Bitsquatting

DNS Hijacking without Exploitation

Artem Dinaburg (artem.dinaburg@raytheon.com, artem@dinaburg.org)
Raytheon Company
July, 2011

1 Introduction

Computer hardware, especially RAM, can suffer from random errors that manifest as corruption of one or more bits. The causes of these errors range from manufacturing defects to environmental factors such as cosmic rays and overheating. While the probability of a single error is small, the total error amount in all RAM connected to the Internet is significant. Malicious attackers can exploit these random errors remotely.

This paper presents an attack called *bitsquatting* that leverages random errors and DNS to direct Internet traffic to attacker-controlled destinations. To prove the feasibility of bitsquatting, several frequently resolved domains were bitsquatted and all HTTP requests to the subsequent domains were logged. An analysis of six months of log data reveals that virtually every operating system and platform is affected. Fortunately, bitsquatting attacks are easy to mitigate.

The rest of this paper is as follows: Section 2 discusses computer hardware errors, the causes of those errors, and measured error rates. Previous use of bit errors to defeat security mechanisms is reviewed in Section 3. The bitsquatting attack, intuition behind it, and the experiment design is described in Section 4. Six months of bitsquatting logs from the experiment are analyzed in Section 5. Effective mitigations and countermeasures against bitsquatting attacks are presented in Section 6.

2 Computer Hardware Errors

Software writers implicitly assume computer hardware and other electronic components operate correctly. This assumption is not always true. Computer hardware, including various information transmission and storage media, does experience errors. One possible hardware error is a bit-error. When a bit-error occurs, one or more of the internal representations of a bit stored in hardware transition from a 0 to a 1, or from a 1 to a 0.

Bit-errors and other hardware errors occur in production hardware. Section 2.1 describes several causes of bit-errors, and documented cases of their occurrence. Section 2.2 summarizes previous studies measuring bit-error rates due to various causes. The use (or lack thereof) and effectiveness of error correcting codes is covered in Section 2.3.

2.1 Causes of Bit-errors

Manufacturing defects, contamination, operation outside environmental tolerances, and radiation are some of the documented causes of bit-errors in production hardware. There are several well-known examples of bit-errors occurring in the wild.

2.1.1 Manufacturing Defects and Contamination

A famous design and manufacturing defect of the Sun UltraSparc II CPU is described by Peter Baston in the SPARCproductDIRectory [**1**]. To summarize, servers using UltraSparc II CPUs would randomly fail for no apparent reason. The problem was eventually traced back to SRAM contaminated with an alpha radiation emitter during manufacturing. The radiation would cause bit-errors in the processor's cache, resulting in random data corruption and server crashes. The problem was exacerbated by the lack of error checking and correction on the processor's SRAM. Thousands of processors had to be replaced.

Anecdotal evidence of bad RAM abounds on the Internet. Some of these anecdotes are likely based in fact. Manufacturing defects in DRAM are common enough for applications like memtest86 to exist and to ship on every Ubuntu LiveCD.

2.1.2 Operating Outside Environmental Tolerances

Electrical resistance and conductance are affected by temperature [**2**]. Electronic components are rated to operate at given temperature ranges, humidity levels, and electrical characteristics. Modern desktops and servers rely on active cooling to maintain operational temperature. These cooling systems can fail, leading to operation outside environmental tolerances. Unstable and intermittent power supplies can violate electrical tolerances of electronic components. Researchers have used heat lamps to cause bit-errors in DRAM [**3**], and electrical glitching to induce bit-errors in portable devices and smartcards [**4**] [**5**].

Mobile devices are frequently used outdoors and are especially at risk of operating outside designed environmental tolerances. For example, the operating temperature requirements for several mobile phones are shown in Table 1. The operating temperature range for the iPhone 4 (32° F to 95° F) is not only occasionally exceeded outdoors in most of the world, but populated regions routinely experience climates well outside this range.

Phone	Minimum (°F)	Maximum (°F)	Reference
Apple iPhone 4	32	95	[6]
BlackBerry 8130 Pearl	32	122	[7]
BlackBerry 8700g	32	122	[8]
BlackBerry Bold 9700	32	104	[9]
HTC Imagio	32	104	[10]
Nokia E62	14	104	[11]
Samsung SGH-600	14	130	[12]
Siemens A65	14	130	[13]

Table 1: Mobile phone operating temperatures, in Fahrenheit. The maximums and minimums are routinely exceeded in many populated regions of the world.

2.1.3 Radiation

High-energy particles from cosmic rays can interact with electrical components. Far from being a remote phenomenon, some studies conclude that cosmic rays are actually the primary source of bit-errors in DRAM at ground level [**14**].

Cosmic rays are more prevalent at higher altitudes and at latitudes closer to the poles [**15**]. The direction of cosmic rays also varies; many more come from the vertical direction than the horizontal [**16**]. An IBM study determined that cosmic rays affect DRAM and found error rates at sites above 2600 feet to be five times that of ground level. Denver, CO (altitude 5280 ft.) had error rates ten times that of ground level [**17**]. The same study also concluded that DRAM orientation did not affect error rates, as the cross-section vulnerable to cosmic rays remained constant.

2.2 Bit-error Rates

The first reports of intermittent electronics failure occurred in 1954-1957, during aboveground nuclear testing [**17**]. In 1978 bit errors were first observed in commercially manufactured chips. That year May and Woods of Intel tracked down operational failures in 2107 series 16Kb DRAM chips to alpha particle emitter contamination during manufacture [**18**]. Since then numerous published studies on electronic circuit error rates have been conducted. Terrazon Semiconductor compiled a summary of these studies in 2004 [**19**]. Error rates from the Terrazon summary and from Google data centers [**20**] appear in Table 2.

Type of Memory	Failures In Test/Mbit	Hours until error (128MiB)	Source
Commercial CMOS Memory	4 million - 400 million	0.20	Terrazon
"some" 0.13 micron technologies	10,000 - 100,000	98	Terrazon
1Gbit of memory in 0.25 micron	6,000	160	Terrazon
4M SRAM	<4,200	230	Terrazon
1 Gbit of DRAM (Nite Hawk)	2,300	420	Terrazon
SRAM and DRAM	1,000 - 2,000	980	Terrazon
~8.2 Gbits of SRAM (CRAY YMP-8)	1,300	750	Terrazon
SRAM	1,000	980	Terrazon
256 MBytes	700	1400	Terrazon
160 Gbits of DRAM	700	1400	Terrazon
32 Gbits of DRAM (Cray YMP-8)	600	1600	Terrazon
MoSys 1T-SRAM (no ECC)	500	2000	Terrazon
Micron Estimate, 256 MBytes	120 - 240	8100	Terrazon
"ultra-low" failure rates	50 - 100	20000	Terrazon
Mfg 1, 1GB DIMM	35,000 - 59,000	28	Schroeder, et al.
Mfg 1, 2GB DIMM	7,800 - 18,000	130	Schroeder, et al.
Mfg 1, 4GB DIMM	4,100	240	Schroeder, et al.
Mfg 2, 1GB DIMM	20,000 - 26,000	49	Schroeder, et al.
Mfg 2, 2GB DIMM	9,900	99	Schroeder, et al.
Mfg 3, 1GB DIMM	81,000	12	Schroeder, et al.
Mfg 4, 1GB DIMM	16,000 - 34,000	61	Schroeder, et al.
Mfg 5, 2GB DIMM	36,000	27	Schroeder, et al.
Mfg 6, 2GB DIMM	13,000	75	Schroeder, et al.
Mfg 6, 4GB DIMM	11,000	89	Schroeder, et al.

Table 2: Memory failure rates according to Terrazon and Schroeder, et al. The Terrazon numbers are taken directly from the paper. The Schroeder FIT numbers only include the correctable errors encountered. The FIT numbers are calculated from the error rate per DIMM on Table 2 in the Schroeder publication.

Error rate is typically measured in Failures In Test (FIT), where one FIT is one failure in one billion operating hours. The highest failure rate for DRAM from the table is 81,000 FIT (1 error in 1.4 years) per megabit of DRAM, and the lowest error rate is 120 FIT (1 error in 950 years) per megabit of DRAM. These estimates seem very low, but modern devices have considerably more than 1 megabit of DRAM. A new consumer grade machine with 4GiB of DRAM, will encounter 3 errors a month, even assuming the **lowest** estimate of 120 FIT per megabit.

2.3 Error Correction and Detection

As a testament to the potential frequency and severity of hardware and transmission errors, computer hardware designers often include error detection features in their designs. Buses such as SCSI, PCI, and USB all use error detection. Computer processor caches and main memory may also use error detection and correction features. Information transmission protocols, such as Ethernet, UDP, TCP, and IP all include checksums to detect errors incurred due to faulty wiring, electromagnetic interference, or other disturbances.

Checksums, cyclic redundancy checks (CRC), and other error detection schemes are very effective at detecting bit-errors. While an additive checksum may miss an even number of bit-errors, a 32-bit CRC will detect 1-bit, 2-bit, and all odd amounts of bit-errors [**21**]. Despite their effectiveness, error-checking schemes are often unused. For example, the majority of commodity desktop PCs are sold without error checking and correction (ECC) RAM. Consumer grade mobile devices, even high-end devices such as the Apple iPhone, lack ECC DRAM [**22**]. Computer makers have at times also neglected to include error detection logic in microprocessor caches, such as the case of Sun Microsystems and the UltraSparc II line [**1**].

Even modern enterprise class systems, with error checking and correcting main memory, may not be immune from bit-errors. Such systems are vulnerable to bit-errors due to placement of non-ECC RAM in the path of even the least critical components: NICs, keyboard controllers, and hard drives. For example, hard drives come with as much as 64 MiB of (non-ECC) DRAM cache. Figure 1 shows a close-up of a 146GB 10K RPM enterprise class SAS drive from a major hardware vendor. The chip in the picture is a 128-megabit Samsung K4D263238I GDDR SDRAM, used for caching. After reading all available chip documentation, no mention of error checking or error correction could be found. Enterprise class systems typically include several such drives.

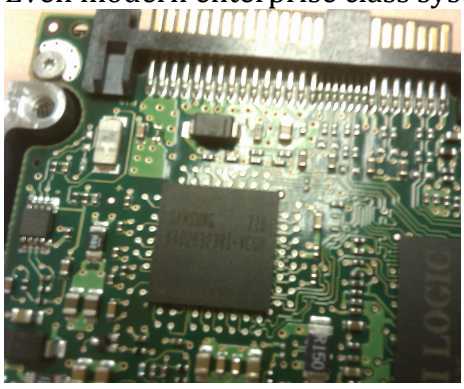

Figure 1: DRAM cache on an enterprise class hard drive from a major vendor. The chip is a 128 megabit GDDR SDRAM. No mention of error detection or correction could be found in the chip documentation.

3 Bit-errors and Security

The obvious danger from bit-errors is random data corruption. Data corruption itself can be considered a security compromise in certain situations. Bit-errors and other error injection attacks can also defeat security mechanisms. Security researchers, reverse engineers, and even criminals have used bit-errors to break deployed software security mechanisms, such as the Java sandbox, and various anti memory dumping technologies.

3.1 Java JVM Sandbox Escape

In 2003, Sudhakar Govindavajhala and Andrew W. Appel presented an innovative way to break out of the Java JVM sandbox [**3**]. To execute the attack, a malicious Java application creates two specially crafted objects: one designed to remain in a static location, and the other to fill up memory with references to the first object. When a bit-error occurs, it becomes possible to obtain two references of different types to the same object. With these references it is possible to write to arbitrary memory locations. Since waiting for an error could take an indeterminate amount of time, the actual demonstration used a heat lamp (shown in Figure 2) to cause bit-errors by heating a computer's DRAM to outside its temperature tolerance.

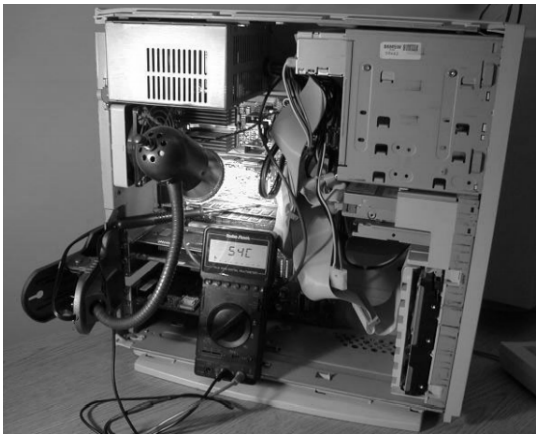

Figure 2: A heat lamp used by Govindavajhala and Appel to cause bit-errors in DRAM.

3.2 Attacks on Smartcards

Satellite TV pirates have used electrical glitching attacks to dump encryption keys from smartcard memory [**5**]. Glitching attacks work by either slowing the clock, creating a physical interruption, or applying electric changes to smartcard electronics. The introduction of these glitches causes bit-errors, which are leveraged to change control flow to a routine that will eventually dump ROM contents. Similar attacks have been used against handheld gaming systems [**4**]. These attacks are performed via direct physical manipulation, but the desired effect is bit-errors in electronics.

4 Bitsquatting

Prior to this research, bit-error attacks have only been used locally, where physical access to a target device is readily available. It is possible to use random bit-errors to attack remote targets over the Internet. The attack, called *bitsquatting*, relies on random bit-errors to redirect connections intended for popular domains.

Bitsquatting requires no exploitation or complicated reverse engineering, and is operating system and architecture agnostic. Experimental observations show that bitsquatting popular websites could redirect non-trivial amounts of Internet traffic to a malicious entity. This section will explain the intuition behind the bitsquatting attack, describe how a bitsquatting attack works, and introduce the bitsquatting experiment.

4.1 Intuition

It was estimated that in 2010, there were approximately 5 billion devices connected to the Internet [**23**] [**24**]. Many of these devices have RAM that is subject to bit errors. Given enough RAM and enough time, it is posited that one of these bit errors will manifest in an outgoing DNS query.

By applying conservative assumptions, it is possible to estimate the total amount of RAM that is subject to bit errors and connected to the Internet. The amount of RAM in all Internet connected devices ranges from a few megabytes in mobile phones, to multiple gigabytes in large servers and workstation machines. Assuming an average of 128 MiB of non-ECC RAM per device, the following calculation:

$$5{\times}10^{9}\ Devices \times 128\ MiB\ of\ RAM = 5.12{\times}10^{12}\ Megabits\ of\ RAM$$

gives a rough order of magnitude estimate for the amount of RAM connected to the Internet. Assuming the lowest-case FIT estimates of 120 (from Section 2.2) yields:

$$5.12{\times}10^{12}\ Megabits \times 120\ FIT = 614{,}400\ Errors/hour$$

The actual amount of bit-errors per hour may diverge greatly from the estimate, depending on how much non-ECC RAM is actually on the Internet and which FIT values are used.

4.2 Bitsquatting Attacks

To a remote attacker, the majority of the 600,000 bit-errors per hour will be completely useless. However, there is data in RAM where a single flipped bit can reliably lead to compromise: domain names. There are numerous places in RAM where domain names are stored during a web browsing session:

- Cached HTML in server memory
- Router memory during routing
- Caches in DNS servers
- Received HTML on the client
- Internal browser representation on the client
- On the client's `gethostbyname()` path
- In RAM-based cache on a disk drive

An undetected bit-error in any of these locations can cause a connection to a domain one bit different from the intended domain. Domains that are resolved very frequently are more likely to be in memory at the right time.

This realization leads to an attack called bitsquatting. To execute the bitsquatting attack, a malicious entity first registers domains one bit different from popular domain names; these are referred to as bitsquat domains. Second, a random bit error causes a domain name in memory to change to the bitsquat domain. A connection is then established to the bitsquat domain. At this point, the malicious entity can send phishing pages, browser exploits, steal cookies, or perform other malicious actions.

Since it is similar in spirit to typosquatting [**25**], the attack is called bitsquatting. While typosquatting relies on users to make mistakes, bitsquatting relies on computers to make mistakes. Although humans are more prone to mistakes, computers make considerably more DNS queries.

While the probability of a bit-error in just the right location to affect a DNS query is small, it is nonzero. An experiment was performed to determine the likelihood of such bit-errors.

4.3 Experiment

To determine whether bit-errors can redirect connections to attacker controlled sites, the bitsquat domains in Table 3 were registered, and all HTTP requests to the domains were logged. The domains, such as `li6e.com`, `mic2osoft.com`, and `fjcdn.net` are very unlikely to be typos or keyboard errors.

Some of the original domains were chosen because they are content delivery and advertising networks. These domains should be resolved often, but typed by users

very infrequently. Other domains, such as `microsoft.com`, were chosen based on worldwide popularity among both people and machines.

A DNS server was configured to answer all queries for the domains with two answers: one answer with the original domain, and one with the bitsquat domain. If a bit-error occurred somewhere along the DNS resolution path, a reply with the original could be seen as valid by the requestor.

An Apache HTTP server was configured to log all HTTP requests. No HTTP content was returned; instead an HTTP 404 error was presented for every requested URL. HTTP requests were logged from September 26, 2010 to May 5, 2011.

The HTTP server and DNS server were run on the same virtual machine hosted at a cloud computing provider, henceforth called the bitsquat server.

Bitsquat Domain	Original Domain
ikamai.net	akamai.net
aeazon.com	amazon.com
a-azon.com	amazon.com
amazgn.com	amazon.com
microsmft.com	microsoft.com
micrgsoft.com	microsoft.com
miarosoft.com	microsoft.com
iicrosoft.com	microsoft.com
microsnft.com	microsoft.com
mhcrosoft.com	microsoft.com
eicrosoft.com	microsoft.com
mic2osoft.com	microsoft.com
micro3oft.com	microsoft.com
li6e.com	live.com
0mdn.net	2mdn.net
2-dn.net	2mdn.net
2edn.net	2mdn.net
2ldn.net	2mdn.net
2mfn.net	2mdn.net
2mln.net	2mdn.net
2odn.net	2mdn.net
6mdn.net	2mdn.net
fbbdn.net	fbcdn.net
fbgdn.net	fbcdn.net
gbcdn.net	fbcdn.net
fjcdn.net	fbcdn.net
dbcdn.net	fbcdn.net
roop-servers.net	root-servers.net
doublechick.net	doubleclick.net
do5bleclick.net	doubleclick.net
doubleslick.net	doubleclick.net

Table 3: Bitsquat domains registered for the experiment.

5 Bitsquat Data Analysis

The biggest surprise from the bitsquatting experiment was that it actually worked. The HTTP logs were filled with computer-generated requests caused by random bit-errors. In this case, computer-generated means that the URLs were not typed into a web browser, but instead automatically inserted by an application and requested in the background. The computer-generated requests that appear in the HTTP log will be referred to as bitsquat requests. These bitsquat requests are best showcased by example; a sampling of the more interesting requests, such as those for Windows Updates, Webmail pages, and executable downloads appears in Appendix A.

During the logging period there were a total of 52,317 bitsquat requests from 12,949 unique IP addresses. This number excludes junk requests that are not a result of bit-errors, such as automated web vulnerability scanners, search engine crawlers, and other web spiders. The removal of junk requests was performed manually. It is possible that some bitsquat requests may have been removed, and a few junk requests may still remain in the data.

5.1 Daily Traffic Volume

Figure 3 shows the daily count of unique IP addresses appearing in bitsquat server HTTP logs. A steady trickle of background traffic is at times transformed into a torrent of requests, only to return to previous levels. These outlying events, labeled as A, B, and C in Figure 3 are described below.

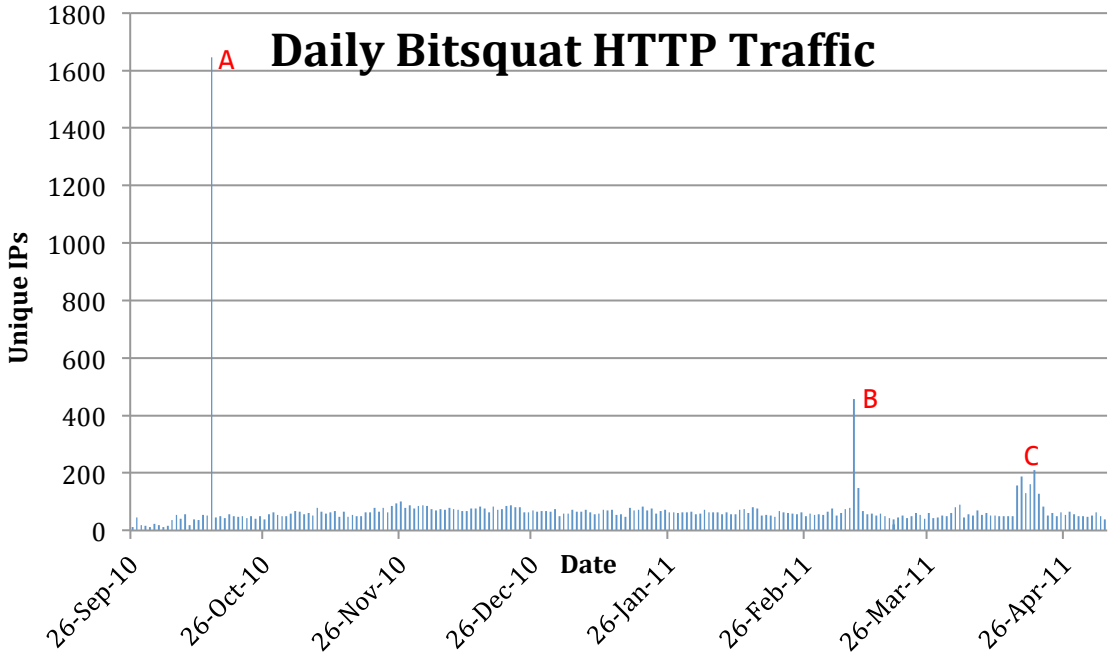

Figure 3: The amount of unique IPs requesting URLs from the bitsquat HTTP server, including all days with an unusually large amount of traffic.

5.1.1 Event A

Event A occurred on October 14th, 2010. The traffic volume for the day was 3,434 bitsquat requests originating from 1,645 unique IPs. The average day only has requests originating from 59 unique IPs. After manually examining the data, 2,555 requests from 1,227 unique IPs had the string "farmville" in the referring URL. All of these requests used only four domains in the HTTP `Host` header, as shown in Table 4. Three out of four domains contain more than one bit-error, ruling out a simple mistype of `fbbdn.net` for `fbcdn.net`. While the true cause is not known, the data points to bit-errors somewhere in the Zynga (the creators of Farmville) server farm.

Requests	Host Header
3	rtatic.ak.fbbdn.net
5	b.btatic.ak.fbbdn.net
36	btatic.ak.fbbdn.net
277	profile.ak.fbbdn.net
2234	pbofile.ak.fbbdn.net
2555	**Total**

Table 4: HTTP Host header contents for anomalous event A. The majority has more than one bit error, indicating that a mistype was unlikely.

5.1.2 Event B

Event B occurred From March 9th, 2011 to March 10th, 2011. During that time, there was a marked increase in the amount of bitsquat requests. The circumstances behind event B are similar to those of event A, but of a smaller magnitude. Once again there was a large percentage of "farmville" referrers, but the requested domain was `profile.ak.fbgdn.net`.

5.1.3 Event C

Event C is perhaps the most interesting of all three events. The upsurge in traffic from April 15th, 2011 to April 20th, 2011 had two things in common: the destination host of `s0.2mdn.net`, and the source network of `69.171.163.0/24`. Multiple source IP addresses were resolving a popular advertising domain to the bitsquat server. The bit error likely occurred at a central place: either the DNS resolver for `69.171.163.0/24`, or a transparent proxy for the network. The primary suspect for this event is DNS poisoning by way of bit-errors.

When outlying events are removed, as shown in Figure 4, there is still variance in daily connection volume. The low count was connections from 13 unique IPs on September 30th, and the high count was 100 unique IPs on November 26th. The average volume per day is 59 unique IPs, using only 30 bitsquat domains.

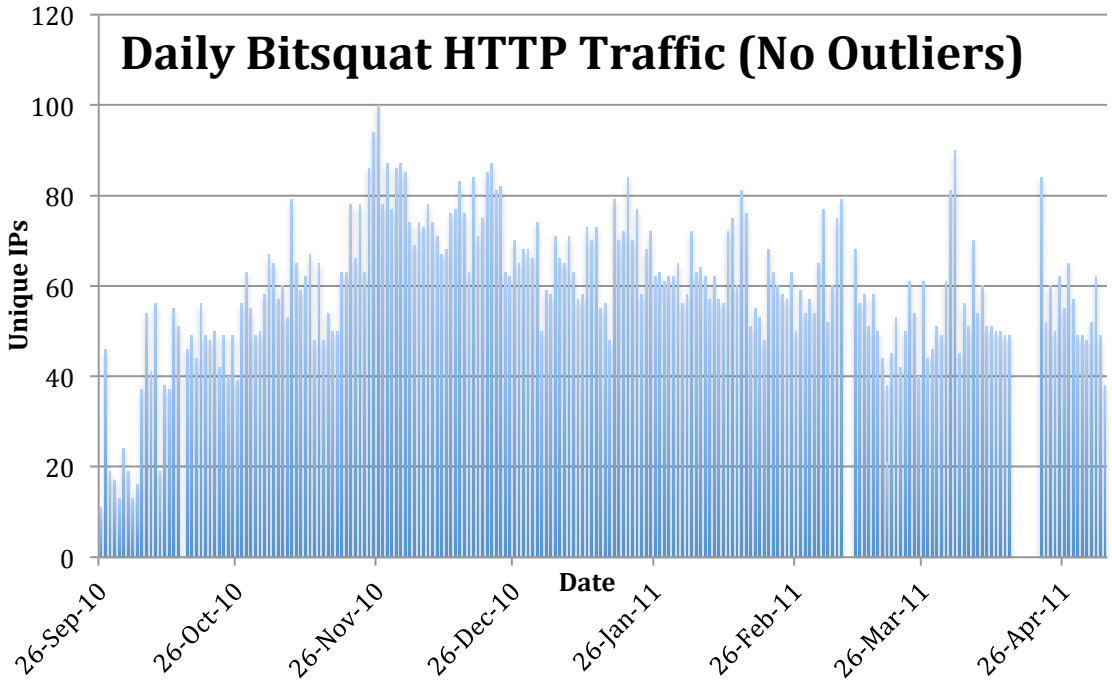

Figure 4: Daily amount of unique IPs connecting to the bitsquat server, with anomalous events removed for clarity.

5.2 Platforms/Operating Systems

Bitsquatting affects virtually every platform and operating system. Assuming the HTTP `User-Agent` strings in the log data are accurate, the bitsquat server saw traffic from x86, x86-64, ARM, PPC, and MIPS based devices running Windows, Linux, Mac OS, iOS, Symbian, and other operating systems.

To ascertain if a particular operating system is more vulnerable to memory errors than another, the operating systems from the bitsquat logs were compared to the operating systems of visitors to Wikipedia in March 2011 [**26**] (Figure 5).

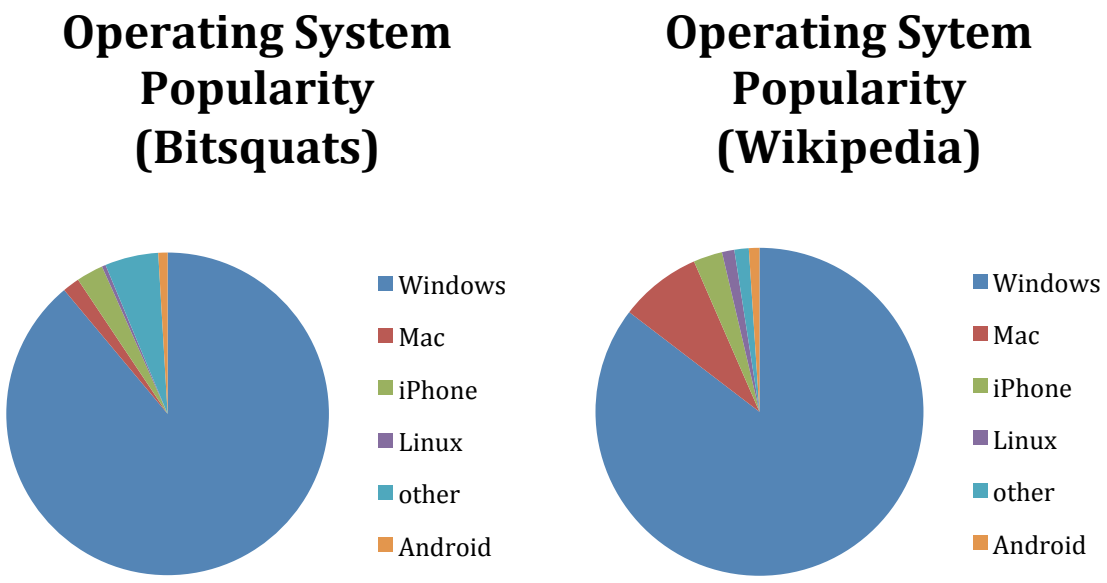

Figure 5: Operating systems seen in HTTP `User-Agent` headers of bitsquat traffic compared to Wikipedia visitors for May 2011. The bitsquat traffic contains fewer Macintosh computers, but considerably more mobile devices.

There are two major differences in the graphs: far fewer Macintoshes and far more "other" systems made bitsquat requests. The "other" operating system list is a long tail of various mobile devices and gaming consoles, including but not limited to: PlayStation 3, PlayStation Portable, BlackBerry phones, Symbian based devices, Nokia Phones, Nintendo Wiis, iPads, Samsung phones, SonyEricsson phones, and Motorola phones. Noticeably absent were Xbox gaming consoles.

5.3 Geographic Location

The majority of traffic to the bitsquat server was due to bitsquats of `fbcdn.net`, the Facebook content delivery network. Since the Facebook user base is

disproportionately high in the US as compared to the rest of the world, geographic location of all visitors makes a poor measurement.

`Microsoft.com`, however, should be resolved with approximately the same frequency throughout the world, due to the popularity of the Windows operating system. Figure 6 shows the originating country of all IPs making a `microsoft.com` bitsquat request.

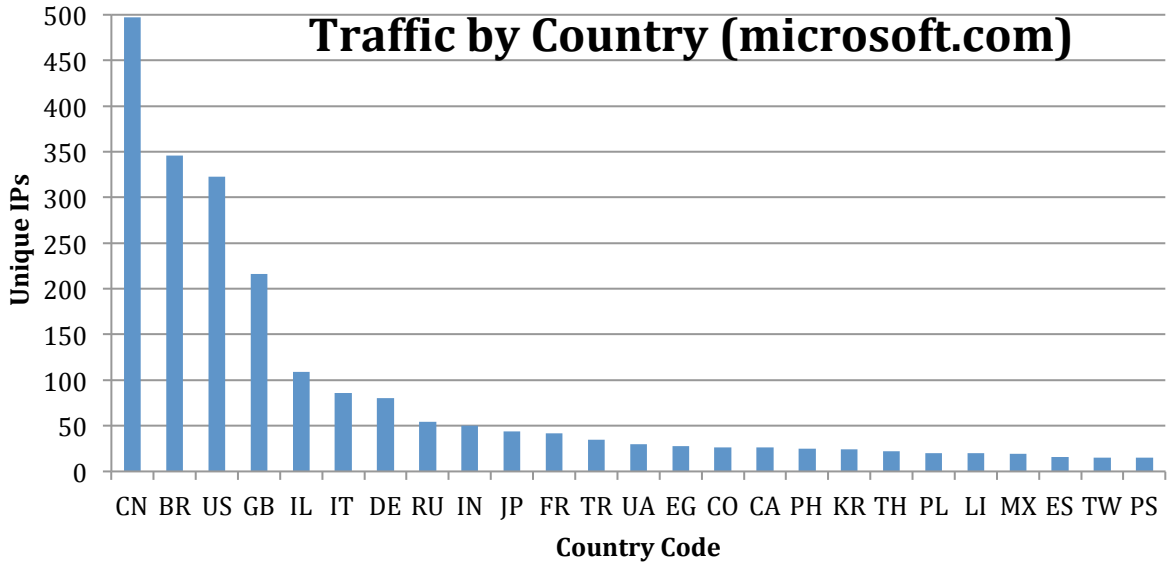

Figure 6: Source country for all IPs that used a microsoft.com bitsquat in their HTTP `Host` header. The IP to country mapping was done via the MaxMind GeoLite database.

The geographic location of each IP was identified using the MaxMind GeoLite database [**27**]. The graph neither matches population nor the amount of Internet users in the source countries. Currently no reasonable explanation for the discrepancies has been identified, and the discrepancies remain an interesting area for future research.

5.4 Where Do Bit-errors Occur?

As described in Section 2.2, there are numerous places a bit-error may occur, such as main RAM, disk cache, or during transmission. There are two main vulnerability paths for bit-errors that can be successfully leveraged in bitsquatting attacks, as shown in Figure 7.

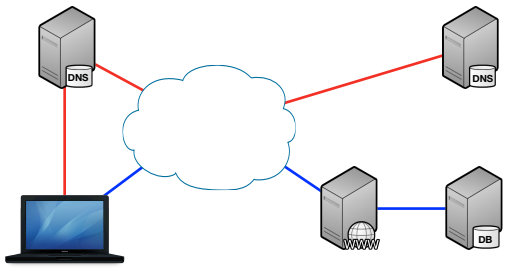

Figure 7: Two potential memory corruption paths leading to exploitation via bitsquatting. The red path represents DNS corruption, and the blue path represents application content corruption.

The red path represents bit-errors along the DNS resolution path, be it in transit, in host RAM, or in the RAM of a recursive resolver. The blue path represents bit-errors occurring prior to name resolution, such as those in application data like HTML.

The bitsquat experiment logs can differentiate between the two corruption paths by use of the HTTP `Host` header. The HTTP 1.1 specification requires the use of a `Host` header when making HTTP requests [**28**]. The `Host` header contains the domain the HTTP client resolved to connect to the bitsquat server. If the `Host` header matches the original domain, the corruption occurred on the red path (DNS path). If the `Host` header matches a bitsquat domain, the corruption occurred on the blue path (content path).

Figure 8 shows the path distribution of bitsquat experiment data. The vast majority of connections (96%) occurred via the content path. DNS corruption accounted for at least 3%, if not the full 4%. The "other" domains are either corrupted so badly that they match neither the bitsquat nor original domains, or they are completely unrelated domains, such as `apple.com`. These unrelated domains can be tracked to the use of long CNAME chains, and hence are a result of DNS corruption.

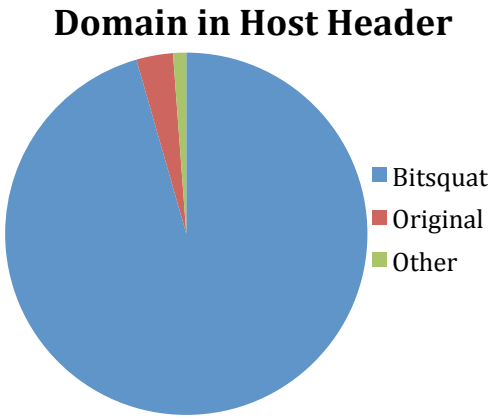

Figure 8: Domains seen in HTTP `Host` headers in 6 months of bitsquat request logs. The vast majority is corrupted prior to being resolved.

5.5 Domain Popularity

Bitsquats of `fbcdn.net` (and all subdomains) were the most frequently requested. Figure 9 displays the top 35 second-level domains; the total list contains 57 different domains, many having only a single bitsquat request. All subdomains were combined into the second-level domain count. Interestingly, `microsoft.com` is not more popular, given the amount of Windows machines that automatically contact `microsoft.com` on a regular basis.

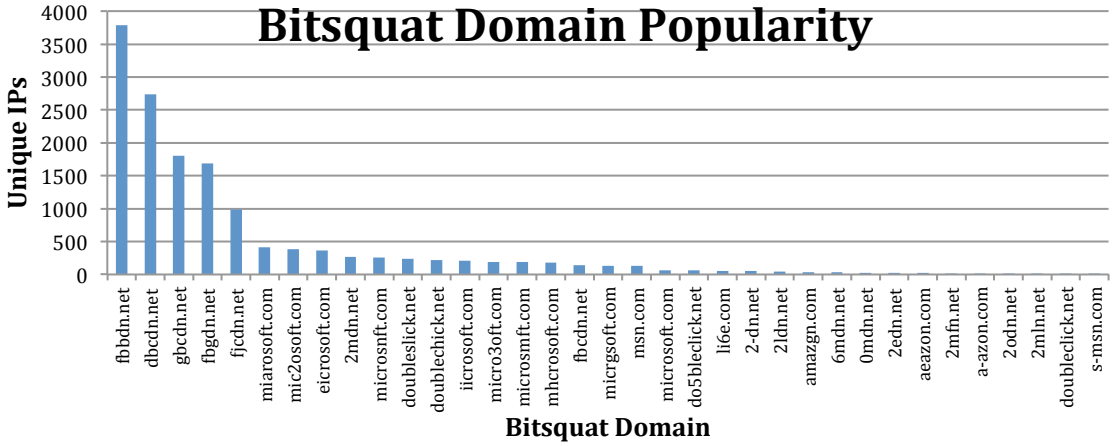

Figure 9: Domain popularity, ordered by the number of unique IPs with the domain in the HTTP `Host` header.

6 Mitigation

Bitsquatting attacks are possible, trivial to perform, and very easy to mitigate. The exploitability of bitsquatting is based on undetected data corruption and the availability of bitsquat domain names. Both issues have easy solutions, with removing availability of bitsquat domains being the easiest to implement.

6.1 Pre-registration

The easiest way to stop bitsquatting attacks is to register all available bitsquats of a domain. Bitsquatting is only viable against very popular domains. These domains are almost exclusively owned by entities with significant resources. Therefore, extra costs associated with registering all bitsquat domains are spread to those who can most easily afford them.

An upper bound on the actual cost is straightforward to calculate. The ASCII characters with the most bitsquats are "q" (bitsquats: p, s, u, y, a, 1) and "r" (bitsquats: s, p, v, z, b, 2), with 6 each. Assuming the worst-case scenario, a domain consisting of only the letters q and r, the cost per domain has an upper bound of:

$$Domain\ Length \times Price \times 6$$

With the market price of a `.com` registration being approximately $8 USD, the cost to pre-register all bitsquats of even a relatively lengthy domain, such as `googleusercontent.com`, would only be $816 USD/year.

Pre-registration immediately removes the exploitability of bitsquatting for all users, even those with damaged hardware and in harsh environments.

6.2 Cyclic Redundancy Checks

When storing critical data, be it on disk or in memory, include a CRC or other integrity check to ensure content has not unexpectedly changed. A necessary place for such checks is in distributed memory caches that power modern web platforms, such as memcached [**29**]. An extra CRC calculation increases integrity without a large performance drop. Had this been implemented, the bit-errors at the Zynga farm described in Section 5.1.1 might have never occurred.

Integrity checks are also critical in mobile devices that are subject to extreme environmental stress. The integrity of critical data, such as domain names about to be resolved, should be verified before transmission.

6.3 ECC Memory

An extremely simple mitigation for bitsquatting and general random data corruption is pervasive use of ECC memory. The technology is very mature and used for the main memory of virtually every enterprise class server. Unfortunately most consumer grade desktops, laptops, and mobile devices are still sold without ECC memory. Even hard disks for enterprise class servers have large non-ECC caches. In the age of pervasive Internet connectivity and electronic storage of vital data, the lack of pervasive ECC memory is unfortunate.

7 Conclusion

Computer hardware can and does experience errors. In memory technologies, error rates have been estimated to be anywhere from 150 to 81,000 FIT/Megabit of DRAM (anywhere from 3 errors per month to 3 errors per hour for a laptop with 4GiB of memory). Even the lowest estimates, magnified by the 5 billion devices connected to the Internet, yield 600,000 bit-errors per day in the world's computing devices.

Some of these bit-errors can be leveraged by malicious entities. In an attack called bitsquatting, malicious entities register domain names one bit different from a popular domain and wait for a random bit-error to occur. Eventually a bit-error occurs in a machine at just the right time and place to direct an unsuspecting victim to the malicious entity.

Bitsquatting was shown to be a real, experimentally verified, attack vector. In six months of measurement, over 16,000 unique computers were affected by bitsquatting several popular domains.

Fortunately, there are easy mitigations against bitsquatting attacks. Every entity that owns a frequently resolved domain should register all bitsquats to prevent malicious entities from doing so first. Another mitigation is the pervasive use of ECC RAM, a technology that has existed for decades.

It is important to remember the security implications of random failure, because on the Internet even a low probability can result in significant risk.

8 Appendix A: Selected Bitsquat Log Entries

These log entries are included to showcase the wide variety of devices affected by bitsquatting, potential attacks, and to give the reader a familiarity with what is meant by "computer generated" queries. Personally identifiable information, and the date and time of each request have been removed.

Computer Architecture Roundup

Many architecture and operating system combinations are vulnerable to bitsquatting. Some of the more exotic combinations are listed below. With the exception of MIPS, considerably more than a single request per architecture appears in the bitsquat logs.

MIPS

```
static.ak.gbcdn.net 69.80.8.xxx "GET /rsrc.php/zs/r/vJRBjt5XzbL.gif HTTP/1.1"
"http://www.facebook.com/home.php?" "Mozilla/5.0 (X11; U; Linux mips; en-US; rv:1.8.1.1)
Gecko/20070628 BonEcho/2.0.0.1"
```

PPC

```
ad-apac.do5bleclick.net 121.216.13.xxx "GET /adi-
onl.cl.ore.domain.realestate/home%3bcat=home%3bpos=2%3bsz=149x170?ord=814387366 HTTP/1.1"
"http://www.domain.com.au/Search/buy/Property/House/State/NSW/Area/Parramatta/Region/Sydney-
Region/Suburb/Merrylands/Merrylands-West/xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" "Mozilla/5.0
(Macintosh; U; PPC Mac OS X 10_5_4; en-us) AppleWebKit/525.18 (KHTML, like Gecko) Version/3.1.2
Safari/525.20.1"
```

PS3

```
btatic.ak.fbbdn.net 98.252.44.xxx "GET /rsrc.php/z9/r/LV5YAOOm14g.js HTTP/1.1"
"http://www.facebook.com/" "Mozilla/5.0 (PLAYSTATION 3; 1.00)"
```

PSP

```
profile.ak.dbcdn.net 173.3.183.xxx "GET /hprofile-ak-snc4/174300_1580807553_5121509_q.jpg
HTTP/1.1" "http://m.facebook.com/home.php?refid=11" "Mozilla/4.0 (PSP (PlayStation Portable);
2.00)"
```

Android

```
js.microsnft.com 62.201.142.xxx "GET /library/svy/office/production/broker.js HTTP/1.1"
"http://office.microsoft.com/fr-be/excel-help/afficher-ou-masquer-un-classeur-ou-une-feuille-
HP005199782.aspx" "Mozilla/5.0 (Linux; U; Android 2.2; fr-fr; Desire_A8181 Build/FRF91) AppleWe
bKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1"
```

Symbian

```
photos-c.ak.dbcdn.net 202.152.243.xxx "GET /hphotos-ak-snc4/hs160.a024/163618_547_91206110082
HTTP/1.1" "Nokia3230/2.0 (3.0505.2) SymbianOS/7.0s Series60/2.1 Profile/MIDP-2.0
Configuration/CLDC-1.0"
```

iPad

```
s0.2mln.net 96.242.164.xxx "GET /1490927/300x250_flu_girl.jpg HTTP/1.1" "Mozilla/5.0 (iPad; U; CPU
OS 4_2_1 like Mac OS X; en-us) AppleWebKit/533.17.9 (KHTML, like Gecko) Mobile/8C148"
```

iPhone

```
blstj.msn.com 166.137.139.xxx "GET /br/om/js/s_code.js HTTP/1.1"
"http://astrocenter.astrology.msn.com/msn/MyToday.aspx?When=1" "Mozilla/5.0 (iPhone; U; CPU iPhone
OS 3_1_3 like Mac OS X; en-us) AppleWebKit/528.18 (KHTML, like Gecko) Version/4.0 Mobile/7E18
Safari/528.16"
```

BlackBerry

```
platform.ak.dbcdn.net 68.171.231.xxx "GET
/www/app_full_proxy.php?app=114724435219699&v=1&size=z&cksum=30ee553bf1615278e82834fb7299e422&src=
http%3A%2F%2Fwww.xxxxxxxxx.com%2FFaceBook%2FPublica%2Fshowimage.php%3Fidfoto%3Dxxx HTTP/1.0"
"BlackBerry9700/5.0.0 Profile/MIDP-2.1 Configuration/CLDC-1.1 VendorID/215"
```

Solaris on x86

```
profile.ak.dbcdn.net 173.79.254.xxx "GET /hprofile-ak-
snc4/hs338.snc4/41760_100000660667407_1047_q.jpg HTTP/1.1" "http://www.facebook.com/" "Mozilla/5.0
(X11; U; SunOS i86pc; en-US; rv:1.9.2.3) Gecko/20100403 Firefox/3.6.3"
```

Wii

```
s0.2mdn.net 69.171.163.xxx "GET /3005201/0111103_01_MA_529_SetSights_15sec_300x250.swf HTTP/1.1"
404 220 "http://www.coolmath-games.com/" "Opera/9.30 (Nintendo Wii; U; ; 3642; en)"
```

Sensitive Activities

Some online activities are easier for miscreants to abuse than others. Bit-errors can happen when one is engaged in checking email or downloading Windows updates. Even without content insertion, an attacker could capture session cookies from a webmail or Facebook session. While the examples cited are not the only sensitive activities in the logs, they serve to showcase request diversity.

Checking webmail

```
sn110w.snt110.mail.li6e.com 187.92.218.xxx "GET /mail/clear.gif HTTP/1.1"
"http://sn110w.snt110.mail.live.com/mail/InboxLight.aspx?FolderID=00000000-0000-0000-0000-
000000000001&n=xxxxxxxxx" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0; GTB6.5;
.NET CLR 2.0.50727; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)"

secure.s%28ared.li6e.com 190.95.125.xxx "GET
/_/F$Live.SiteContent.Messenger/4.2.57151/Messenger.html HTTP/1.1" 404 215
"http://bl145w.blu145.mail.live.com/default.aspx?wa=wsignin1.0" "Mozilla/5.0 (Windows; U; Windows
NT 5.1; en-US) AppleWebKit/534.3 (KHTML, like Gecko) Chrome/6.0.472.63 Safari/534.3"

ad.doubleslick.net 118.102.133.xxx "GET
/ad/N6290.153730.YAHOO/B4557995.25;sz=1x1;ord=1288158383614325 HTTP/1.0" 404 229
"http://us.mc431.mail.yahoo.com/mc/showMessage?sMid=12&fid=%2540B%2540Bulk&sort=date&order=down&st
artMid=0&filterBy=&.rand=976320272&midIndex=12&mid=1_4963_63816_AEt5%2FNgAAHw5TMLUIwUV2GL9ww8&from
Id=xxxxxxxxxxxxxxxxxxxxx" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; GTB6.5)"

s0.0mdn.net 98.111.200.xxx "GET /viewad/2723889/pixel.jpg HTTP/1.1"
"http://us.mc316.mail.yahoo.com/mc/fc?cb=YAHOO.ads.darla._loaded&p=mail&f=xxxxxxxxx&l=MNW%2CN%2CRE
C%2CRS%2CRS2&en=CP1252&npv=1&indirect=MNW&rn=1285672337935&em=%7B%22site-
attribute%22%3A%20%22content%3Dno_expandable%3Bajax_cert_expandable%3B%22%7D&tgt=_blank&vw=showFol
der" "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.0; Trident/4.0; SuperSearchSearchToolbar
1.2; SLCC1; .NET CLR 2.0.50727; Media Center PC 5.0; .NET CLR 3.5.21022; .NET CLR 3.5.30729; .NET
CLR 3.0.30729; .NET4.0C; ShopperReports 3.0.489.0; SRS_IT_E8790576B1765C563EAB94)"

s0.2ldn.net 66.82.9.xxx "GET /879366/flashwrite_1_2.js HTTP/1.1"
"http://webmail.satx.rr.com/_uac/adpage.html" "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1;
WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media
Center PC 6.0; HPNTDF; AskTB5.2)"
```

Updating Windows

```
download.iicrosoft.com 91.198.175.xxx "GET /v9/windowsupdate/redir/muv4wuredir.cab?1010161718
HTTP/1.1" "Windows-Update-Agent"

www.update.microsnft.com 201.144.5.xxx "HEAD /v9/windowsupdate/selfupdate/wuident.cab?1011010912
HTTP/1.1" "Windows-Updatf-Agent"

beta.update.microsmft.com 113.117.249.xxx "GET /microsoftupdate/v6/default.aspx?1290162381
HTTP/1.1" "http://beta.update.microsoft.com/microsoftupdate/v6/default.aspx" "Mozilla/4.0
(compatible; MSIE 8.0; Windows NT 6.0; Trident/4.0; GTB6.6; SLCC1; .NET CLR 2.0.50727; .NET CLR
3.5.30729; .NET CLR 3.0.30618)"

download.microsoft.com 91.13.83.xxx "HEAD /WM/MicrosoftUpdate/redir/duredir.cab HTTP/1.1"
"Windows-Mobile-Device-Update-Agent"

msgr.dlservice.mic2osoft.com 213.178.224.xxx "GET /download/A/6/1/A616CCD4-B0CA-4A3D-B975-
3EDB38081B38/ar/wlsetup-cvr.exe HTTP/1.1" 404 268 "-" "Microsoft BITS/6.6"
```

Fetching Certificate Revocation Lists

```
www.microsmft.com 125.67.62.xxx "GET /pki/certs/MicrosoftWinIntPCA.crt HTTP/1.1" "Microsoft-
CryptoAPI/5.131.2600.2180"

www.eicrosoft.com 93.41.209.xxx "GET /pki/certs/MicrosoftRootCert.crt HTTP/1.1" "Microsoft-
CryptoAPI/5.131.2600.5512"

mscrl.eicrosoft.com 82.81.5.xxx "GET /pki/mscorp.crl/mswww(4).crl HTTP/1.1" "Microsoft-
CryptoAPI/6.1"
```

Downloading Executables

```
qh.dlservice.microsnft.com 180.117.126.xxx "GET /download.1/2/E/12E723C2-5946-40E1-BBB7-
1C4C0B955092/WindnwsXP-KB979687-x86-CHS.exe HTTP/1.1" "Mozilla/4.0(compatible;MSIE 5.00;Windows
98)"

qh.dlservice.microsnft.com 180.117.126.xxx "GET /download/5/A/1/5A132A68-342E-4E3C-AF8D-
881C9EB3E6D6/offibe2003-KB2344911,FullFile-CHS.exd HTTP/1.1" "Mozilla/4.0(compatible;MSIE
5.00;Windows 98)"
```

Problems with CNAME Chains

Each domain in a CNAME chain is a potential bitsquat target. Below are examples of what can happen when bit-errors interact with long CNAME chains.

```
ax.init.itunes.apple.com 81.225.40.xxx "GET /WebObjects/MZInit.woa/wa/initiateSession?ix=2
HTTP/1.1" 404 202 "-" "iTunes-iPhone/4.1 (4; 16GB)"
mmv.admob.com 109.175.185.xxx "GET /static/iphone/img/app@2x.png HTTP/1.1" "Mozilla/5.0 (iPhone;
U; CPU iPhone OS 4_1 like Mac OS X; HW iPhone2,1; en_gb) AppleWebKit/525.18.1 (KHTML, like Gecko)
(AdMob-iSDK-20101108; iphoneos4.2)"

lifestyle.msn.com 70.91.78.xxx "GET /article_toolbar_your_home.aspx HTTP/1.1" "Mozilla/4.0
(compatible; MSIE 7.0; Windows NT 5.1; GTB6.6; (R1 1.5); .NET CLR 1.0.3705; .NET CLR 2.0.50727;
.NET CLR 3.0.04506.30; .NET CLR 3.0.04506.648; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)"

emea.rel.msn.com 194.50.104.xxx "GET
/default.aspx?di=10230&pi=95517&ps=33229&pageid=8522097&mk=ru-ru&tp=http%3A%2F%2Fru.
msn.com%2F&fk=default&gp=S&optkey=default&parsergroup=hops HTTP/1.1" 404 184
"http://ru.msn.com/?ocid=iehp" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT
 5.1; .NET CLR 2.0.50727; .NET4.0C; .NET4.0E; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)"
```

Windows Crashes

A common argument against the plausibility of bitsquatting is that the machines will crash from random memory corruption before an attacker can accomplish anything. Some machines do crash, others connect to bitsquat domains, and some do both.

```
watson.microsnft.com 115.143.103.xxx "GET
/StageOne/acrord32_exe/9_3_3_177/msvcr80_dll/8_0_50727_3053/00008aa0.htm?OS=5.1.2600.2.00010100.3.
0&lcid=1042 HTTP/1.1" "MSDW"

watson.microsnft.com 187.73.247.xxx "GET
/StageOne/notepad_exe/5_1_2600_5512/uxtheme_dll/6_0_2900_5512/00004b43.htm?OS=5.1.2600.2.00010100.
3.0&lcid=1046 HTTP/1.1" "MSDW"

watson.mic2osoft.com 202.156.10.xxx "GET
/StageOne/Generic/FaultTolerantHeap/SearchIndexer_exe/7_0_7600_16385/4A5BD212/ffffbaad.htm?LCID=18
441&OS=6.1.7600.2.00010100.0.0.48.16385&SM=System%20manufacturer&SPN=System%20Product%20Name&BV=14
56 HTTP/1.1" "MSDW"

watson.micrgsoft.com 95.55.6.xxx "GET
/StageOne/Generic/WindowsWcpOtherFailure3/6_1_7600/base_wcp_sil_merged_ntu_ntsystem_cpp/Windows__R
tl__SystemImplementation__DirectRegistryProvider__SysOpenKey/3676/c0000034/0xf781db3c.htm?LCID=104
9&OS=6.1.7600.2.00010300.0.0.11.16385&SM=O.E.M&SPN=O.E.M&BV=080015&MID=xxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx&Queue=1 HTTP/1.1" "MSDW"

watson.microsnft.com 81.48.195.xxx "GET
/StageOne/avp_exe/9_0_0_754/4c62c63e/ntdll_dll/6_1_7600_16559/4ba9d132/c0000005/00055c55.htm?LCID=
1036&OS=6.1.7600.2.00010300.0.0.3.16385&SM=xxxxxxx%20xxxxxxxxxxx%20xx._%20LTD.&SPN=R780_R778&BV=05
JA.M020.20100128.LDG&MRK=144D_xxxxxxx_N_R780_04JA&MID=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx&Queue=1
HTTP/1.1" "MSDW"

watson.iicrosoft.com 97.79.151.xxx "GET
/StageOne/Generic/MSHTMLLAYOUTHARDASSERT/iexplore_exe/8_00_7600_16385%20(win7_rtm_090713-
1255)/mshtml_dll/8_00_7600_16385%20(win7_rtm_090713-
1255)/0x00000000002E4DA2.htm?LCID=1033&OS=6.1.7600.2.00010100.0.0.1.16385&SM=System%20manufact
urer&SPN=System%20Product%20Name&BV=2005&MID=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx HTTP/1.1"MSDW"
```

9 Bibliography

[1] Peter Baston. (2002, January) Unsafe At Any Speed? - Looking under the hood at Sun's 2001 SF engine problems. [Online]. http://www.sparcproductdirectory.com/artic-2002-jan-pb.html

[2] Wikipedia. (2011, July) Electrical resistance and conductance. [Online]. http://en.wikipedia.org/wiki/Electrical_resistance_and_conductance#Temperature_depender

[3] Sudhakar Govindavajhala and Andrew W. Appel, "Using Memory Errors to Attack a Virtual Mac *Symposium on Security and Privacy*, Berkeley, CA, USA, 2003, p. 12.

[4] Costis Sideris. (2009, September) FPGABoy. [Online]. http://www.fpgb.org/?page_id=17

[5] Ross J. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, 2nc 2008.

[6] Apple, Inc. (2011, July) iPhone 4 Technical Specifications. [Online]. http://www.apple.com/iphone/specs.html

[7] RIM. (2006) [Online]. http://www.rcom.co.in/Rcom/personal/blackberry/pdf/BB_Pearl8130.

[8] RIM, Inc. (2005, November) BlackBerry 8700g Wireless Handheld: Safety and Product Informa http://www.entelpcs.cl/modequipos/manuales/13131.1.manual.pdf

[9] RIM, Inc. (2010) BlackBerry Bold 9700 Smartphone: Safety and Product Information. [Online]. http://docs.blackberry.com/en/smartphone_users/deliverables/11261/BlackBerry_Bold_970 US.pdf

[10] HTC, Inc. (2009) HTC Imagio User Manual. [Online]. http://cache.vzw.com/multimedia/mim/htc_imagio/htc_imagio.pdf

[11] Nokia, Inc. (2006) Nokia E62 User Guide. [Online]. http://static.highspeedbackbone.net/pdf/Nokia_E62_Manual.pdf

[12] Samsung Electronics Co Ltd. GSM Mobile Cellular Phone SGH-600 Service Manual. [Online]. http://deblocage77.free.fr/sam/root/Samsung%20SGH-600%20service%20manual.pdf

[13] Siemens AG. (2004) Siemens Mobile A65. [Online]. http://www.mercator.iac.es/observing/A3 A40-1-4A19_ba_body_NET.pdf

[14] Eugene Normand, "Single event upset at ground level," *IEEE Transactions on Nuclear Science*, v 2742 - 2750, December 1996.

[15] Arthur H. Compton, "Variation of the Cosmic Rays with Latitude," *Physical Review Letters*, vol. 4 111-113, July 1932.

[16] Stanford Linear Accelerator Project. (2011, February) Introduction to Cosmic Rays. [Online]. http://www2.slac.stanford.edu/vvc/cosmic_rays.html

[17] H. W. Curtis, H. P. Muhlfeld, C. J. Montrose, B. Chin, M. Nicewicz, C. A. Russell, W. Y. Wang, L. B. F Hosier, L. E. LaFave, J. L. Walsh, J. M. Orro, G. J. Unger, J. M. Ross, T. J. O'Gorman, B. Messina, T. D Sykes, H. Yourke, T. A. Enger, V. Tolat, T. S. Scott, A. H. Taber, R. J. Sussman, W. A. Klein, C. W. W Ziegler, "IBM experiments in soft fails in computer electronics (1978-1994)," *IBM Journal of Re Development*, vol. 40, no. 1, pp. 3-18, 1996.

[18] T. C. May and M. H. Woods, "Alpha-particle-induced soft errors in dynamic memories," *IEEE Tr Electron Devices*, vol. 26, no. 1, pp. 2-9, January 1979.

[19] Terrazon Semiconductor. (2004, January) Soft Errors in Electronic Memory – A White Paper. [(http://www.tezzaron.com/about/papers/soft_errors_1_1_secure.pdf

[20] Bianca Schroeder, Eduardo Pinheiro, and Wolf-Dietrich Weber, "DRAM Errors in the Wild: A La
Study," in *ACM SIGMETRICS/Performance*, Seattle, WA, USA, 2009.

[21] Michael Barr. (2011) CRC Mathematics and Theory. [Online]. http://www.netrino.com/Embed
Systems/How-To/CRC-Math-Theory

[22] Chipworks. (2010, June) Teardown of the Apple iPhone 4 Smart Phone. [Online].
http://www.chipworks.com/en/technical-competitive-analysis/resources/recent-
teardowns/2010/06/silicon-teardown-of-the-apple-iphone-4-smart-phone/

[23] IMS Research. (2010, August) Internet Connected Devices About to Pass the 5 Billion Milestone
http://www.businesswire.com/news/home/20100816005081/en/Internet-Connected-Devic
Billion-Milestone

[24] Cisco Systems. (2011, June) Cisco Visual Networking Index: Forecast and Methodology, 2010-2
http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white
481360_ns827_Networking_Solutions_White_Paper.html

[25] Wikipedia. (2011, July) Typosquatting. [Online]. http://en.wikipedia.org/wiki/Typosquatting

[26] Wikimedia Foundation. Wikimedia Traffic Analysis Report - Operating Systems. [Online].
http://stats.wikimedia.org/archive/squid_reports/2011-03/SquidReportOperatingSystems.ht

[27] MaxMind, Inc. (2011, May) MaxMind - GeoLite City | Free Geolocation Database. [Online].
http://www.maxmind.com/app/geolitecity

[28] Hypertext Transfer Protocol -- HTTP/1.1. [Online]. http://www.w3.org/Protocols/rfc2616/rfc
sec14.html#sec14.23

[29] memcached - a distributed memory object caching system. [Online]. http://memcached.org/