYPE="submit" NAME="Transfer">
```

Which could result in the following HTTP POST from the client browser:

```
POST /ebanking/transfers/doit.aspx HTTP/1.1
Referer: http://www.mybank.com/ebanking/transfers/balance
Accept-Language: en-gb
Content-Type: application/x-www-form-urlencoded
Proxy-Connection: Keep-Alive
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Host: www.mybank.com
Content-Length: 46
Pragma: no-cache
Cookie: sessionid=898939289834

funds=34000&agent=kelly02&action=submit
```

Background on Cookies

Each time a client web browser accesses content from a particular domain or URL, if a cookie exists, the client browser is expected to submit any relevant cookie information as part of the HTTP request. Thus cookies can be used to preserve knowledge of the client browser across many pages and over periods of time. Cookies can be constructed to contain expiry information and may last beyond a single interactive session. Such cookies are referred to as "persistent cookies", and are stored on the client browsers hard-drive in a location defined by the particular browser or operating system. Omitting expiration information from a cookie, the client browser is expected to store the cookie in memory only. These "session cookies" should be erased when the browser is closed.

For example, within the plain text of the HTTPS server response:

```
Set-Cookie: sessionID="IE60012219"; path="/"; domain="www.example.com"; expires="1999-
06-01 00:00:00GMT"; version=0; secure
```

3.2.3. Remove Application Variables from URL's

As part of keeping application URL's as short and memorable as possible, organisations must ensure that application variables are not visible from within the URL. As a minimum, all application variables should be handled through hidden HTML form submissions. Ideally, all application variables should be stored at the server-side and associated with the unique SessionID allocated to the logged in customer.

3.3. Serial Host Naming

Many organisations adopt the use of serial naming procedures for individual host servers. In most cases, access to these servers is through a single well known host name which uses some kind of load balancing or round-robin allocation of traffic to direct customer requests to each individual host.

For example, responding to the well known name www.myretailer.com there could be 3 individual hosts called www1.myretailer.com, www3.myretailer.com and www5.myretailer.com.

Problems with the adoption of a serial host naming convention lie with the probability that attackers will cycle through individual host names in order to discover forgotten or insecure hosts. Frequently, while an organisation may have many load-balanced hosts typically available through a well known host name or URL, some of these hosts may not be configured as well as the others. An attacker may use the individual hosts name to connect directly to the server – and attempt to compromise its weaknesses.

If a serial host naming convention has been implemented by the organisation, it is a trivial task for the attacker to cycle through probable host names and potentially uncover hosts that may have been removed from service previously (i.e. available through the main URL). As such, organisations should not use a host naming policy that makes it easy for an attacker to discover non-public servers.

For example, an attacker notices that there are two public servers called Earth.myretailer.com and Saturn.myretailer.com. It does not take a genius to try other planet names and uncover additional (non-public) hosts. Similarly, the use of the seven-dwarfs, countries, cartoon characters, super heroes, car manufacturers, colours and diseases etc. should not be encouraged.

A three phased defensive practice is recommended:

1. Do not use sequential or closely related host names. Instead apply individual unrelated names to hosts.
2. Do not provide Internet accessible forward or reverse DNS entries for hosts that do not actually require named access over the Internet. Access to these hosts can be governed through appropriate load balancing technologies and address translation.

3. Manage the authoritative DNS servers correctly to ensure that only authorised hosts appear within the public DNS entries and that the DNS server itself is correctly configured to disallow zone transfers.

3.4. Domain Registration Monitoring

There are a number of commercial services available that help organisations monitor top level domains and alert when potentially threatening new domains are registered. Similarly, alerting services exist that will observe popular hacking chat rooms and posting forums for discussions on phishing and other spoofing scams.

It is strongly recommended that organisations either subscribe to third-party agencies that provide these services or develop an internal process that reviews newly registered domain names on a daily basis.

Section 4: Conclusions

The implementation of a well thought-out host naming and URL referencing convention can provide a sizable contribution to an organisations defence-in-depth posture. With so many customer-focused threats out there, the adoption of these simple best security practices will considerably aid the ability of an organisation to combat future targeted attacks and significantly increase their customer's confidence in their online offerings.

By applying common sense strategies to host naming conventions and the simplification of customer-visible URL's, organisations can expect to reduce the success of attacks as attackers struggle to 'socially engineer' their customers or transfer their attacks to another organisation which is a softer target.

4.1. Additional Resources

"The Phishing Guide", *Gunter Ollmann, 2004*

"URL Encoded Attacks", *Gunter Ollmann, 2002*

"HTML Code Injection and Cross-site scripting", G*unter Ollmann, 2001*

About Next Generation Security Software (NGS)

NGS is the trusted supplier of specialist security software and hi-tech consulting services to large enterprise environments and governments throughout the world. Voted "best in the world" for vulnerability research and discovery in 2003, the company focuses its energies on advanced security solutions to combat today's threats. In this capacity NGS act as adviser on vulnerability issues to the Communications-Electronics Security Group (CESG) the government department responsible for computer security in the UK and the National Infrastructure Security Co-ordination Centre (NISCC). NGS maintains the largest penetration testing and security cleared CHECK team in EMEA. Founded in 2001, NGS is headquartered in Sutton, Surrey, with research offices in Scotland, and works with clients on a truly international level.

About NGS Insight Security Research (NISR)

The NGS Insight Security Research team are actively researching and helping to fix security flaws in popular off-the-shelf products. As the world leaders in vulnerability discovery, NISR release more security advisories than any other commercial security research group in the world.