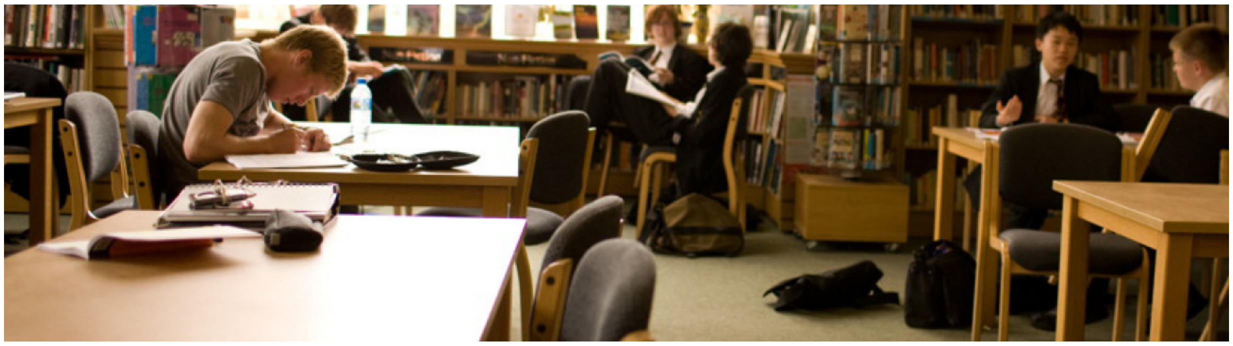
White Paper

Examining the Bitsquatting Attack Surface



Bit errors in memory, when they occur in a stored domain name, can direct Internet traffic to the wrong domain potentially compromising security. When a domain name one bit different from a target domain is registered, this is called “bitsquatting”. This paper describes several previously unknown forms of bitsquatting, and also proposes potential mitigations which do not involve the mass registration of additional bitsquat domains. The conclusion is that the possibility of bitsquat attacks is more widespread than originally thought, but several techniques exist for mitigating the effects of these new attacks.

Introduction

In the early 1980s, the 7-bit ASCII table became the de facto means of representing text inside computers. Several of the specific bitsquats that are possible today owe their very existence, or their non-existence, to the layout of the ASCII table. The 7-bit ASCII code is actually not a product of modern computers, but is descended from the early 5-bit “Baudot” codes used in the late nineteenth century and early twentieth centuries by printing telegraph machines. When computers became much more prevalent during the 1950s, it became necessary to standardize the representation of characters between different devices so they could better communicate. By the 1960s, the 5-bit codes used by the telegraph companies had given way to multiple 6-bit codes. Finally in 1963, a seven bit ASCII code was born which was essentially an amalgamation of the FIELDATA military specification, plus the existing ITA-2 telegraph alphabet [1][2].

If you analyze the layout of the ASCII table, some remnants of the old teletypes can be found. For example occupying the very last slot in the 7-bit ASCII table is the “DEL” or Delete character. In the olden days of punched tape and printing telegraphs, errors could be corrected by punching **all** the possible holes in a particular row of the tape. So, to this day the “DEL” character occupies the very last character in the 7-bit ASCII code, as it is represented by a string of all ones. It is in the context of the ASCII binary encoding of characters that we find our potential bitsquats – domains that are one binary digit different than another domain.

A memory error is a condition that occurs any time one or more bits being read from memory have changed state from what was previously written. Memory errors can be caused by a variety of conditions including cosmic radiation, operating devices outside their recommended environmental specifications, defects in manufacturing, and even nuclear explosions. While any bit in memory may be subject to errors, it is when bit errors occur inside of a stored domain name that subsequent Internet traffic may be misdirected. For example, by changing only one bit in the underlying ASCII representation, a popular target domain such as “twitter.com” can become the bitsquat domain “twitte2.com”. An attacker can take advantage of these bit errors by registering the bitsquat domain, and then intercepting data destined for the target domain, returning malicious data to the client, or performing other similar malicious activity.

In the original published research on bitsquatting, Dinaburg noted that the majority of the estimated 600,000 memory errors per day across the Internet are useless to a remote attacker [3]. Dinaburg therefore concluded that bitsquatting is most effective against the most frequently resolved domain names, since those domains are the most likely to appear in memory when bit errors occur. Our research supports this claim. However Dinaburg’s estimate of bit error rates was extremely conservative [4] and since that time most consumer grade computing devices being manufactured continue to lack error correcting memory. Further, the amount of memory per device and number of devices connected to the Internet are both increasing. Cisco estimates that there will be 37 billion “intelligent things” connected to the Internet by 2020 [5]. This is all good news for bitsquatters, as it means that domains that were previously not considered “popular” enough to attack will actually produce a useful amount of bitsquat traffic.

Additionally, it is not just the domain names themselves which are susceptible to bit errors in memory. Bit errors can and do occur anywhere. Sometimes bit errors occur simultaneously in multiple different locations. In fact, Dinaburg’s collected DNS data showed bit errors occurring in requested DNS record type values (ex. A, MX, NS, etc.) [6]. It is a certainty that the effects of bit errors are not confined to domain names themselves. Therefore bit errors must also affect commonly used Internet application layer protocols which rely on domain names, such as SMTP, SIP, or HTTP for example.

This all adds up to a landscape where bitsquatting attacks are more practical than ever before. In Section I, this paper demonstrates some previously unknown bitsquatting techniques using examples from real bitsquat domains that have been registered. Section II, suggests potential bitsquatting mitigations that can be used to help minimize, or even eliminate the potential for bitsquatting attacks altogether.

Section I – New Bitsquatting Attack Vectors

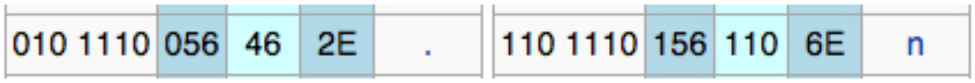
Subdomain Delimiter Bitsquatting

RFC1035 declared the valid syntax for domain name labels, which was later refined under RFC1123. The following BNF notation describes valid domain name label syntax. Essentially, the only allowed characters are A-Z, a-z, 0-9, and the hyphen.

```
<domain> ::= <subdomain> | " "
<subdomain> ::= <label> | <subdomain> "." <label>
<label> ::= <let-dig> [ [ <ldh-str> ] <let-dig> ]
<ldh-str> ::= <let-dig-hyp> | <let-dig-hyp> <ldh-str>
<let-dig-hyp> ::= <let-dig> | "-"
<let-dig> ::= <letter> | <digit>
<letter> ::= any one of the 52 alphabetic characters A through Z in upper case
and a through z in lower case
<digit> ::= any one of the ten digits 0 through 9
```

However when checking for bitsquat domains, limiting the search to characters in <let-dig-hyphen> neglects an important character that is also valid inside domain names: the dot character. This first new bitsquatting technique relies on bit errors which result in a letter “n” (binary 0**1**101110) becoming a dot “.” (binary 0**0**101110) and vice-versa. The technique functions because dots are used to delimit subdomains.

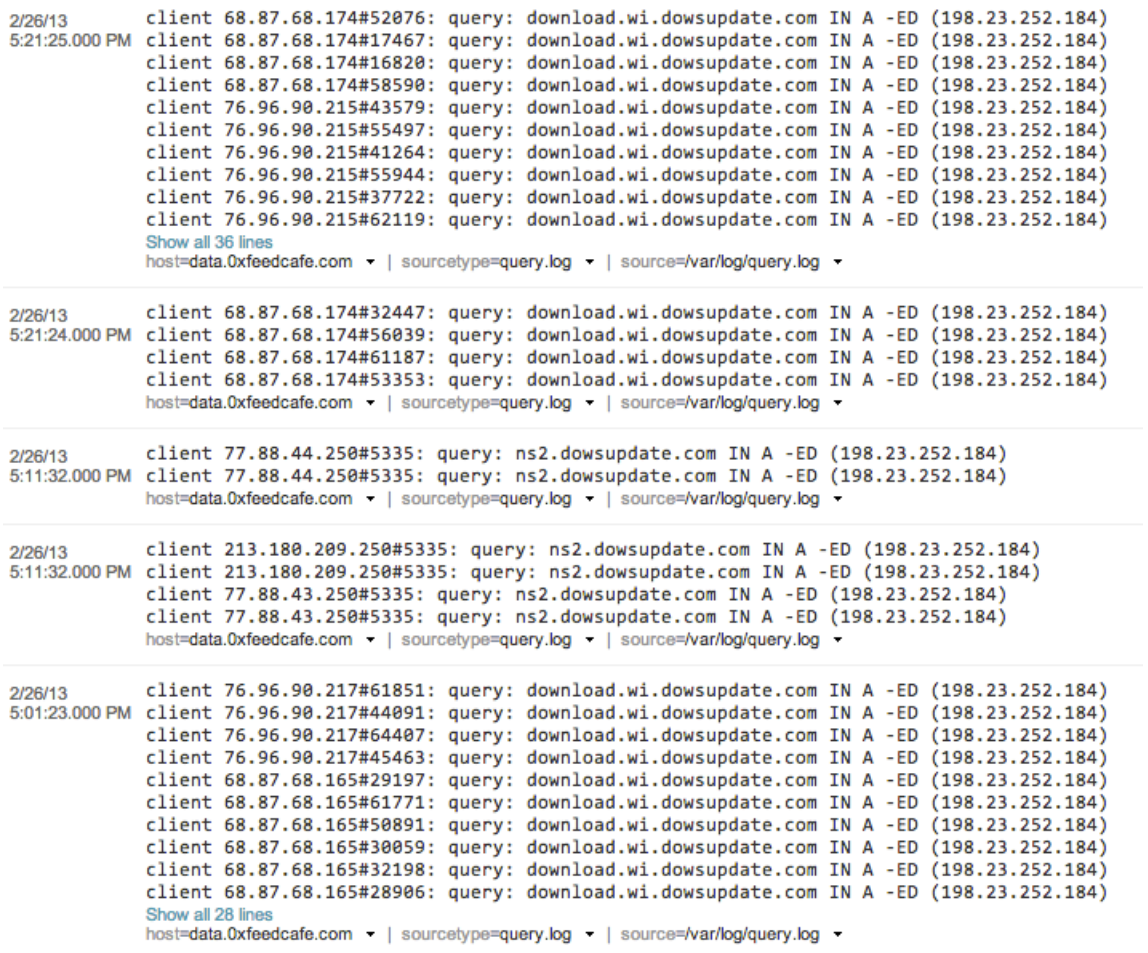
Figure 1.   A comparison of the ASCII representation of the dot '.' versus the letter 'n'



010 1110	056	46	2E	.	110 1110	156	110	6E	n

There are actually two distinct varieties of subdomain delimiter bitsquats. The first type occurs when there is a letter “n” present in the second level domain name. Domain names that contain a letter “n” character with 2 or more characters after the letter “n” are potential targets. The resulting bitsquat domain is shorter than the target domain. An example is the target domain “windowsupdate.com”. When the letter ‘n’ in this domain changes to a dot, the traffic is directed at the bitsquat domain "dowsupdate.com" instead as demonstrated in Figure 2.

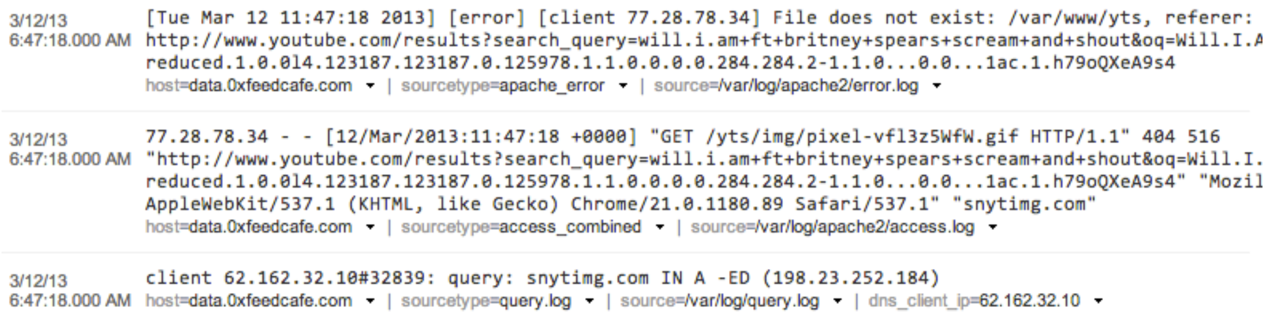
Figure 2. An example from the bitsquat domain "dowsupdate.com"



```
2/26/13          client 68.87.68.174#52076: query: download.wi.dowsupdate.com IN A -ED (198.23.252.184)
5:21:25.000 PM   client 68.87.68.174#17467: query: download.wi.dowsupdate.com IN A -ED (198.23.252.184)
                 client 68.87.68.174#16820: query: download.wi.dowsupdate.com IN A -ED (198.23.252.184)
                 client 68.87.68.174#58590: query: download.wi.dowsupdate.com IN A -ED (198.23.252.184)
                 client 76.96.90.215#43579: query: download.wi.dowsupdate.com IN A -ED (198.23.252.184)
                 client 76.96.90.215#55497: query: download.wi.dowsupdate.com IN A -ED (198.23.252.184)
                 client 76.96.90.215#41264: query: download.wi.dowsupdate.com IN A -ED (198.23.252.184)
                 client 76.96.90.215#55944: query: download.wi.dowsupdate.com IN A -ED (198.23.252.184)
                 client 76.96.90.215#37722: query: download.wi.dowsupdate.com IN A -ED (198.23.252.184)
                 client 76.96.90.215#62119: query: download.wi.dowsupdate.com IN A -ED (198.23.252.184)
                 Show all 36 lines
                 host=data.0xfeedcafe.com ▾ | sourcetype=query.log ▾ | source=/var/log/query.log ▾

2/26/13          client 68.87.68.174#32447: query: download.wi.dowsupdate.com IN A -ED (198.23.252.184)
5:21:24.000 PM   client 68.87.68.174#56039: query: download.wi.dowsupdate.com IN A -ED (198.23.252.184)
                 client 68.87.68.174#61187: query: download.wi.dowsupdate.com IN A -ED (198.23.252.184)
                 client 68.87.68.174#53353: query: download.wi.dowsupdate.com IN A -ED (198.23.252.184)
                 host=data.0xfeedcafe.com ▾ | sourcetype=query.log ▾ | source=/var/log/query.log ▾

2/26/13          client 77.88.44.250#5335: query: ns2.dowsupdate.com IN A -ED (198.23.252.184)
5:11:32.000 PM   client 77.88.44.250#5335: query: ns2.dowsupdate.com IN A -ED (198.23.252.184)
                 host=data.0xfeedcafe.com ▾ | sourcetype=query.log ▾ | source=/var/log/query.log ▾

2/26/13          client 213.180.209.250#5335: query: ns2.dowsupdate.com IN A -ED (198.23.252.184)
5:11:32.000 PM   client 213.180.209.250#5335: query: ns2.dowsupdate.com IN A -ED (198.23.252.184)
                 client 77.88.43.250#5335: query: ns2.dowsupdate.com IN A -ED (198.23.252.184)
                 client 77.88.43.250#5335: query: ns2.dowsupdate.com IN A -ED (198.23.252.184)
                 host=data.0xfeedcafe.com ▾ | sourcetype=query.log ▾ | source=/var/log/query.log ▾

2/26/13          client 76.96.90.217#61851: query: download.wi.dowsupdate.com IN A -ED (198.23.252.184)
5:01:23.000 PM   client 76.96.90.217#44091: query: download.wi.dowsupdate.com IN A -ED (198.23.252.184)
                 client 76.96.90.217#64407: query: download.wi.dowsupdate.com IN A -ED (198.23.252.184)
                 client 76.96.90.217#45463: query: download.wi.dowsupdate.com IN A -ED (198.23.252.184)
                 client 68.87.68.165#29197: query: download.wi.dowsupdate.com IN A -ED (198.23.252.184)
                 client 68.87.68.165#61771: query: download.wi.dowsupdate.com IN A -ED (198.23.252.184)
                 client 68.87.68.165#50891: query: download.wi.dowsupdate.com IN A -ED (198.23.252.184)
                 client 68.87.68.165#30059: query: download.wi.dowsupdate.com IN A -ED (198.23.252.184)
                 client 68.87.68.165#32198: query: download.wi.dowsupdate.com IN A -ED (198.23.252.184)
                 client 68.87.68.165#28906: query: download.wi.dowsupdate.com IN A -ED (198.23.252.184)
                 Show all 28 lines
                 host=data.0xfeedcafe.com ▾ | sourcetype=query.log ▾ | source=/var/log/query.log ▾
```

The second variety of subdomain delimiter bitsquat lengthens the 2nd level domain name and relies on the presence of 3rd level subdomains. An attacker can convert the dot separating the 3rd and 2nd level domain names into a “n” character, and register the resulting 2nd level domain. For an example, consider the hostname “s.ytimg.com” which is a host at the content delivery network used by YouTube. The resulting bitsquat domain is “snytimg.com”. Indeed, bitsquat traffic is going to this domain, and the HTTP requests for images have a Referrer HTTP header set to YouTube as shown in Figure 3.

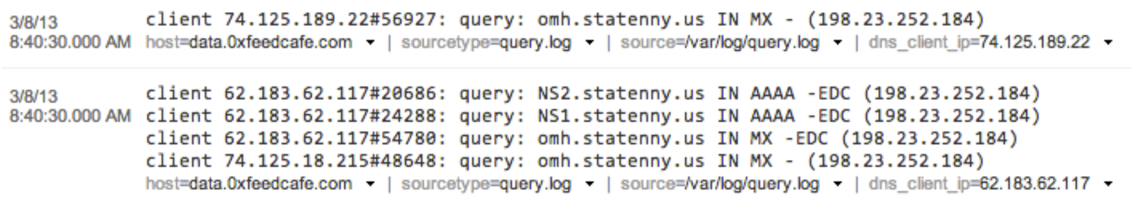
Figure 3. An example using the bitsquat domain 'snytimg.com"



```
3/12/13          [Tue Mar 12 11:47:18 2013] [error] [client 77.28.78.34] File does not exist: /var/www/yts, referer:
6:47:18.000 AM   http://www.youtube.com/results?search_query=will.i.am+ft+britney+spears+scream+and+shout&oq=Will.I.A
                 reduced.1.0.0l4.123187.123187.0.125978.1.1.0.0.0.0.284.284.2-1.1.0...0.0...1ac.1.h79oQXeA9s4
                 host=data.0xfeedcafe.com ▾ | sourcetype=apache_error ▾ | source=/var/log/apache2/error.log ▾

3/12/13          77.28.78.34 - - [12/Mar/2013:11:47:18 +0000] "GET /yts/img/pixel-vfl3z5WfW.gif HTTP/1.1" 404 516
6:47:18.000 AM   "http://www.youtube.com/results?search_query=will.i.am+ft+britney+spears+scream+and+shout&oq=Will.I.
                 reduced.1.0.0l4.123187.123187.0.125978.1.1.0.0.0.0.284.284.2-1.1.0...0.0...1ac.1.h79oQXeA9s4" "Mozil
                 AppleWebKit/537.1 (KHTML, like Gecko) Chrome/21.0.1180.89 Safari/537.1" "snytimg.com"
                 host=data.0xfeedcafe.com ▾ | sourcetype=access_combined ▾ | source=/var/log/apache2/access.log ▾

3/12/13          client 62.162.32.10#32839: query: snytimg.com IN A -ED (198.23.252.184)
6:47:18.000 AM   host=data.0xfeedcafe.com ▾ | sourcetype=query.log ▾ | source=/var/log/query.log ▾ | dns_client_ip=62.162.32.10 ▾
```

Even less popular domains are susceptible to these subdomain delimiter bitsquatting techniques. Below are some example DNS requests meant for the State of New York's domain: state.ny.us. Given that the .us TLD is also available for general public registration, it makes little sense for government organizations to use these TLDs because of bitsquatting or malicious typosquatting possibilities. This attack against state.ny.us would not be as easy if the domain was hosted at .gov instead; the more restrictive .gov registration process shields organizations that are entitled to use it from casual attackers.

Figure 4. An example using the bitsquat domain "statenny.us"

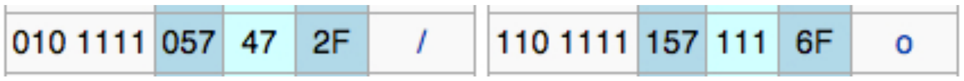


```
3/8/13           client 74.125.189.22#56927: query: omh.statenny.us IN MX - (198.23.252.184)
8:40:30.000 AM   host=data.0xfeedcafe.com ▾ | sourcetype=query.log ▾ | source=/var/log/query.log ▾ | dns_client_ip=74.125.189.22 ▾

3/8/13           client 62.183.62.117#20686: query: NS2.statenny.us IN AAAA -EDC (198.23.252.184)
8:40:30.000 AM   client 62.183.62.117#24288: query: NS1.statenny.us IN AAAA -EDC (198.23.252.184)
                 client 62.183.62.117#54780: query: omh.statenny.us IN MX -EDC (198.23.252.184)
                 client 74.125.18.215#48648: query: omh.statenny.us IN MX - (198.23.252.184)
                 host=data.0xfeedcafe.com ▾ | sourcetype=query.log ▾ | source=/var/log/query.log ▾ | dns_client_ip=62.183.62.117 ▾
```

URL delimiter squatting – "/" and "o"

Another useful technique for identifying potential bitsquat domains is to consider not only the valid characters in the domain names themselves, but also to consider the context in which a domain name might appear. One very popular context for domain names is within a URL. Inside a typical URL, forward slash characters "/" will act as a delimiter separating the scheme from the hostname from the URL path. The forward slash character (binary 0**0**101111) can by the flip of one bit become the letter "o" (binary 0**1**101111), and vice-versa.

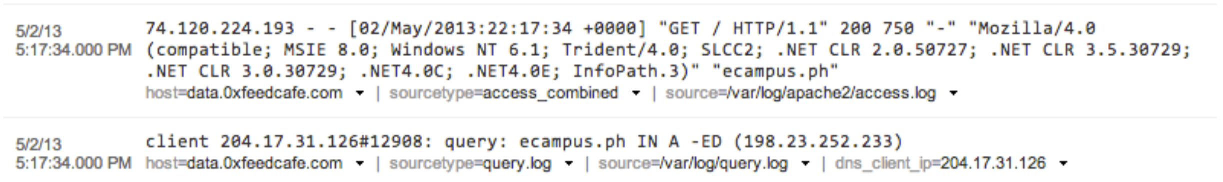
Figure 5. A comparison of the ASCII representation of the forward slash '/' versus the letter 'o'



010 1111	057	47	2F	/	110 1111	157	111	6F	o

The first bitsquatting technique in this category relies on the letter "o" inside the target domain becoming a forward slash, effectively terminating the domain name. This form of bitsquat is possible whenever the letter "o" appears in a domain name, and the preceding characters form a valid domain name. For an example, consider the URL **https://ecampus.phoenix.edu/**. If the letter "o" in the word "phoenix" is flipped to a "/" in memory, then the resulting corrupted URL will be **https://ecampus.ph/enix.edu/**. The traffic for that URL will be directed to the Philippines domain "ecampus.ph" instead of "phoenix.edu". Perhaps the most interesting aspect of this specific technique is that it works against target domains that are **registered under different, non-public gTLDs like ".edu", ".gov", or ".mil".**

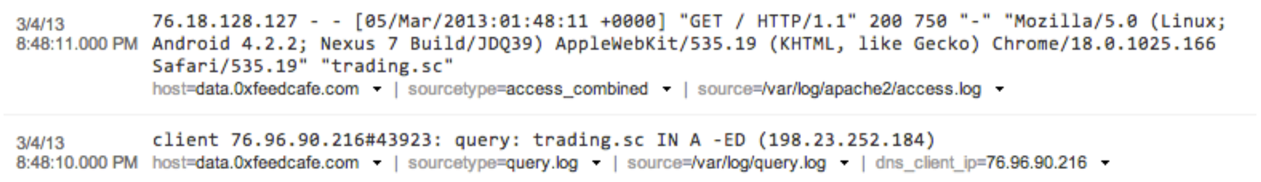
Figure 6. An example using the bitsquat domain "ecampus.ph"



```
5/2/13            74.120.224.193 - - [02/May/2013:22:17:34 +0000] "GET / HTTP/1.1" 200 750 "-" "Mozilla/4.0
5:17:34.000 PM    (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729;
                  .NET CLR 3.0.30729; .NET4.0C; .NET4.0E; InfoPath.3)" "ecampus.ph"
                  host=data.0xfeedcafe.com | sourcetype=access_combined | source=/var/log/apache2/access.log

5/2/13            client 204.17.31.126#12908: query: ecampus.ph IN A -ED (198.23.252.233)
5:17:34.000 PM    host=data.0xfeedcafe.com | sourcetype=query.log | source=/var/log/query.log | dns_client_ip=204.17.31.126
```

And here is another example of the same technique, this time stemming from the site "trading.scottrade.com":

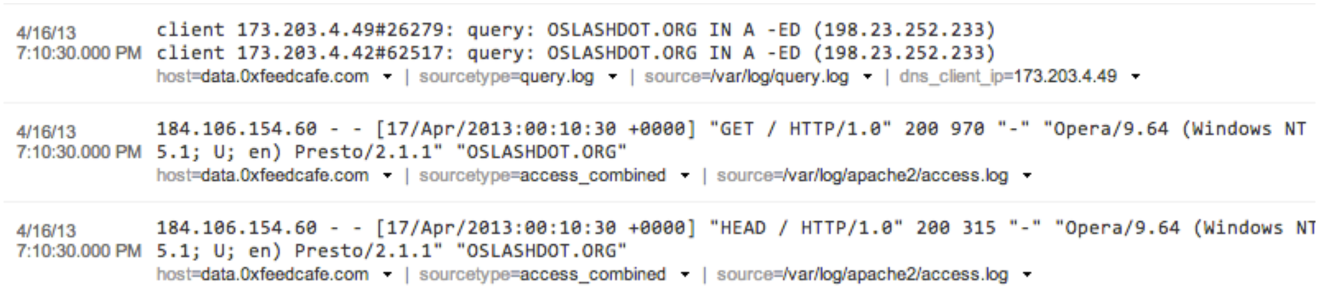
Figure 7. An example using the bitsquat domain "trading.sc"



```
3/4/13            76.18.128.127 - - [05/Mar/2013:01:48:11 +0000] "GET / HTTP/1.1" 200 750 "-" "Mozilla/5.0 (Linux;
8:48:11.000 PM    Android 4.2.2; Nexus 7 Build/JDQ39) AppleWebKit/535.19 (KHTML, like Gecko) Chrome/18.0.1025.166
                  Safari/535.19" "trading.sc"
                  host=data.0xfeedcafe.com | sourcetype=access_combined | source=/var/log/apache2/access.log

3/4/13            client 76.96.90.216#43923: query: trading.sc IN A -ED (198.23.252.184)
8:48:10.000 PM    host=data.0xfeedcafe.com | sourcetype=query.log | source=/var/log/query.log | dns_client_ip=76.96.90.216
```

The bidirectional nature of bits flipping means that the slashes that delimit the parts of the URL can also flip to become a letter "o", however only bit flips of the second or third slashes will produce a viable bitsquat. Bit flips of the second slash yield bitsquat domains when no 3rd level domain names are generally present. For example, if the second slash in the URL **http://slashdot.org/** flips a bit in memory it can become **http:/oslashdot.org/**. While that syntax is not a valid URL syntax, modern browsers helpfully correct the error in the double slash authority delimiter, and direct traffic to the bitsquat domain "oslashdot.org".

Figure 8. An example using the bitsquat domain "oslashdot.org"



```
4/16/13           client 173.203.4.49#26279: query: OSLASHDOT.ORG IN A -ED (198.23.252.233)
7:10:30.000 PM    client 173.203.4.42#62517: query: OSLASHDOT.ORG IN A -ED (198.23.252.233)
                  host=data.0xfeedcafe.com | sourcetype=query.log | source=/var/log/query.log | dns_client_ip=173.203.4.49

4/16/13           184.106.154.60 - - [17/Apr/2013:00:10:30 +0000] "GET / HTTP/1.0" 200 970 "-" "Opera/9.64 (Windows NT
7:10:30.000 PM    5.1; U; en) Presto/2.1.1" "OSLASHDOT.ORG"
                  host=data.0xfeedcafe.com | sourcetype=access_combined | source=/var/log/apache2/access.log

4/16/13           184.106.154.60 - - [17/Apr/2013:00:10:30 +0000] "HEAD / HTTP/1.0" 200 315 "-" "Opera/9.64 (Windows NT
7:10:30.000 PM    5.1; U; en) Presto/2.1.1" "OSLASHDOT.ORG"
                  host=data.0xfeedcafe.com | sourcetype=access_combined | source=/var/log/apache2/access.log
```

When no 3rd level subdomain is used, the bitsquat domain is formed by simply adding the letter “o” to the beginning of that 2nd level domain name. Domains that begin with the letter “o” are also at risk in a similar fashion. For if the URL **http://oreilly.com/** experiences a bit error in memory, and the leading letter “o” becomes a slash, then the resulting URL would be **http:///reilly.com/**. This is bad syntax, but yet again, the error in the double slash authority delimiter is in fact corrected by the browser, and the traffic directed to racle.com.

Finally, bit errors that corrupt the 3rd slash in a URL into a letter “o” are 100% dependent on the path in the URL to terminate in a valid domain name. For an example, consider a hypothetical URL such as:

```
http://www.example.com/cisco.com?stuff=1
```

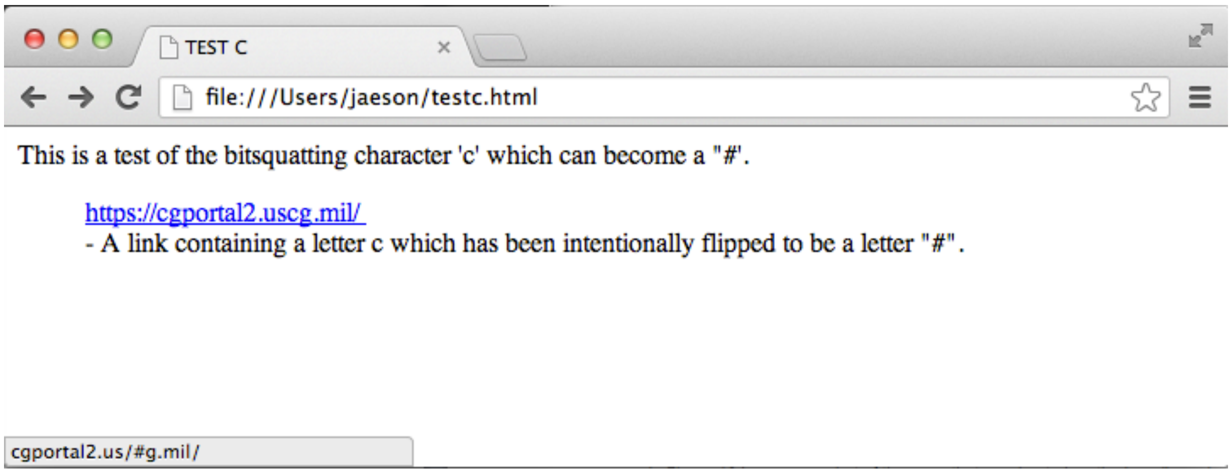
If the 3rd slash experiences a bit error and becomes a letter "o", the URL would instead read:

```
http://www.example.comocisco.com?stuff=1
```

This URL would direct its traffic to the bitsquat domain "comocisco.com". These types of bitsquats are exceedingly rare, but definitely possible if the URL had the right format and was popular enough.

URL delimiter squatting – “#” and “c”

When considering the other valid delimiter characters within a URL that might result in a bitsquat, we must also include the “#” character. Typically, inside a URL the pound character “#” will denote anchor tags within the current web page. It is possible for the letter “c” to change one binary digit to become the “#” character, and when this happens inside of a domain name it can create additional bitsquats. While strictly speaking the syntax is not valid, many browsers will helpfully correct the link, as indicated by the status bar at the bottom.

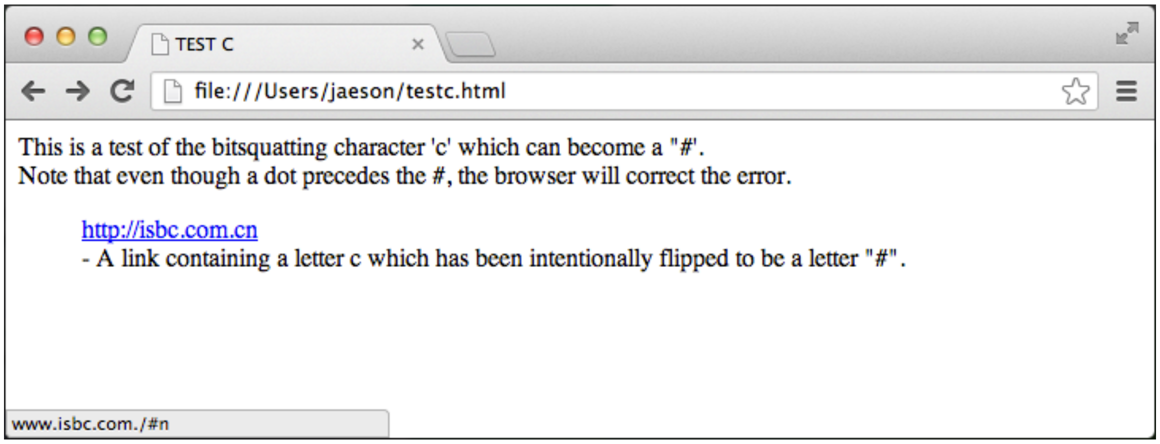


This is a test of the bitsquatting character 'c' which can become a "#'.

https://cgportal2.uscg.mil/
- A link containing a letter c which has been intentionally flipped to be a letter "#".

cgportal2.us/#g.mil/

Figure 9 Notice the hover link at the bottom. The traffic will not be directed to uscg.mil.



This is a test of the bitsquatting character 'c' which can become a "#'.
Note that even though a dot precedes the #, the browser will correct the error.

http://isbc.com.cn
- A link containing a letter c which has been intentionally flipped to be a letter "#".

www.isbc.com./#n

Figure 10 This time the c in .cn flips to a "#". Despite the trailing dot after "com" the bitsquat link still functions

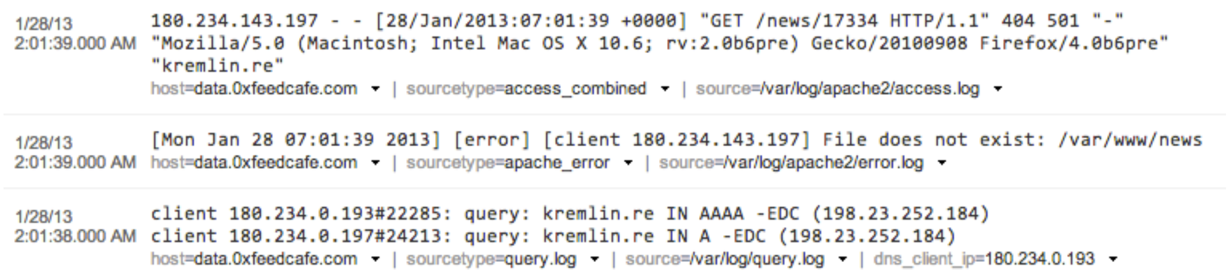
TLD bitsquatting

A search for bitsquats cannot be focused exclusively on 2nd level domain names. If bit errors can occur anywhere, then they can also occur inside the Top Level Domain (TLD) of a domain name. Most of the generic TLDs (gTLDs) have no bitsquats whatsoever, however there are two gTLDs that contain URL delimiter type bitsquats stemming from the presence of the letter "o". These are the gTLDs ".pro" and ".coop" with corresponding URL delimiter type bitsquats at the country code TLDs (ccTLDs): .pr (Puerto Rico) and .co (Colombia) respectively. Fortunately, the limited popularity of the .pro and .coop gTLDs inside URLs seems to preclude the possibility of finding many useful bitsquats in this space. So generally gTLDs are safe, but what about other TLDs? There happen to be several ccTLDs where bitsquats exist. It is interesting to note that some ccTLDs have no valid bitsquats while other ccTLDs have many. After surveying all valid Internet TLDs and checking the number of possible bitsquats, the following was found:

All 44 Internationalized Domain Name (IDN) TLDs are safe
4 ccTLDs are safe (nl –Netherlands, py –Paraguay, uy –Uruguay, za –S.Africa)
15 ccTLDs have one bitsquat (incl. uk –United Kingdom, hk –Hong Kong)
33 ccTLDs have two bitsquats (incl. us –United States, de –Germany, jp –Japan)
43 ccTLD have three bitsquats (incl. fr – France, no – Norway, va –Vatican
56 ccTLDs have four bitsquats (incl. ru –Russia, kr –South Korea)
43 ccTLDs have five bitsquats (incl. ca –Canada, it –Italy, eu –Europe)
37 ccTLDs have six bitsquats (incl. es –Spain, gr –Greece, in –India)
14 ccTLDs have seven bitsquats (incl. co –Colombia, ch –Switzerland)
2 ccTLDs have eight bitsquats (cm –Cameroon, cn –China)
1 ccTLD has nine bitsquats (cg –Republic of Congo)
1 ccTLD has ten bitsquats (ci –Ivory Coast)

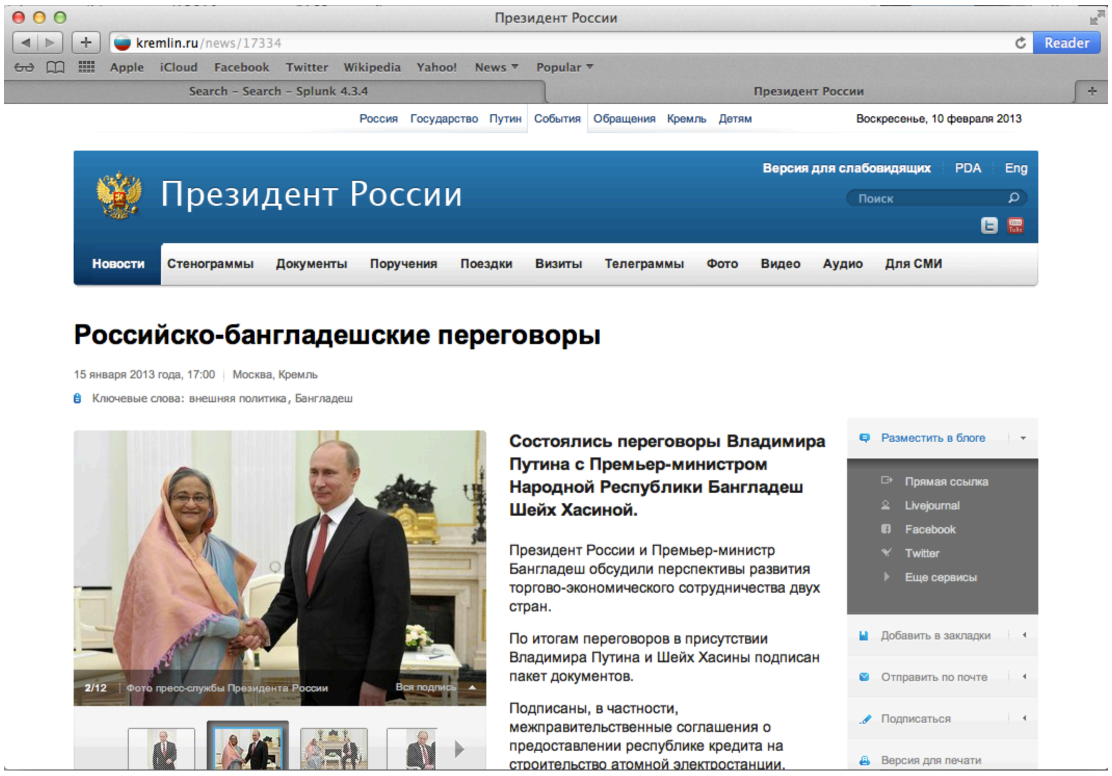
One ccTLD bitsquat that was registered and tested was a ccTLD bitsquat of the domain “kremlin.ru” (Russia). The bitsquat domain in this case is ‘kremlin.re’ (Reunion Island). Figure 9 is an example of a bitsquat http request and in Figure 10 is a screen shot of the page that was hosted on the kremlin.ru domain at the time.

Figure 11. An example using the bitsquat domain "kremlin.re"



```
1/28/13          180.234.143.197 - - [28/Jan/2013:07:01:39 +0000] "GET /news/17334 HTTP/1.1" 404 501 "-"
2:01:39.000 AM   "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.6; rv:2.0b6pre) Gecko/20100908 Firefox/4.0b6pre"
                 "kremlin.re"
                 host=data.0xfeedcafe.com | sourcetype=access_combined | source=/var/log/apache2/access.log

1/28/13          [Mon Jan 28 07:01:39 2013] [error] [client 180.234.143.197] File does not exist: /var/www/news
2:01:39.000 AM   host=data.0xfeedcafe.com | sourcetype=apache_error | source=/var/log/apache2/error.log

1/28/13          client 180.234.0.193#22285: query: kremlin.re IN AAAA -EDC (198.23.252.184)
2:01:38.000 AM   client 180.234.0.197#24213: query: kremlin.re IN A -EDC (198.23.252.184)
                 host=data.0xfeedcafe.com | sourcetype=query.log | source=/var/log/query.log | dns_client_ip=180.234.0.193
```

Figure 12. The intended web page at kremlin.ru.


Президент России
kremlin.ru/news/17334
Reader
Apple iCloud Facebook Twitter Wikipedia Yahoo! News Popular
Search – Search – Splunk 4.3.4
Президент России
Россия Государство Путин События Обращения Кремль Детям
Воскресенье, 10 февраля 2013
Президент России
Версия для слабовидящих PDA Eng
Поиск
Новости Стенограммы Документы Поручения Поездки Визиты Телеграммы Фото Видео Аудио Для СМИ
2/12 Фото пресс-службы Президента России Вся подпись
Разместить в блоге
Прямая ссылка
Livejournal
Facebook
Twitter
Еще сервисы
Добавить в закладки
Отправить по почте
Подписаться
Версия для печати

Российско-бангладешские переговоры

15 января 2013 года, 17:00 | Москва, Кремль

Ключевые слова: внешняя политика, Бангладеш

Состоялись переговоры Владимира Путина с Премьер-министром Народной Республики Бангладеш Шейх Хасиной.

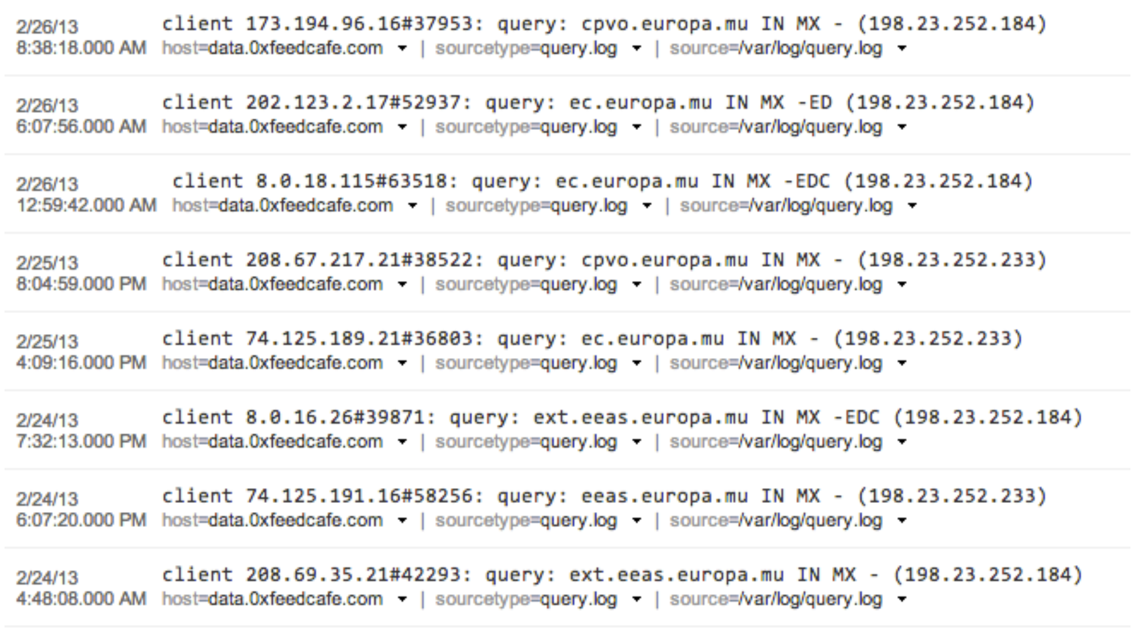
Президент России и Премьер-министр Бангладеш обсудили перспективы развития торгово-экономического сотрудничества двух стран.

По итогам переговоров в присутствии Владимира Путина и Шейх Хасины подписан пакет документов.

Подписаны, в частности, межправительственные соглашения о предоставлении республике кредита на строительство атомной электростанции.

An example of another bitsquat domain that was registered for which bitsquat-related requests were received is europa.mu. The domain europa.mu is one of the ccTLD bitsquat domains of europa.eu, a domain belonging to European Parliament. Figure 11 demonstrates some DNS MX requests received for subdomains of europa.eu.

Figure 13. An example using the bitsquat domain "europa.mu"



```
2/26/13          client 173.194.96.16#37953: query: cpvo.europa.mu IN MX - (198.23.252.184)
8:38:18.000 AM   host=data.0xfeedcafe.com ▾ | sourcetype=query.log ▾ | source=/var/log/query.log ▾

2/26/13          client 202.123.2.17#52937: query: ec.europa.mu IN MX -ED (198.23.252.184)
6:07:56.000 AM   host=data.0xfeedcafe.com ▾ | sourcetype=query.log ▾ | source=/var/log/query.log ▾

2/26/13          client 8.0.18.115#63518: query: ec.europa.mu IN MX -EDC (198.23.252.184)
12:59:42.000 AM  host=data.0xfeedcafe.com ▾ | sourcetype=query.log ▾ | source=/var/log/query.log ▾

2/25/13          client 208.67.217.21#38522: query: cpvo.europa.mu IN MX - (198.23.252.233)
8:04:59.000 PM   host=data.0xfeedcafe.com ▾ | sourcetype=query.log ▾ | source=/var/log/query.log ▾

2/25/13          client 74.125.189.21#36803: query: ec.europa.mu IN MX - (198.23.252.233)
4:09:16.000 PM   host=data.0xfeedcafe.com ▾ | sourcetype=query.log ▾ | source=/var/log/query.log ▾

2/24/13          client 8.0.16.26#39871: query: ext.eeas.europa.mu IN MX -EDC (198.23.252.184)
7:32:13.000 PM   host=data.0xfeedcafe.com ▾ | sourcetype=query.log ▾ | source=/var/log/query.log ▾

2/24/13          client 74.125.191.16#58256: query: eeas.europa.mu IN MX - (198.23.252.233)
6:07:20.000 PM   host=data.0xfeedcafe.com ▾ | sourcetype=query.log ▾ | source=/var/log/query.log ▾

2/24/13          client 208.69.35.21#42293: query: ext.eeas.europa.mu IN MX - (198.23.252.184)
4:48:08.000 AM   host=data.0xfeedcafe.com ▾ | sourcetype=query.log ▾ | source=/var/log/query.log ▾
```

Future gTLD Bitsquatting

Besides the bitsquatting that is possible using current TLDs, in 2013 ICANN is approving a large number of new gTLDs. Some of these proposed new gTLDs contain subdomain delimiter bitsquats for the entire TLD. Possessing one of these would allow the attacker to mount a bitsquat attack against all domains registered under the target gTLD.

.cleaning -> clea.ing (new gTLD .ing)
.exchange -> excha.ge (Georgia)
.helsinki -> helsi.ki (Kiribati)
.holdings -> holdi.gs (S.Georgia and S.Sandwich Islands)
.international -> internatio.al (Albania)
.tennis -> ten.is (Iceland)

Additionally, several of the proposed new gTLDs will have URL delimiter bitsquats in ccTLD space. Here is a list based on the letter “o”.

.boo -> .bo (Bolivia)
.bio -> .bi (Burundi)
.cooking -> .co (Colombia)
.cool -> .co (Colombia)
.cloud -> .cl (Chile)
.ecom -> .ec (Ecuador)
.food -> .fo (Faroe Islands)
.football -> .fo (Faroe Islands)
.global -> .gl (Greenland)
.kyoto -> .ky (Cayman Islands)
.ngo -> .ng (Nigeria)
.photo -> .ph (Philippines)
.photography -> .ph (Philippines)
.photos -> .ph (Philippines)
.prof -> .pr (Puerto Rico)
.property -> .pr (Puerto Rico)
.properties -> .pr (Puerto Rico)
.scot -> .sc (Seychelles)
.shop -> .sh (St. Helena)

Finally here is a list of several proposed new gTLDs that have URL delimiter bitsquats in ccTLD space, this time based on the bit flips of the letter “c” bit flipping into a “#”.

.rocks -> .ro (Romania)
.auction -> .au (Australia)
.doctor -> .do (Dominican Republic)
.accountant -> .ac (Ascension Island)
.archi -> .ar (Argentina)
.architect -> .ar (Argentina)
.recipes -> .re (Reunion Island)
.soccer -> .so (Somalia)
.inc -> .in (India)

Past Bitsquatting: Domainers the first to capitalize bitsquat domains

Looking at the whois records for some of bitsquat domains that have already been registered also yields some interesting findings. For example, the bitsquat domain wwwnfacebook.com was registered back in 2009, a full 2 years before the initial research paper on bitsquatting was published. The same is true for the domain “otwitter.com”. Thus some of the earliest bitsquat domain registrations have come from "domainers" -- organizations that register domain names to place ads or redirect traffic for profit. These domainers essentially noticed and capitalized on traffic destined for bitsquat domains long before any bitsquatting research was ever conducted. Domainers might show us just how popular a domain name must be in order to have a worthwhile number of bitsquat requests. There will be a threshold of domain popularity at which the domainers still make money off registration of the bitsquat domain due to wayward traffic. The tools used by domainers to analyze potential domains for purchase would also be quite valuable to potential bitsquatters as well.

Section II - Mitigation of bitsquatting attacks

The original research by Dinaburg suggested two possible mitigations. First was that self-registration of the bitsquat domain variants is one good way to remove the possibility of having your data misdirected. Second was the prescription to install ECC memory. Neither of these mitigations are optimal. The self registration can be costly to maintain, depending on the length of the domain name, and there is always the possibility that someone has already beaten you to the domain name. The prescription for ECC memory sounds nice on the surface, but in reality the entire base of installed devices would have to upgrade simultaneously for bitsquatting to be prevented worldwide.

The good news is that these are not the only techniques a network defender can use to protect their users from accidentally misdirecting their Internet traffic. There are additional techniques that can be used. With sufficient adoption, these techniques could actually eliminate the bitsquatting problem almost completely.

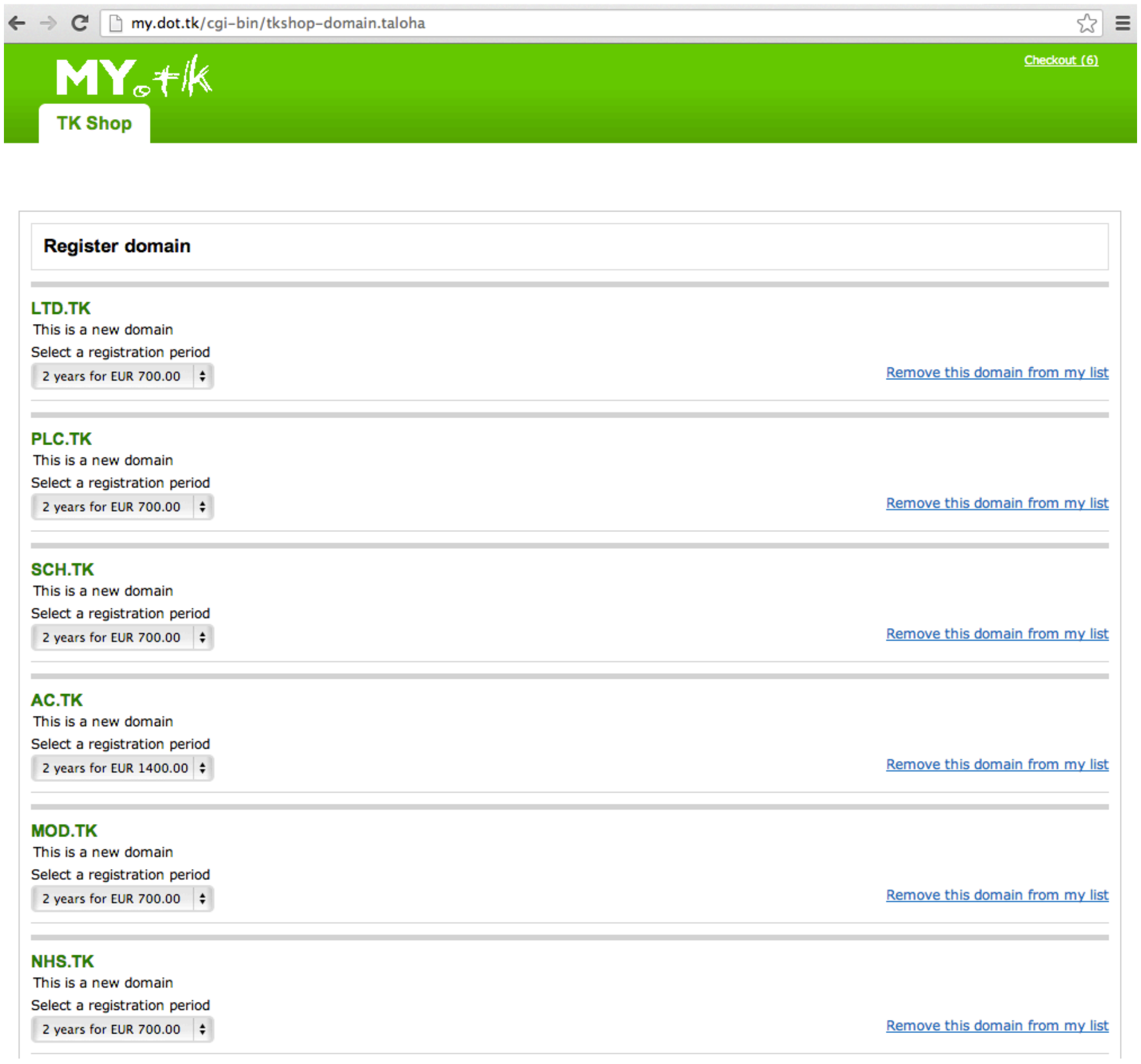
Choose a TLD which has no bitsquats

With the exception of the URL delimiter bitsquats available for .pro and .coop, there are no TLD bitsquats available for the currently available gTLDs or IDN TLDs (including the newly approved gTLDs from 2013).   So, they would all make excellent choices for eliminating potential bitsquats in the TLD. By choosing a domain at one of these TLDs you can effectively remove any possibility of a bit error in the TLD from misdirecting traffic.

If using a ccTLD, choose your domain name carefully

Having the ccTLD registry restrict the 2nd level domains that can be registered, like the ccTLD .uk (United Kingdom) does, is not necessarily an effective way to prevent bitsquats. In fact it can be even more dangerous. For only a few thousand dollars, one could register ltd.tk, plc.tk, sch.tk, ac.tk, mod.tk and tld.tk from Tokelau. Then the attacker will receive bitsquats from every domain registered under the corresponding second level domains ltd.uk, plc.uk, sch.uk, ac.uk, mod.uk and tld.uk. mod.uk corresponds with the UK's Ministry of Defense, and all the one bit errors occurring in that .uk ccTLD are going to a ***single*** location. Another ccTLD NIC with a similarly restrictive 2nd level domain policy is the ccTLD .br (Brazil). A domain like eng.cr is still available in Costa Rica, and that enables a bitsquatter to receive traffic from every single domain registered under eng.br.

Figure 14. A list of *available* domains in Tokelau which correspond to reserved 2nd level domains under .uk



Register domain

LTD.TK
This is a new domain
Select a registration period
2 years for EUR 700.00
Remove this domain from my list

PLC.TK
This is a new domain
Select a registration period
2 years for EUR 700.00
Remove this domain from my list

SCH.TK
This is a new domain
Select a registration period
2 years for EUR 700.00
Remove this domain from my list

AC.TK
This is a new domain
Select a registration period
2 years for EUR 1400.00
Remove this domain from my list

MOD.TK
This is a new domain
Select a registration period
2 years for EUR 700.00
Remove this domain from my list

NHS.TK
This is a new domain
Select a registration period
2 years for EUR 700.00
Remove this domain from my list

Fortunately, many ccTLDs that might be good locations for registering bitsquat domains do not allow certain common keywords (such as “www”, “gov”, etc.) to be registered, or do not allow 2nd level domains shorter than 3 characters, making these types of names good choices for use as 3rd level subdomains, and good protection against the URL delimiter bitsquatting techniques described in Section I.   There are also several other ccTLDs with restrictions such as local presence or citizenship in a particular country [7]. Though not impossible, these restrictions complicate registration of certain bitsquat variants.

Change/rotate subdomains frequently – the moving target defense

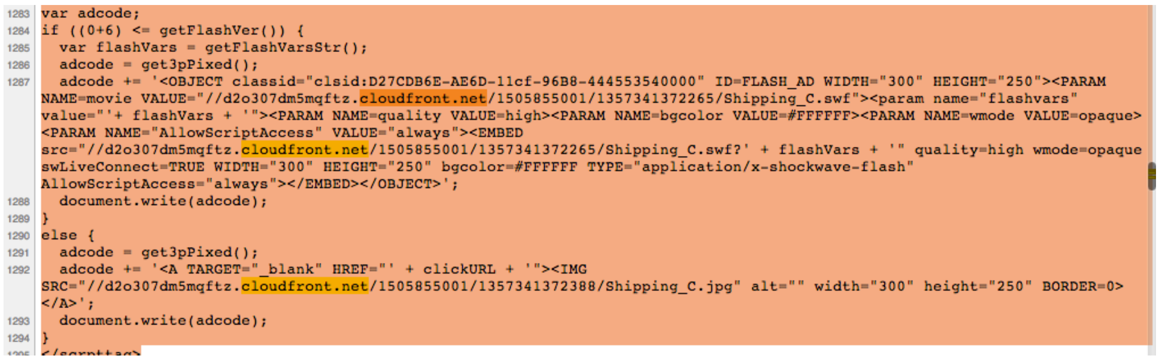
Both the domain delimiter and URL delimiter bitsquatting attack vectors can make use of a domain's 3rd level domain name label. Clever choice and use of 3rd level subdomains can thwart attempts by bitsquatters who use bitsquatting techniques targeted at 3rd level domain names.

If a 2nd level domain eliminates entirely its use of 3rd level subdomains (a.k.a. "naked" domains), then registering a URL delimiter bitsquat in a ccTLD, and registering a domain delimiter bitsquat using a 3rd level subdomain are both impossible. This does, however, expose you to URL delimiter bitsquats based off of the second slash of a URL, plus an additional bitsquat if your domain happens to begin with the letter "o". As of December 2012 the team from no-www.org have catalogued 60,000 domains that do not use 3rd level subdomains [8]. While eliminating use of subdomains helps eliminate some of the new attacks, there are actually even better mitigations.

A more effective technique is to subdivide your 2nd level domain traffic among a large number of 3rd level domains. Each subdomain takes on a small slice of the overall potential bitsquat traffic and therefore becomes much less likely to result in a successful bitsquat attack. Using a large number of subdomains creates much more work and expense for a potential bitsquat attacker. If next, those subdomains are changed or updated with any frequency, the bitsquatter will have practically no chance at a successful attack.

For a real world example, consider amazon.com. Amazon includes in their web pages content from a domain named cloudfront.com. The 3rd level domain names here normally would make great URL delimiter bitsquats because the "o" in cloudfront yields a valid ccTLD in .cl (Chile). Although this would seem at first to be a great target for a bitsquat, Amazon changes the subdomain at cloudfront.com frequently enough, that this thwarts attempts to capitalize on bitsquat traffic. By changing the 3rd level domain name frequently enough Amazon leaves a very small window of time in which to set-up and collect bitsquat traffic. This particular technique is actually the most effective protection against both domain delimiter and URL delimiter based bitsquat attacks.

Figure 15. Example code from amazon.com showing potential URL delimiter bitsquats at cloudfront.net

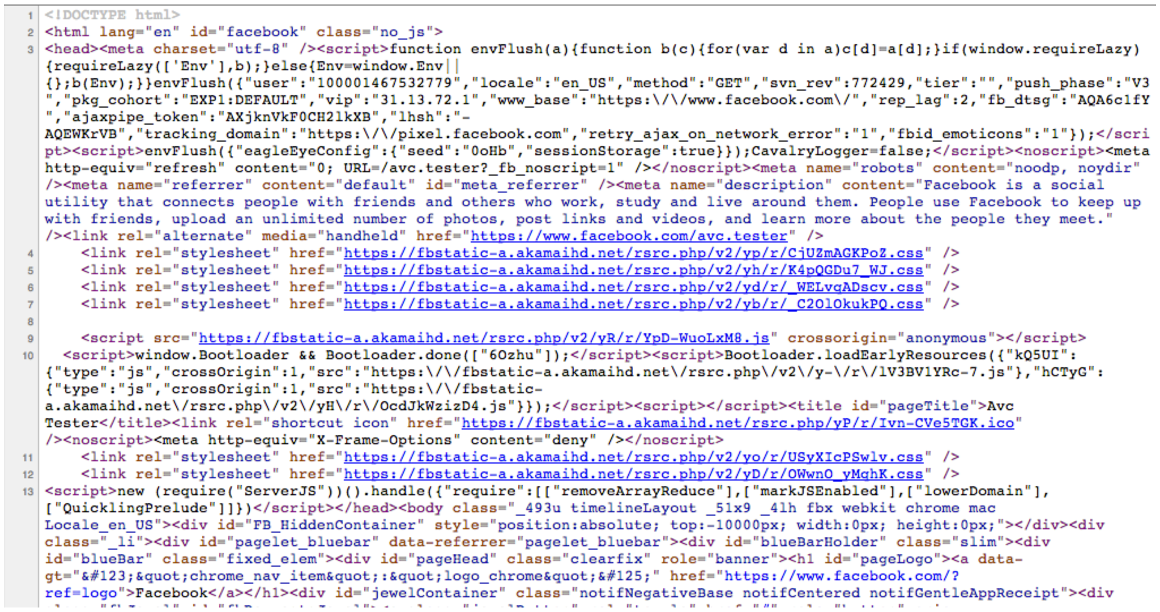


```
var adcode;
if ((0+6) <= getFlashVer()) {
  var flashVars = getFlashVarsStr();
  adcode = get3pPixed();
  adcode += '<OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" ID=FLASH_AD WIDTH="300" HEIGHT="250"><PARAM
NAME=movie VALUE="//d2o307dm5mqftz.cloudfront.net/1505855001/1357341372265/Shipping_C.swf"><param name="flashvars"
value="'+ flashVars + '"><PARAM NAME=quality VALUE=high><PARAM NAME=bgcolor VALUE=#FFFFFF><PARAM NAME=wmode VALUE=opaque>
<PARAM NAME="AllowScriptAccess" VALUE="always"><EMBED
src="//d2o307dm5mqftz.cloudfront.net/1505855001/1357341372265/Shipping_C.swf?' + flashVars + '" quality=high wmode=opaque
swLiveConnect=TRUE WIDTH="300" HEIGHT="250" bgcolor=#FFFFFF TYPE="application/x-shockwave-flash"
AllowScriptAccess="always"></EMBED></OBJECT>';
  document.write(adcode);
}
else {
  adcode = get3pPixed();
  adcode += '<A TARGET="_blank" HREF="' + clickURL + '"><IMG
SRC="//d2o307dm5mqftz.cloudfront.net/1505855001/1357341372388/Shipping_C.jpg" alt="" width="300" height="250" BORDER=0>
</A>';
  document.write(adcode);
}
</scripttag>
```

Use relative instead of absolute references in HTML

Bit errors, being indiscriminate as to where they occur, will affect domain names that are frequently loaded / accessed from memory. Thus to reduce the exposure to any potential URL delimiter bitsquats, it is best if the links and content loaded from HTML pages is referenced in a relative fashion instead of an absolute fashion. By using the current URL as a base href or specifically establishing a base href for an HTML page, the relative hrefs contained in the rest of the HTML document will eliminate many of the places where bitsquats might occur. The domain name will appear only once per HTML page. The downside here is that if a bit error does occur in the base href, then all links in the document would go to the same bitsquat domain. Figure 14 shows some of the HTML the source of the facebook.com website. Facebook seem to go out of their way to include an absolute link in each href.

Figure 16. Some source code from facebook.com website



```
<!DOCTYPE html>
<html lang="en" id="facebook" class="no_js">
<head><meta charset="utf-8" /><script>function envFlush(a){function b(c){for(var d in a)c[d]=a[d];}if(window.requireLazy)
{requireLazy(['Env'],b);}else{Env=window.Env||
{};b(Env);}}envFlush({"user":"100001467532779","locale":"en_US","method":"GET","svn_rev":772429,"tier":"","push_phase":"V3
","pkg_cohort":"EXP1:DEFAULT","vip":"31.13.72.1","www_base":"https:\/\/www.facebook.com\/","rep_lag":2,"fb_dtsg":"AQA6c1fY
","ajaxpipe_token":"AXjknVkF0CH21kXB","lhsh":"-
AQEWKrVB","tracking_domain":"https:\/\/pixel.facebook.com","retry_ajax_on_network_error":"1","fbid_emoticons":"1"});</scri
pt><script>envFlush({"eagleEyeConfig":{"seed":"0oHb","sessionStorage":true}});CavalryLogger=false;</script><noscript><meta
http-equiv="refresh" content="0; URL=/avc.tester?_fb_noscript=1" /></noscript><meta name="robots" content="noodp, noydir"
/><meta name="referrer" content="default" id="meta_referrer" /><meta name="description" content="Facebook is a social
utility that connects people with friends and others who work, study and live around them. People use Facebook to keep up
with friends, upload an unlimited number of photos, post links and videos, and learn more about the people they meet."
/><link rel="alternate" media="handheld" href="https://www.facebook.com/avc.tester" />
    <link rel="stylesheet" href="https://fbstatic-a.akamaihd.net/rsrc.php/v2/yp/r/CjUZmAGKPoZ.css" />
    <link rel="stylesheet" href="https://fbstatic-a.akamaihd.net/rsrc.php/v2/yh/r/K4pQGDu7_WJ.css" />
    <link rel="stylesheet" href="https://fbstatic-a.akamaihd.net/rsrc.php/v2/yd/r/_WELvqADscv.css" />
    <link rel="stylesheet" href="https://fbstatic-a.akamaihd.net/rsrc.php/v2/yb/r/_C2OlOkukPQ.css" />

    <script src="https://fbstatic-a.akamaihd.net/rsrc.php/v2/yR/r/YpD-WuoLxM8.js" crossorigin="anonymous"></script>
  <script>window.Bootloader && Bootloader.done(["6Ozhu"]);</script><script>Bootloader.loadEarlyResources({"kQ5UI":
{"type":"js","crossOrigin":1,"src":"https:\/\/fbstatic-a.akamaihd.net\/rsrc.php\/v2\/y-\/r\/lV3BV1YRc-7.js"},"hCTyG":
{"type":"js","crossOrigin":1,"src":"https:\/\/fbstatic-
a.akamaihd.net\/rsrc.php\/v2\/yH\/r\/OcdJkWzizD4.js"}});</script><script></script><title id="pageTitle">Avc
Tester</title><link rel="shortcut icon" href="https://fbstatic-a.akamaihd.net/rsrc.php/yP/r/Ivn-CVe5TGK.ico"
/><noscript><meta http-equiv="X-Frame-Options" content="deny" /></noscript>
    <link rel="stylesheet" href="https://fbstatic-a.akamaihd.net/rsrc.php/v2/yo/r/USyXIcPSwlv.css" />
    <link rel="stylesheet" href="https://fbstatic-a.akamaihd.net/rsrc.php/v2/yD/r/OWwnO_yMqhK.css" />
<script>new (require("ServerJS"))().handle({"require":[["removeArrayReduce"],["markJSEnabled"],["lowerDomain"],
["QuicklingPrelude"]]})</script></head><body class="_493u timelineLayout _51x9 _41h fbx webkit chrome mac
Locale_en_US"><div id="FB_HiddenContainer" style="position:absolute; top:-10000px; width:0px; height:0px;"></div><div
class="_li"><div id="pagelet_bluebar" data-referrer="pagelet_bluebar"><div id="blueBarHolder" class="slim"><div
id="blueBar" class="fixed_elem"><div id="pageHead" class="clearfix" role="banner"><h1 id="pageLogo"><a data-
gt="&#123;&quot;chrome_nav_item&quot;:&quot;logo_chrome&quot;&#125;" href="https://www.facebook.com/?
ref=logo">Facebook</a></h1><div id="jewelContainer" class="notifNegativeBase notifCentered notifGentleAppReceipt"><div
```

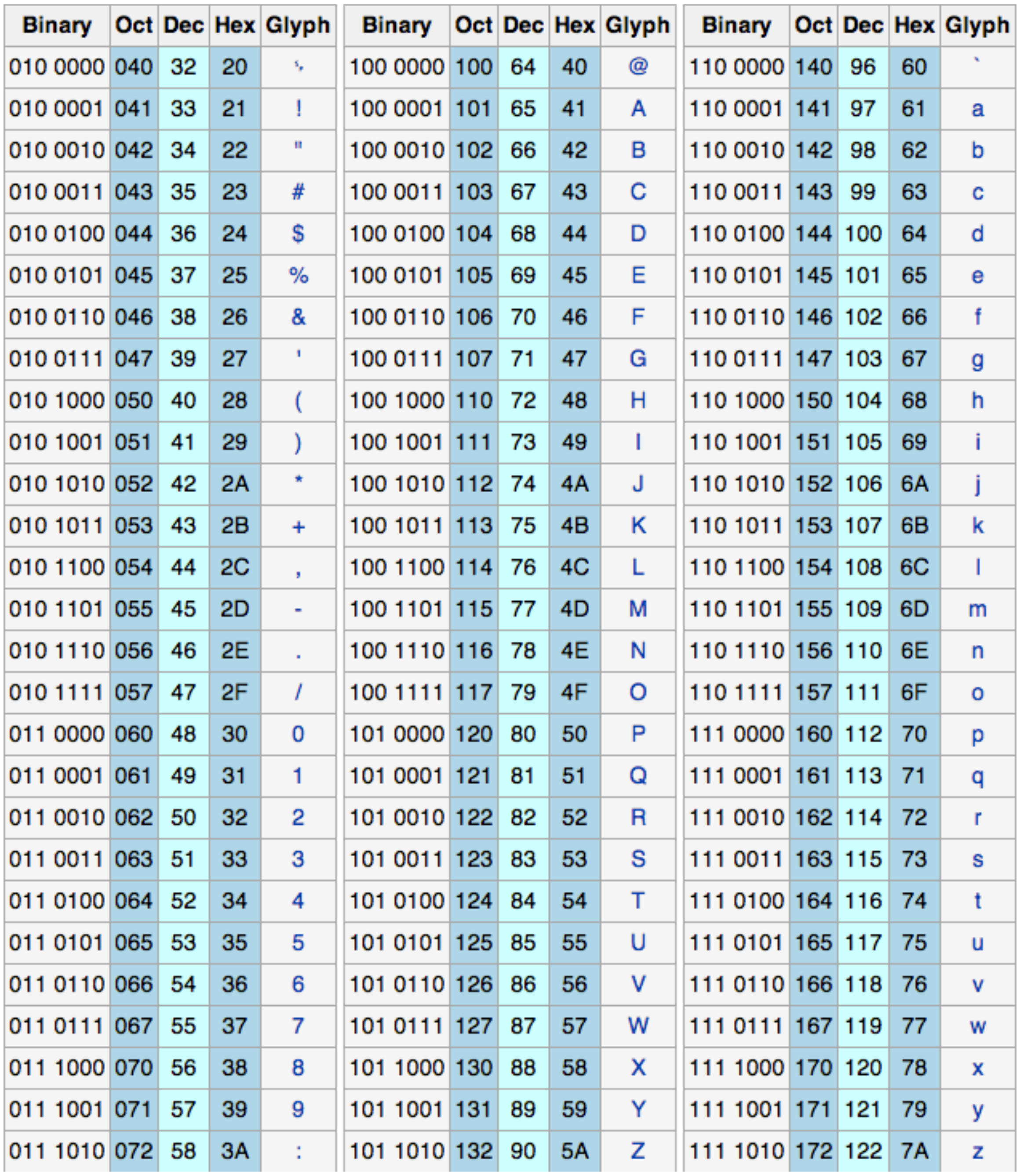
Use CAPITAL letters in URLs

The ASCII table is laid out so that the lowercase alphabet is one bit different from the uppercase alphabet. The capital letters 'A' (01**0**00001) through 'Z' (01**0**11010) differ by only one bit from their lowercase equivalents 'a' (01**1**00001) through 'z' (01**1**11010). However, bit-errors in lowercase 'p' (0**1**110000) through lowercase 'y' (0**1**111001) have bitsquats in the digit range zero (0**0**110000) through nine (0**0**111001). The uppercase versions do not possess these numeric bitsquats.

Capital letters are also immune to several punctuation-based bitsquats. The capital letter 'N'(0**10**01110) cannot via a single bit error become a dot '.' (0**01**01110). Neither can a capital letter 'O' (0**10**01111) flip one bit to become a forward slash (0**01**01111). Similarly, the capital letter 'C' (0**10**00011) cannot by the flip of one bit become a '#' (0**01**00011).

Thus storing capital letter versions of the domain names inside HTML pages makes a good choice for avoiding domain delimiter, URL delimiter, as well as individual bitsquats involving lowercase letters changing to digits.

Figure 17. A view of the ASCII table which demonstrates the binary representations of characters and punctuation. Image from wikipedia.org.



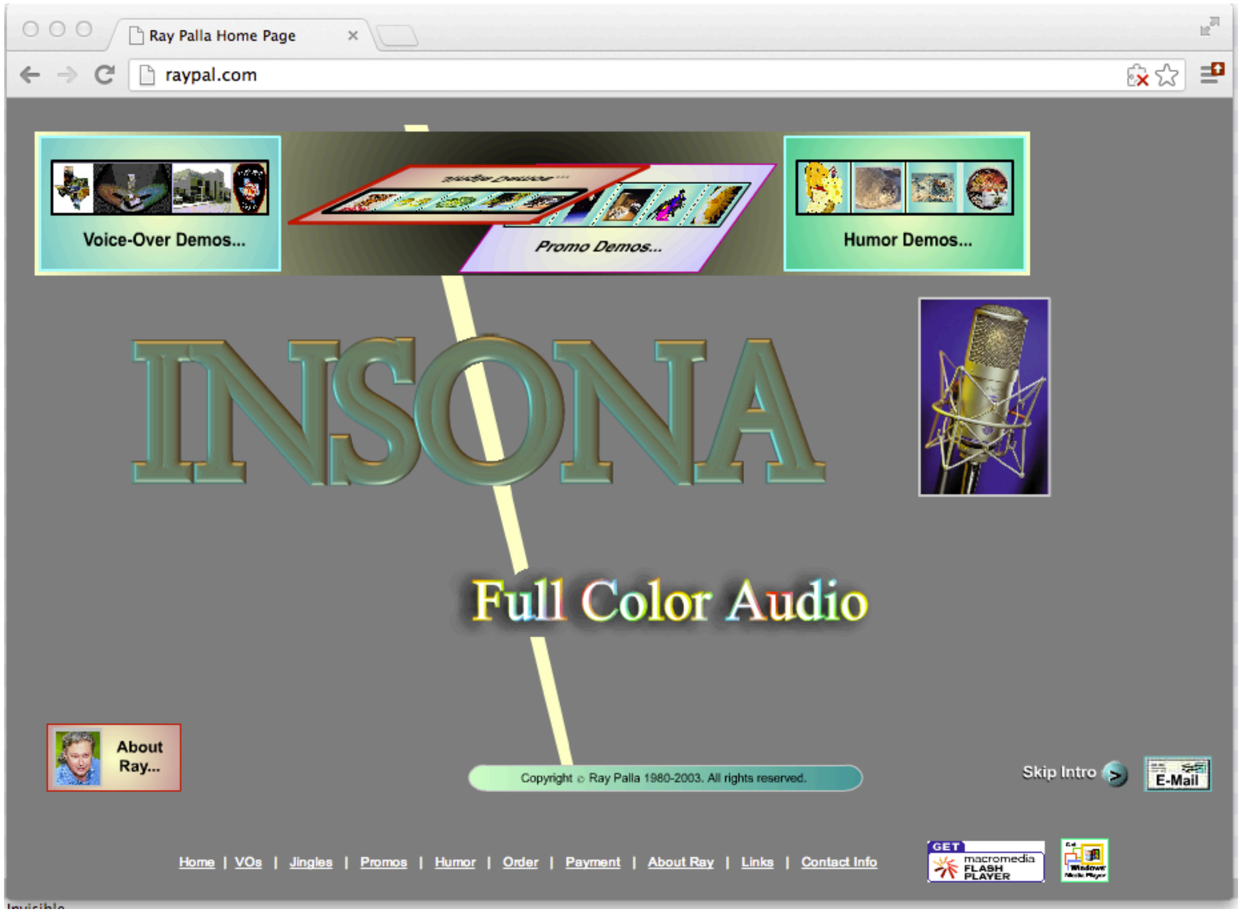
Binary	Oct	Dec	Hex	Glyph	Binary	Oct	Dec	Hex	Glyph	Binary	Oct	Dec	Hex	Glyph
010 0000	040	32	20	␠	100 0000	100	64	40	@	110 0000	140	96	60	`
010 0001	041	33	21	!	100 0001	101	65	41	A	110 0001	141	97	61	a
010 0010	042	34	22	"	100 0010	102	66	42	B	110 0010	142	98	62	b
010 0011	043	35	23	#	100 0011	103	67	43	C	110 0011	143	99	63	c
010 0100	044	36	24	$	100 0100	104	68	44	D	110 0100	144	100	64	d
010 0101	045	37	25	%	100 0101	105	69	45	E	110 0101	145	101	65	e
010 0110	046	38	26	&	100 0110	106	70	46	F	110 0110	146	102	66	f
010 0111	047	39	27	'	100 0111	107	71	47	G	110 0111	147	103	67	g
010 1000	050	40	28	(	100 1000	110	72	48	H	110 1000	150	104	68	h
010 1001	051	41	29	)	100 1001	111	73	49	I	110 1001	151	105	69	i
010 1010	052	42	2A	*	100 1010	112	74	4A	J	110 1010	152	106	6A	j
010 1011	053	43	2B	+	100 1011	113	75	4B	K	110 1011	153	107	6B	k
010 1100	054	44	2C	,	100 1100	114	76	4C	L	110 1100	154	108	6C	l
010 1101	055	45	2D	-	100 1101	115	77	4D	M	110 1101	155	109	6D	m
010 1110	056	46	2E	.	100 1110	116	78	4E	N	110 1110	156	110	6E	n
010 1111	057	47	2F	/	100 1111	117	79	4F	O	110 1111	157	111	6F	o
011 0000	060	48	30	0	101 0000	120	80	50	P	111 0000	160	112	70	p
011 0001	061	49	31	1	101 0001	121	81	51	Q	111 0001	161	113	71	q
011 0010	062	50	32	2	101 0010	122	82	52	R	111 0010	162	114	72	r
011 0011	063	51	33	3	101 0011	123	83	53	S	111 0011	163	115	73	s
011 0100	064	52	34	4	101 0100	124	84	54	T	111 0100	164	116	74	t
011 0101	065	53	35	5	101 0101	125	85	55	U	111 0101	165	117	75	u
011 0110	066	54	36	6	101 0110	126	86	56	V	111 0110	166	118	76	v
011 0111	067	55	37	7	101 0111	127	87	57	W	111 0111	167	119	77	w
011 1000	070	56	38	8	101 1000	130	88	58	X	111 1000	170	120	78	x
011 1001	071	57	39	9	101 1001	131	89	59	Y	111 1001	171	121	79	y
011 1010	072	58	3A	:	101 1010	132	90	5A	Z	111 1010	172	122	7A	z

Create a bitsquat RPZ

Response Policy Zones (RPZs) have been a configuration option since BIND v9.8.1, but patches exist for earlier versions of BIND. An RPZ is a local zone file which allows the DNS resolver to respond to specific DNS requests by saying that the domain name does not exist (NXDOMAIN), or redirecting the user to a walled garden, or other possibilities. To mitigate the effects of single bit errors for users of a DNS resolver the resolver administrator can create a Response Policy Zone that protects against bitsquats of frequently resolved, or internal-only domain names. For example, the RPZ can be set up such that any requests made to the DNS resolver for bitsquat variants of these domains will get a NXDOMAIN response, silently "correcting" bit errors without any work on the part of the client experiencing the bit error. If a domain is unavailable to potential victims of a bitsquatting attack, then this removes much of the incentive for attackers to bitsquat a target domain.

The downside to configuring your DNS server in this manner is the possibility of False Positives (FPs). For example, I may be looking to buy a jingle from a man named Ray Palla who runs raypal.com. This domain also happens to be a one bit variant of the popular domain name paypal.com. If the DNS request for raypal.com results in a NXDOMAIN response, none of my users will ever be able to contact Ray. This isn't terribly fair to Ray. Careful consideration **must** be paid to the one bit variants blocked as a result of any local RPZ to prevent false positives.

Figure 18. A legitimate site, raypal.com, which happens to be one bit different from paypal.com


Ray Palla Home Page
raypal.com
Voice-Over Demos...
Promo Demos...
Humor Demos...
INSONA
Full Color Audio
About Ray...
Copyright © Ray Palla 1980-2003. All rights reserved.
Skip Intro
E-Mail
Home | VOs | Jingles | Promos | Humor | Order | Payment | About Ray | Links | Contact Info
GET macromedia FLASH PLAYER

Additionally, although it has not yet been confirmed in-the-wild, it is also technically possible to bitsquat IP addresses which are stored in memory. Given the shortage of IPv4 address space many networks have turned to the RFC 1918 private networks in 10.0.0.0/8, 192.168.0.0/16, and 172.16.0.0/12. The one bit variants of these IPs must be receiving intranet traffic from all over the world. It would be difficult to find and subsequently control the exact one bit variant IP, but this task is not impossible either. All one bit variants of the most critical intranet address space can be calculated beforehand, and afterwards added to a firewall DROP list such that IP based bitsquats do not also result in misdirected traffic by bypassing the RPZ/DNS.

Conclusion

While the evidence to date that suggests that there hasn't been a wide adoption of bitsquatting as a real-world attack vector that is being exploited, the fact that organizations belonging to the education, government, and military under restricted Top Level Domains can also be vulnerable to some bitsquatting attacks is alarming. The ease, and relative anonymity of which bitsquatting attacks can be conducted means that society collectively needs to take precautions to protect the critical domain name infrastructure that is used to provide essential services and information.

References

[1] Eric Fischer. '*The Evolution of Character Codes, 1874-1968*'. November 2002. http://www.transbay.net/~enf/ascii/ascii.pdf. Accessed April 2013.

[2] American Standards Association. '*American Standard Code for Information Interchange, ASA X3.4-1963*'. ANSI. June 17, 1963.

[3] Artem Dinaburg. '*Bitsquatting: DNS Hijacking without exploitation*'. Blackhat Technical Security Conference. August, 2011.

[4] Bianca Schroeder, Eduardo Pinheiro, and Wolf-Dietrich Weber, '*DRAM Errors in the Wild: A Large-Scale Field Study*'. Proceedings of the 11th International Joint Conference on Measurement and Modeling of Computer Systems (SIGMETRICS). June 2009.

[5] Dave Evans. '*Thanks to IoE the next decade looks positively "nutty"*'. Cisco Platform Blog. http://blogs.cisco.com/news/thanks-to-ioe-the-next-decade-looks-positively-nutty/. Accessed March 2013.

[6] Artem Dinaburg. '*Bitsquatting PCAP Analysis Part 3: Bit-error distribution*'. Artem Dinaburg's Blog. http://blog.dinaburg.org/2012/11/bitsquatting-pcap-analysis-part-3-bit.html. Accessed December 2012.

[7] ICANN Wiki. '*CcTLD*'. http://icannwiki.com/index.php/ccTLD. Accessed March 2013.

[8] No-WWW. '*www. Is deprecated.*'. http://no-www.org/. Accessed March 2013.