Journal of Information Technology Education: Discussion Cases

04/27/2017

TAUFEEQ MOHAMMED, UTKARSH SHRIVASTAVA, ASHISH K. DAS, QUYNH T. NGUYEN

GRANDON.COM GOT HACKED![1]

You can do reverse engineering, but you can't do reverse hacking.
—Anonymous

Dr. T. Grandon Gill, a Professor in the Information Systems and Decision Sciences Department at the University of South Florida, was traveling with his family in England when he received a strange phone message. Not being able to respond, he ignored it until—a couple of days later—he was notified that access to his personal website had been suspended (see Exhibit 1). *Grandon.com* had, once again, been hacked--for the 7th time.

Getting his website hacked was not a new experience for Grandon Gill. In the past, however, getting the site back up and running had been a quick fix involving replacing the corrupted files. This time it was different. Based on the email and his service provider's response, his site now contained links to PayPal phishing sites. Without significant changes, he could become complicit in fraud if the situation was not remedied. This was a problem that could no longer be ignored.

After Gill had re-read the email, he pondered the various options available to him. Given the amount of trouble it was causing him, he wondered if he needed the website at all. To maintain the domain name grandon.com, which he had held for more than 20 years, all he needed to do was to put up a simple landing page with a message: "Hi, I am Grandon—go to my school account to find out more." At the other extreme, he could completely re-engineer the site to make it much less vulnerable—a process that could take days, if not weeks. Between the two extremes, there were many other possibilities. These included changing hosts, simplifying the site so that it contained only the most critical information, dropping its WordPress component, or even going to a pure WordPress model. He had a suspicion, based on previous experience, that vulnerabilities in WordPress may have been the source of the hack. But were these vulnerabilities intrinsic to the application, or were they simply the result of his inattentive management? Whatever he decided, he needed to take action soon. It was very embarrassing, and perhaps professionally damaging, to have his site showing an unavailable message. He thought back to a popular ironic quote that said: "Good decisions come from experience, and experience comes from bad decisions." *What should he do now?*

Grandon.com

Grandon.com was the personal website of Dr. T. Grandon Gill, a Professor in the Information Systems and Decision Sciences Department at the University of South Florida. He acquired the *grandon.com* domain in 1996—quite early in the evolution of the web—and had kept the site operational since that time.

Ironically, Grandon was not Gill's first name and—before he turned 25—he had always been known as Tom. What people called him changed by accident when he enrolled in the MBA program at Harvard Business School (HBS). At the time he was accepted by the program, Gill was working as a nuclear-trained submarine officer. While he was filling out his enrollment card for the school, his submarine was rolling on the surface, and Gill was quite seasick. As a consequence, he failed to notice that the box he thought was for his middle name was actually labeled "nickname". After mailing in the card, he completely forgot about it. Then, upon arriving at HBS six months later, he discovered that "Grandon Gill" was printed on all his name cards. To make matters worse, the professors had memorized that name. After five years in the Navy, he knew it would be easier to change what he was called than to get everybody else to change.

The Past

In many ways, having a unique name was a boon in the internet age. In 1995, when the World Wide Web was relatively new, Gill acquired the domain name "*Grandon.com*". Initially, he used the site to create static content of personal stuff like pictures of him and his family. He posted baby pictures when each of his two sons was born. (It was a bygone era, and nobody worried about privacy). A few years later, when he decided to join the faculty at the University of South Florida in Tampa, he used the website to post detailed information relating to the listing of his Boca Raton home for sale. That content stayed up for years after he moved to USF in 2001.

The *Grandon.com* website remained mostly a curiosity until 2011. In that year, he got a grant from the National Science Foundation (NSF) to develop a series of case studies for his department's undergraduate capstone course. One of the cases he developed, about a construction engineer trying to decide if she needed a website for her business, made him realize how woefully out of date his knowledge of web technology had become. As a result, he decided to create a new version of his personal website that employed a wide variety of different technologies, so that he could better understand them.

In designing the website, Gill didn't really care about the aesthetics. Instead, he saw the site as a vehicle for displaying lots of information and doing experimentation any time he felt like it. He wrote the entire code of the website on his own and found that to be a good learning experience. In appearance, his website looked like a group of boxes, with each box containing some information. It was a set of PHP scripts with RSS feeds linked to them (see Exhibit 2). In addition to feeds from a WordPress site that he created, he linked to a variety of external feeds (e.g., from the *Wall Street Journal*, the *Dilbert* comic strip, his university, and so forth). He also experimented with a number of different open source web applications. In addition to WordPress, these included Lime Survey (a survey tool) and Moodle (a learning management system). He didn't really worry about hacking. Who would care enough to hack him? Indeed, after his website first got hacked—possibly by Chinese hackers—he jokingly said that now there are at least two people who read stuff on his website!

As time progressed, more and more content was added to Gill's website. He was a strong advocate of case studies and had written dozens of them, so he began to post them on the site. He similarly posted the articles and books he had written. He was not the least bit concerned about generating traffic. In fact, the

site mainly served as a convenient portal through which he could access his materials. Indeed, the catch phrase on the site's title bar was:

> *Looks ugly, loads slowly, content overload… What's not to like?*

In 2014, *Grandon.com* took on a more serious role. At that time, USF was preparing to launch the Doctor of Business Administration (DBA) program. Gill was the incoming Academic Director for that program, which was awaiting approval by USF's Board of Trustees and the state's Board of Governors. Until that approval was received, however, the program could not advertise its existence on USF's own websites.

Complicating the inability to advertise was the fact that the program was quite unique. Specifically, it was a 3-year doctoral degree intended to meet the needs of working executives with a minimum of 12 years of professional experience—at least 5 of which were in a senior role. Classes would meet only on one weekend each month. It was also expensive, with a cost of $30,000 per year.

It was hard to attract students to a program that was being kept secret. Nevertheless, inquiries about the program had started pouring in. To address this, Gill created "unofficial" content describing the program. This content included videos, white papers, links to similar programs, and announcements. These were all hosted on *Grandon.com*. Whenever questions came in, potential applicants were directed to that website. Despite its amateurish appearance, *Grandon.com* served its purpose. Well before the program became "official," a large number of potential applicants had been identified. The DBA programs launch proved to be a huge success, and Gill's website had played a significant role in its formative year. Less than a year later, the program was approved, and all the relevant information about the DBA program had been put on the official website of USF. *Grandon.com* didn't seem as critical as it had been, and Gill turned his attention elsewhere.

Shortly after Gill ceased paying attention to the site, it was hacked for the first time. One day, when he opened his browser, the main page of *Grandon.com* was replaced by an animated (and noisy) page that claimed to be the work of the "Malaysian Hacker's Army". After a few hours, Gill found all the affected pages, deleted and replaced them with his originals, and changed all his relevant passwords. The site continued unaltered for several months, and Gill thought the problem had been solved.

Unfortunately, his optimism proved to be short-lived. Over the following year, his site was corrupted four more times. The corruption was so simple to fix—and the site served such a minor purpose—that he simply repaired it each time. He knew that the hackers got in somehow, but he didn't know how.

Cut to the present day--Gill's website was hacked for the 7th time, and the defacement was quite different from the previous ones. Hackers linked to a PayPal phishing account on every WordPress post summarized on the website. That caused the ISP to block the website and bring it down temporarily.

Trends in Web Development

In September 2014, the World Wide Web reached the milestone of 1 billion websites. It was a huge number, given the fact that commercial web browsers had not existed prior to 1991. The web had undergone many changes and had come a long way since then (see Exhibit 3). Different types of websites were created for various needs. As the web evolved, dedicated websites for e-commerce, photo sharing, social interactions, job searches, entertainment, news, and discussions became standard, with new applications continually emerging.

The impetus to online presence and web use has led to business models that revolved around the services provided through the internet. Amazon was a prime example of a company providing many such services.

Beyond businesses, however, many individuals and professionals sought to harness the power of the internet by creating their own personal websites. *Grandon.com* was such a personal website for sharing Gill's research and professional information.

Web Development Technologies

In the early days of the web, pages were primarily built by writing instructions in the HyperText Markup Language (HTML). Web browsers (such as Mozilla Firefox, Google Chrome, and Internet Explorer) decoded the script to present information as a web page displayed to the user. A valid HTML script was simply a structured organization of elements such as tags, attributes, and content, and could be written using a simple text editor, such as the Notepad utility in Windows.

As time passed, web developers sought to improve the way the content was displayed on web pages and wanted to make the websites look more aesthetic. But the standardization and rigid nature of HTML limited their creative freedom. That led to the development of stylesheet languages such as CSS (Cascading Style Sheet) that dramatically improved the presentation and consistency of the structured documents written in markup languages like HTML. CSS allowed web developers to separate the structure of a document's content from its presentation.

In addition to issues of styling, the web developers desired to have capabilities that could be used to make the websites interactive. The most popular solution was JavaScript. JavaScript enabled developers to make the websites more interesting to the users, and this changed the way they viewed websites. A lot of interesting features started appearing on websites, and the user experience improved dramatically. The combination of HTML, CSS, and Javascript enabled the display of animations, adaptive interfaces, and came to be known as Dynamic HTML (Crowder, 2010).

> *HTML tells a story to the browser. CSS clothes and animates the characters. JavaScript whispers from the wings, changing the plot. —Thenewcode.com*

Grandon.com also utilized JavaScript to display dynamic RSS (Rich Site Summary or Really Simple Syndication, depending upon who you asked) feeds from social media and news websites, while HTML and CSS were integral to the web page themes.

Website Personalization

A top level executive from Google once reportedly said, "People get frustrated when they don't see the search results that they expect."

Web personalization took the user experience to the next level. While JavaScript enabled web developers to make web pages interactive, web personalization customized the website for each user. Companies like Amazon, Google, and Yahoo! personalized their web pages to each user. The result was both improved user experience and revenues.

Early dynamic and personalized webpages were built by writing programs in C and Perl programming languages and executing them on the web server through Common Gateway Interface (CGI). This approach had two disadvantages. First, the web developers had to learn new programming languages. Second, the compilation of the executable files made running the scripts time-consuming and inefficient. To solve this problem, open source server-side scripting languages, such as PHP, were developed. These made dynamic web pages much easier to develop. *Grandon.com* for instance, was almost entirely based on a set of PHP scripts.

Typically, scripts were embedded within the HTML code of a page, but were executed on the web server to modify the basic HTML output whenever a specified trigger was encountered. For instance, a holiday could trigger a personalized website to change the page theme and display special messages to the visitors. PHP frequently made calls to database applications, such as MySQL, to make the requisite changes. This allowed it to store and retrieve data for later access (Davis & Phillips, 2007). Other platforms, such as Microsoft's Active Server Pages (ASP) and Sun's Java Server Pages (JSP) also performed similar tasks.

Hacking: Who, Why and How?

One consequence of the explosion of flexible web technologies and platforms on the internet was an influx of unskilled and inexperienced webmasters and service providers. Security loopholes in a website allowed notorious web enthusiasts and cyber criminals to exploit website vulnerabilities for personal or organizational gains. A cyber security report found that Google blacklisted about 20,000 websites each week for malware and around 50,000 websites a week for phishing (Sucuri Security, 2016).

Given the low profile of his website, Gill had originally anticipated that *Grandon.com* would not be particularly targeted by malicious attacks. Hacking activity, however, was often conducted by automated systems referred to as bots. Unfortunately, such bots did not necessarily take the traffic and popularity of a site into account when conducting an attack, for example one defacer alone managed to deface 13,405 different websites on one day in December 2012 (Das, Nguyen, & Thomas, 2017). Using such automated bots to crawl the World Wide Web for vulnerabilities was a common approach employed by hackers; multiple automated attacks increased the odds of success through economies of scale. In terms of website defacement, hackers used mass defacement techniques to vandalize a number of websites at once. The case of Hmei7 attackers who successfully defaced as many as a thousand websites per day was an example to illustrate the power of botnet in terms of large scale attacks. Therefore, even vulnerable websites of modest popularity had a high probability of getting hacked. This phenomenon was less widely known than the targeted attacks on popular websites, such as the hacking of Sony Corporation's website in 2014 by North Korean hackers seeking to avenge Sony's unwillingness to halt the production of *The Interview*, a movie that featured North Korea in an unflattering way.

The costs of hacking websites have declined over time while the benefits have increased. Some of the benefits the hackers derived from hacking a website included:

- *Computing Resources*: A website could be the target of a hacker for its server's computing resources. The hacked servers could be employed for cluster computing, or to perform any illegal activity through the web server without disclosing identity.

- *Monetary Benefits*: The hacker could use the email functionality of the webserver to send SPAM emails to prospective customers. These emails might also include links to certain commercial websites that were invisible to the visitor--improving the page ranking of the advertised site. Redirecting the visitors to an affiliated website could also be a source of revenue for the hackers.

- *Entertainment Purposes*: The study conducted by Das and his colleagues (2017) found out that the majority of hackers attack websites for their own entertainment activities by using file inclusion techniques. Nagpal, Chauhan and Singh (2015) pointed out that attackers aimed to modify the code and get basic technical information about the web application like the database being used at the back end.

- *Self-Identity:* Hackers could practice developing and launching web application bugs/worms that were written in a high-level language and could run on different operating systems or even

architectures (Levy & Arce, 2006). The most popular and harmful web application bugs were built using XSS (cross-site scripting) (Watson, 2007; Wang, Li, & Guo, 2011; Ami & Malav, 2013). The malware was propagated by using a symbiotic relationship between the client and the server when delivering a web page to a client that contained some malicious code injected by hackers with the use of XSS (Levy & Arce, 2006). With an increase of web exploit toolkits, which were "packaged" as attack frameworks traded online, attackers were even more encouraged to intrude on pages owing to their high success rate (Caldwell, 2011). This kind of incident could simply aim to cause some damages to the target pages as many attackers found that causing some damage gave them more satisfaction than just breaking into the system (Floyd, Harrington, & Hivale, 2007). These hackers were highly motivated by challenges—with successful sabotage being a key black-hat achievement (Xu, Hu, & Zhang, 2013).

- *Injecting Malicious Software*: A hacker could use a vulnerable website to inject viruses into the computers of its visitors. The malicious script injected by the hacker on the website would execute in the visitor's browser to extract personal information or install spyware. By secretly installing backdoors and key trackers, a visitor's private information could either be stolen or blocked (ransomware) for ransom. Recent statistics showed that backdoors were mostly responsible for a website hack (see Exhibit 4). Gill strongly suspected that such a backdoor could have led to the problems he had recently encountered.

Symptoms of a Hacked Website

Hackers used a variety of methods to gain access to the web infrastructure and sometimes it was not easy for a webmaster to identify if the website had been hacked or not. Modern hackers frequently tried to avoid detection, so that they could keep collecting information, installing malware, and spreading viruses to other machines on the internet. Among the signs and symptoms of a hacked website were the following:

- The presence of advertisements related to pornography, drugs, or illegal services on the website's header or footer. In one of the later hacks of *Grandon.com*, a hacker had put ads related to male enhancement products on every single page of the blog on the website. It took two hours for Gill to clean up everything.

- A communication from website visitors may also serve as an alert regarding vulnerability that was not visible to the web administrator. Hacks that redirected the visitors to a malicious site might not be visible to the webmaster. Generally, sophisticated web browsers like Google Chrome would warn about dangers of visiting a vulnerable site (see Exhibit 5).

- If the website was used for sending spam emails, then the ISP or web host might alert the site's owner by sending reports about the malicious activities on the website. Spammers usually sent the URL of the infected website in the mass emails to avoid getting filtered by spam filters. Upon clicking the URL, the visitors were redirected to the site of the spammers' interest.

- Within the website architecture, the presence of unknown files or admin users in the database, and modification or duplication of existing PHP files such as index.php or wp-config.php could indicate malicious activity on a website. A majority of the early hacks of *Grandon.com* involved an overwriting of the site's main index.php file. Because this file was the first file loaded by the server when a user accessed the site, once it was changed the entire site came under the control of the hacked code.

Hacking Attacks on Websites

Attempts to hack a website frequently started by gathering information about the installation. The hackers would look for the maintenance or update history of the website to get an idea about the alertness level of the webmaster. A website that was not frequently updated or managed was a soft target and occupied the high position on the hit list. A hacking attack could exploit user, application, or system vulnerabilities. Some of the most important of these are summarized in Exhibit 6. The nature of the website architecture, shady programming, and laidback attitude of webmasters made some websites particularly prone to some specific cyber-attacks. These included:

Reflected Cross Site Scripting (RXSS): Users of a site entered text that included scripting code into an input control (such as a textbox) that assumed that only text had been entered. When the text was rendered on a webpage, the code executed—potentially with malicious consequences. For example, a hacker might message another user the link of a compromised webpage with an embedded script. If the user opened the page on a browser with JavaScript enabled, the script would execute commands that could give the hacker complete or partial control of the unsuspecting user's system.

Persistent Cross Site Scripting (PXSS): This was similar to reflected cross-site scripting, except that the malicious script was entered into a control whose content was saved to the database, such as the database used to store the user content or profile page on a blogging website. The consequences were similar to those of RXSS. Exhibit 7 explains the mechanism of the Cross Site Scripting (XSS) attacks.

SQL Injection: User input was often used as the basis of a search command that executed on the server database. If not properly validated, it was possible to append additional SQL commands to the input. Most commonly this allowed the hacker to access information from the database for which he or she was not authorized.

File Inclusion: Attackers often included different source files, such as database content, images, PHP classes, and more for operating their web application. The ability to execute an arbitrary file as code was a severe security risk (Ami & Malav, 2013).

Buffer Overflow: When the text was received from user input, it was generally placed into a holding area in the server's known buffer. If more text was provided than the buffer could hold, it could conceivably run over into areas that contained other data (such as scripting code). Through that process, the intended executing of the script could be disrupted or, in the worst case, alternative code inserted by the hacker could be executed.

Denial of Service: When a hacker's code was injected through other means, it could cause server performance to degrade or to fail entirely. This could be accomplished through overloading the server CPU, exceeding the server's RAM, or through writing files to the point that they reach the server's disk capacity limits.

Brute-force Attacks: In this type of hacking attack the hackers would try different combinations of usernames and passwords to gain control over of a web application. SSH, MySQL database, and Webadmin services were the key targets of this kind of attack.

Backdoor: The most prevalent vulnerability of websites in general (see Exhibit 4), a backdoor referred to a method of bypassing normal authentication and gaining the ability to remotely access the server while remaining undetected. These also allowed hackers to regain access to the website even after upgrades or the removal of the exploited plugins. Some backdoors allowed users to create a hidden admin username in the database, whereas some more sophisticated backdoors might allow the hacker to execute any PHP

code sent to the browser. Others provided a full-fledged user interfaces, allowing them to send emails to the targeted server, execute SQL queries, and complete nearly any task that the server's true administrator could do.

Phishing Attack: In these types of attacks the hackers would infect the checkout pages of legitimate e-commerce websites and redirect the customers to a hacked website. In the latest hack, Gill was notified of a PayPal phishing attack on his website by a concerned customer. Possibly the hackers hid a malicious PHP script within a subfolder of Gill's site to impersonate *Paypal.com* web pages and steal the customer's credentials (see Exhibit 8). These attacks were hard to detect by online security scanners as hackers would hide payload in a subfolder not linked to the main page (Sinegubko, 2016).

WordPress and Content Management Systems

As the web evolved, the need to develop platforms that supported the easy creation of websites with dynamic content became clear. A particularly popular category of applications was referred to as content management systems (CMS). These systems allowed users to create a website with consistent formatting and easy ability to add new information. They also allowed the site to establish a consistent look and feel. A prime example of a CMS was the open-source application WordPress.

WordPress was originally developed as a platform to support quick and automatic update blogging sites from remote locations. It soon became recognized as an easy tool for novices who just wanted to upload content to a website in spite of not knowing much about web design. Blogger, a service provided by Google, and WordPress rapidly became the most popular web blog service providers.

Unlike Blogger, WordPress was an open source program that resided on the same server that hosted a website. Most of the popular web service providers had the necessary software (e.g., PHP and MySQL) required to run WordPress, and some of them even provided it as a part of the basic service. Installing WordPress required the webmaster to download the installation file, extract its contents to the web server, and link a MySQL database to the application. Over time, WordPress grew to become the most popular tool for personal web development.

WordPress Website Architecture

WordPress was an open source project using PHP and linked to a database. The data used to display the pages and posts of a WordPress site was stored in a local database. A typical WordPress site was therefore connected to a server such as MySQL. The PHP files on the server could access and modify the content in the database (Gill, 2016).

When a web browser on the client's machine sent a request to the webserver hosting the website, the base PHP file was the first loaded and read by the server. The scripts in the base PHP files called the database and other PHP files to acquire additional content to display on the web page. The outcome of the successful execution of the script was an HTML webpage which was then sent to the web browser on the client's machine.

WordPress Features

WordPress' popularity could be attributed to its flexibility, customizability, and ease of use. A WordPress website began with a base level installation that established the database and all necessary scripts to launch a fully functional website (WordPress, 2017). After that, the user could customize the website. Some of the popular tools for customization were:

Themes: WordPress websites could be customized with a variety of themes. Each theme was a collection of PHP files and style sheets that gave each theme a distinct look and feel. Once a theme was selected, the user further had an option to customize by changing fonts, backgrounds, and active components.

Plugins: To add additional functionality to the website, for example hosting a shopping cart or integrating the social media accounts, WordPress plug-ins could be used to add a specific functionality to the website.

Widgets: Along with Themes and Plugins, widgets helped in customizing the website further by enabling the user to add content and features in the widgetized area of the websites, usually sidebar, header, footer, etc. For example, the RSS widget could be used to integrate an external feed source from content into a widget area of the website, say a Twitter account or Facebook posts.

Because WordPress was an open source platform, a developer could also modify existing PHP files within the site and develop a custom website. The customization came at a cost, however. The features of the WordPress site such as Themes and Plugins needed to be regularly updated for efficiency and security of the website. Gill suspected that one of the reasons that his website got hacked was because he only updated the WordPress plug-ins that he used for *Grandon.com* at irregular intervals. Was he sure? Not really.

WordPress Site Deployment Options

Hosting a WordPress website and making it accessible to the world using the internet required a web server with internet connectivity along with other resource management applications. Hosting a website was like owning a house. The web design dealt with the look and feel of the house, the web hosting served as the infrastructure and plumbing, and the domain name was akin to the address. These three elements of a website played a major role in determining how the website was accessed and the amount of customizability features available to the user. The deployment of a WordPress site was most commonly accomplished through the following alternatives:

- *Free hosting on wordpress.com:* The websites deployed using the free plan of WordPress had limited customizability. The *wordpress.com* subdomain was attached to the URL—meaning that a visitor would know that the site was hosted for free—and *WordPress.com* reserved the right to display third party ads on the website.

- *Paid hosting on WordPress:* There were a variety of paid hosting plans available for web developers. The benefits included greater storage space, no advertisements on the webpage, more themes, and the ability to monetize the website by posting custom ads (see Exhibit 9).

- *Dedicated or shared hosting on a third-party server:* The WordPress application could be installed on a third-party server with all forms of customization, and support for themes and plugins. Gill's website utilized a paid subscription on a third party hosting provider, *Gate.com*, that provided general purpose web hosting.

- *Hosting on an owned server.* An individual or organization that owned a web server could download and install WordPress and an open source database, and run a WordPress site at no cost. While offering the maximum degree of flexibility, it also placed the responsibility for server management entirely on the owner.

WordPress Website Vulnerabilities

At the time of the case, WordPress enjoyed around 60% market share in the CMS category, and 38% of top 10,000 websites on the internet were powered by the WordPress. WordPress was used by many Fortune 500 companies (e.g., UPS, Xerox, and Microsoft) and celebrities (e.g., Usain Bolt, Katy Perry, Beyonce, etc.). But unfortunately, that popularity came with a cost. The very number of high traffic WordPress sites made them an attractive target for hackers. Its open source origins also exposed possible vulnerabilities by making its source code public. For example, a survey of 9,000 infected websites in 2016 by the independent firm Sucuri found that 78% of them were supported by WordPress. Attackers kept finding new ways to infect the WordPress websites for their personal gains. In one form of attack, hackers infected vulnerable WordPress sites with the scripts that led to the installation of malware on the machine of an insecure visitor. Also, known as ransomware, this malicious software would block the access to the data until a ransom was paid to the attacker. An instance of a message received by an infected user is displayed in Exhibit 10. In 2017, attackers utilized the vulnerabilities in two popular WordPress plugins (Insert PHP and Exec-PHP) to secretly gain control over WordPress sites (Cid, 2017).

In order to execute a Backdoor attack on a WordPress site the hackers would generally save malicious scripts inside the directories of native themes or plugins. Many site managers avoided upgrading their plugins, since doing so created a risk of incompatibilities with the site's existing code. As a result, these scripts often survived for a long period of time. On a server without any monitor tool, suspicious files could also be uploaded into the uploads directory of the web server hosting WordPress. Similarly, the WordPress configuration file "Wp-config.php" was also particularly attractive for backdoor entries, since it identified key users and database connections (Wpbeginner, 2012). Interestingly, both updated and older versions of the WordPress site were vulnerable to phishing attacks (Cid, 2014). The hackers would gain control of the WordPress site through a vulnerable plugin using remote command execution or SQL injection (see Exhibit 6 for more details on vulnerability of websites).

Grandon.com

The Present and Future

Gill thought about the various problems that might have caused the issue. He wasn't sure of any of them though, and wondered if there was an easy way to find out the cause. He did some retrospection and tried to single out an option from the various possible options though. One of the reasons, he wondered, could have been because of the experimentation on his website. His experimentation on the website could have opened some backdoors. These, in turn, could have been exploited by the potential hackers or bots. Any of the applications he had installed--including Limesurvey and Moodle—could have been the culprits. More likely, since neither WordPress nor its plug-ins had been updated regularly, the hackers could have found a way to exploit any one of dozens of possible entries to the website. He was all too aware that once a backdoor had been established, even if he restored the website, some malicious hidden code could always remain. Like cancer cells, if you did not eradicate all of them, the threat remained—giving the original hackers a free pass to come back and hack anytime.

As Gill thought about the various possible root-causes, he thought back on the good old days when hacking wasn't a big issue. Of course, there were people who would try to hack for personal malicious satisfaction, but their number was few. With the advancement of technology, now there were bots that crawled through the web and found websites that could be hacked easily. One of the parameters that the bots checked was to see when the website was last updated. Sadly, even if he fixed the problem completely, Gill was not sure that he would check for updates as often as needed to keep it secure.

As he considered what to do, Gill gave a thought about what exactly he wanted from the website. Indeed, he wondered whether he needed a website in the first place. As he thought about this, he considered various factors about his age, schedule, and future aspirations. He was 62 years old, healthy, and felt that he had another eight to ten years left in his professional career. Though he would have welcomed some free time, he was currently in one of the busiest phases of his life, second only to his tenure as a Navy submarine officer (where he routinely had18-hour work days).

Between his responsibilities as academic director of the DBA program, pursuit of a relatively active research agenda, teaching 5-6 courses a year, and serving as principle investigator for a couple of research grants, he was now putting in 50-60 work hour weeks. He worried about finding enough time to acquire new skills and keep up with the latest technological trends. Would it make sense to view redesigning his website as a skill acquisition activity? Certainly, if time weren't a constraint, Gill would have liked to experiment with the website, learn more about the technology of website design during the process and, of course, establish a routine to keep the site updated.

One worry that Gill did not have was the theft of his intellectual property—a common concern of individuals with documents and applications hosted on the cloud. He wanted the documents and information on his website to be shared freely on the internet. To succeed as a scholar, you needed your information shared as widely as possible. The research papers that he submitted and the educational videos he created were all covered under a Creative Commons license intended to encourage free distribution. Of course, *Grandon.com* was not the only source of these documents. Reiterating the advantage of having a unique name, Gill's papers, videos, and references were easily found by name—particularly on the Google Scholar website, which was rapidly becoming the "go to" site for academics.

But he also felt that he would miss the flexibility a website offered. He could create and post the content whenever he wanted to, without having to worry about permission from anybody. If he wanted to put something on the university website, it involved a series of steps: contacting the webmaster, getting approvals, etc. But again, as a wise man once said: "You can't have your cake and eat it too." He could also do his posting on social media. He had acquired over a thousand contacts on LinkedIn—but he actually knew only a small percentage of them. And using social media, you were highly constrained with respect to how your content was formatted and what content could conveniently be included.

The more Gill thought about what he wanted, the more he got confused about what to do. There were many trade-offs, and he had to let go of something to have something else. And he was now back to thinking about the problem: What to do?

The Decision: What to Do?

After returning from the vacation, while his website was shut down, Gill pondered the future of his website. Normally, he made decisions very quickly. This decision was different. There were so many options that he could choose from. It would be embarrassing if he, with all his expertise, ended up doing something that would (in retrospect) seem ill-conceived.

He thought about a broad range of options that he could pursue:

- *Stub page*. Since the website was much less important than it used to be, he could simply remove everything from the website and just have a simple landing page that made a statement to the effect: "Hi, I am Grandon Gill--go to my school account to find out more." In choosing this option, Gill could retain the domain name as well as continue using his permanent email address Grandon@Grandon.com. This would be the easiest of all the options--and probably the most secure.

- *Basic website:* Option two was to move to another host and take the key static elements of the various pages that he had created for his original site and tie them into a simple static website with a powerful password. He guessed this could take 4-5 hours, and he could prepare the site offline using the Dreamweaver HTML development tool that he employed for his original site. Since the website would be entirely static, it would become incredibly hard for the hackers to hack again. The downside of this option was that the site would appear old fashioned and would require HTML coding to modify.
- *Recreate original site on a different server*. A very different possibility would involve moving the original website entirely to a new server, hosted by a different company, and then restore the WordPress files from backups. He had succeeded in doing this on a local server (used to create Exhibit 2). The time required to do this would be more than just putting up a static website—perhaps 5-10 hours—but it would provide him with the greatest functionality. By paying for a host that specifically featured WordPress hosting, he could also arrange for automatic updating. While this would restore all the content and blog posts from the original site (hundreds of them), he was not sure that he could ensure that he did not inadvertently reestablish a vulnerability carried over from the original site.
- *Build a WordPress-only site*. He could move to a WordPress host and start afresh, without worrying about backdoors and vulnerabilities--starting afresh (and planning to be more careful). Rather than have his pages created from scratch, calling WordPress RSS feeds (as done in the original site), this option would involve creating pages within the WordPress site itself. This site would be easy to maintain and, assuming a host was selected that did automatic updating, would likely be reasonably secure. The biggest drawback was the time it would take to create a decent fully-functional site (10-20 hours was his best guess). Another drawback was that WordPress made it hard for the novice-to-intermediate author to customize themes; doing so ran the risk of having modifications overridden the next time WordPress or the theme updated. He would lose all his old content as well.
- *New architecture altogether*. As a final option, he considered moving the website to an entirely different architecture. WordPress was good, but the popularity also carried with it a potential for hacks. He wondered if he could choose another content management system with less likelihood of getting hacked. This would again take up a lot of Gills time—20-50 hours as a speculative guess, including the time required to figure out the system—but, obviously, it would involve the greatest amount of learning.

No matter what option he chose, the decision had to be made soon. It was embarrassing to have a website labeled “Temporarily Disabled”. This didn’t speak well for his professional skills… While he was thinking about making an appropriate decision, he remembered a cautionary saying about decision-making: "Making a decision takes a moment. But living the decision takes a lifetime.”

References

Ami, P. V., & Malav, S. C. (2013). Top five dangerous security risks over web application. *International Journal of Emerging Trends & Technology in Computer Science*, *2*(1), 41-43.

Caldwell, T. (2011). Ethical hackers: Putting on the white hat. *Network Security*, *2011*(7), 10-13. doi: 10.1016/S1353-4858(11)70075-7

Cid, D. (2014, October 6). Phishing with help from compromised WordPress sites. [Blog post]. Retrieved from https://blog.sucuri.net/2014/10/phishing-with-help-from-compromised-wordpress-sites.html

Cid, D. (2017, February 9). RCE attempts against the latest WordPress REST API vulnerability. [Blog post]. Retrieved from https://blog.sucuri.net/2017/02/rce-attempts-against-the-latest-wordpress-rest-api-vulnerability.html

Crowder, D. A. (2010). *Building a web site for dummies*. (4th ed.). Hoboken, N.J.: Wiley.

Das, A. K., Nguyen, Q. T., & Thomas, S. (2017). Entertaining whilst defacing websites: Psychological games for hackers. *Issues in Informing Science and Information Technology*, *14*, 219-227.

Davis, M. E., & Phillips, J. A. (2007). *Learning PHP & MySQL: Step-by-step guide to creating database-driven web sites*. (2nd ed.). Beijing: O'Reilly Media, Inc.

Gill, T.G. (2016). Securing the Muma journals. *Muma Case Review, 1*(1), 1-24. Retrieved from http://pubs.mumacasereview.org/2016/MCR-01-01-Gill-SecuringJournals-p1-25.pdf

Floyd, K., Harrington, S. J., & Hivale, P. (2007, September). The autotelic propensity of types of hackers. *In Proceedings of the 4th annual conference on Information security curriculum development* (p. 16). ACM.

Levy, E., & Arce, I. (2006). New threats and attacks on the world wide web. *IEEE Security & Privacy*, 234-266.

Nagpal, B., Chauhan, N., & Singh, N. (2015). Defending against remote file inclusion attacks on web applications. *i-Manager's Journal on Information Technology*, *4*(3), 25-33.

Wang, Y., Li, Z., & Guo, T. (2011, August). Program slicing stored XSS bugs in web application. *In Fifth IEEE International Conference on Theoretical Aspects of Software Engineering (TASE)*. (pp. 191-194). IEEE. doi: 10.1109/TASE.2011.43

Watson, D. (2007). Web application attacks. *Network Security*, *2007*(10), 10-14.

WordPress. (2017). Features. Retrieved from https://wordpress.org/about/features/

Wpbeginner. (2012). How to find a backdoor in a hacked WordPress site and fix it. Retrieved from http://www.wpbeginner.com/wp-tutorials/how-to-find-a-backdoor-in-a-hacked-wordpress-site-and-fix-it/

Sinegubko, D. (2016). Phishing attacks target ecommerce checkout pages. [Blog post]. Retrieved from https://blog.sucuri.net/2016/07/phishing-attacks-target-ecommerce-checkout-pages.html

Sucuri Security. (2016). Website hacked trend report 2016 - Q1. Retrieved from https://sucuri.net/website-security/website-hacked-report

Xu, Z., Hu, Q., & Zhang, C. (2013). Why computer talents become computer hackers. *Communications of the ACM*, *56*(4), 64-74.

Acknowledgements

This case study is based upon work supported by the National Science Foundation under Grant No. 1418711.

Biographies

Taufeeq Ahmed Mohammed is a student at University of South Florida, doing his Master's in Business Analytics and Information Systems. While pursuing his interests in the field of Data Analytics and Decision Sciences, he also works as a Teaching Assistant and a Graudate Assistant, teaching undergraduates and writing business case studies respectively. He considers himself as a creative data scientist in the making. He is also a published author of a narrative non-fiction book and a commercial fiction book, and he intends to write more books in different genres in the near future.

Utkarsh Shrivastava is a doctoral candidate in the Information Systems and Decision Sciences department at the University of South Florida in Tampa, Florida. He also has a Bachelor's degree in Information Technology and an MBA from Indian Institute of IT & Management. His research interests include health information technology, statistical data mining, ICT for development, and cyber security. He has taught systems analysis and design and applied data science courses to undergraduates at USF. Utkarsh likes to travel and present his work at research conferences in the information systems and business analytics domain.

Ashish K. Das is a Lecturer in Business Information Systems at Royal Melbourne Institute of Technology (RMIT) University Vietnam. He has more than 12 years of work experience in the software development industry in the U.S., including Software and Solutions Architect and Program Manager positions in various Fortune 500 companies and start-ups.

Quynh T. Nguyen (PhD) is a certified IBM SPSS & Modeler Specialist and Data Analytics Specialist, having obtained her Master's degree (with distinction) and PhD from Coventry University, UK, in 2007 and 2016, respectively. She has published 31 papers in peer-reviewed journals and conferences and is an invited speaker at De La Salle University, Manila, Philippines.

Exhibit 1: Email from Gate.com

From: Technical Support [xxxxxxxx]
Sent: Sunday, December 25, 2016 8:32 AM
To: Thomas Gill <xxxxxxx >
Subject: Ticket Solved: #xxxxxxxx:Incident-xxxxxxxx-Website Security Notificaiton -grandon.com
Your ticket (# xxxxxxxx) has been solved. To reopen this ticket, please reply to this email.

XXXXX (Gate Help Center)

Dear Valued Customer,

Thank you for choosing xxxxxxxxx. We are sending you this e-mail to notify you that our System Administrators have detected some suspicious activity on your hosting space for grandon.com. It appears the site might have been compromised as it is hosting a phishing website. Our Administrators were also able to find the phishing kit used. We have disabled the webhosting space, to prevent the abuse. We do strongly recommend the following steps in order to increase the security of your web hosting account:

1) Change FTP password via your SiteControl. A good password contains lower and upper case letters, numbers and special characters, and should be approximately 8 characters or more.

2) Have your web designer evaluate all your website scripts.

3) If you are using some sort of CMS software (Like Joomla, Wordpress and such) we suggest you have it always updated to the latest release and version.

4) Check your computer for viruses and malicious software. Note that such attack might be caused by malicious software installed on your computer that infects the web files upon uploading.

5) Remove the phishing scripts used.

The URL(s) of the fraudulent site:

hxxp://grandon[.]com/info/websc-login[.]php

The IP address hosting this phish is xxx.xxx.xxx.8

Please investigate and shut down the site(s) immediately.

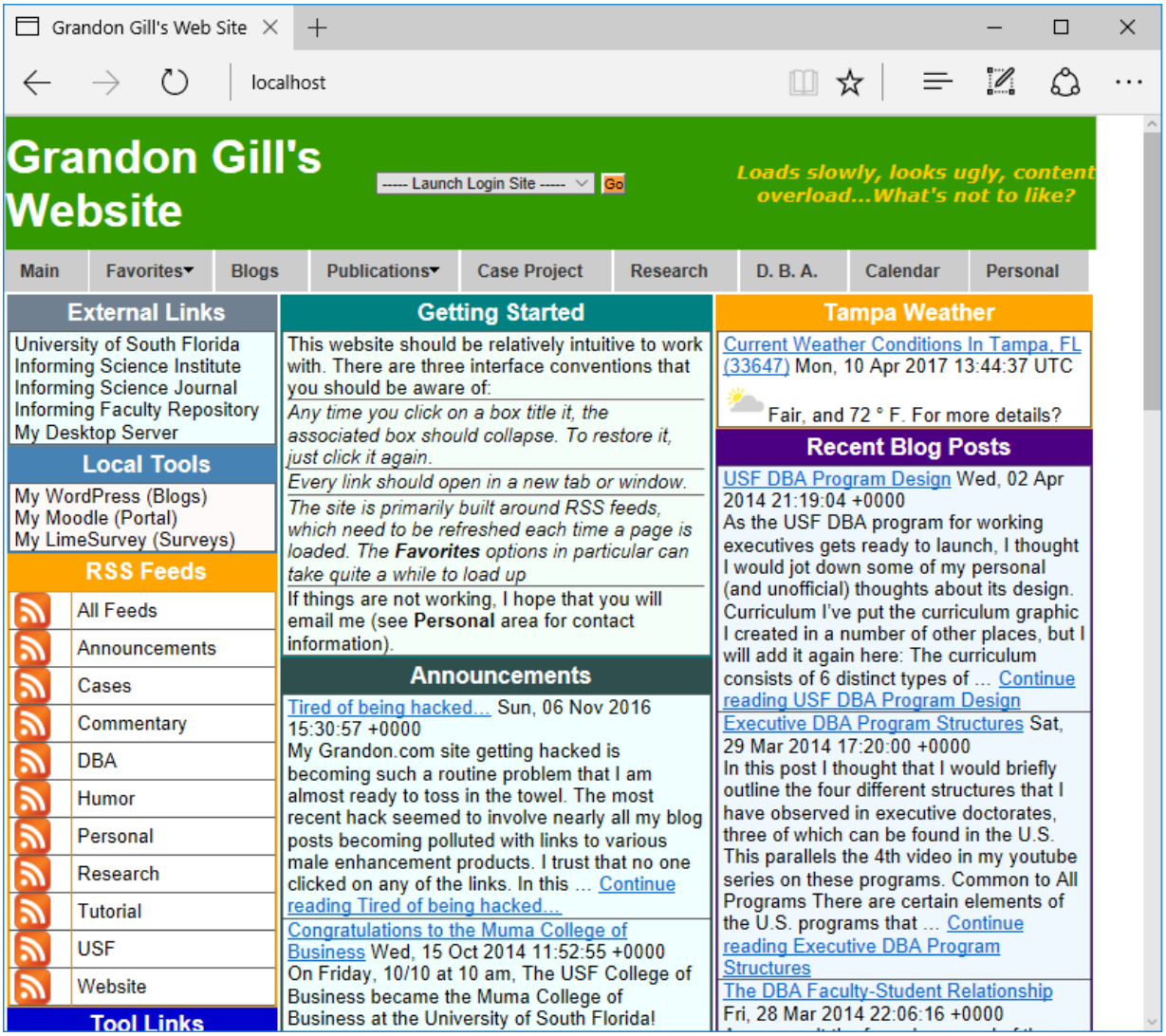
Should you have any questions, please call us at +1-xxx-xxx-xxx6.
Please include the Ticket number(s), MM# xxxxx, in all communications on this issue.

Thank you,

xxxxx xxxxx

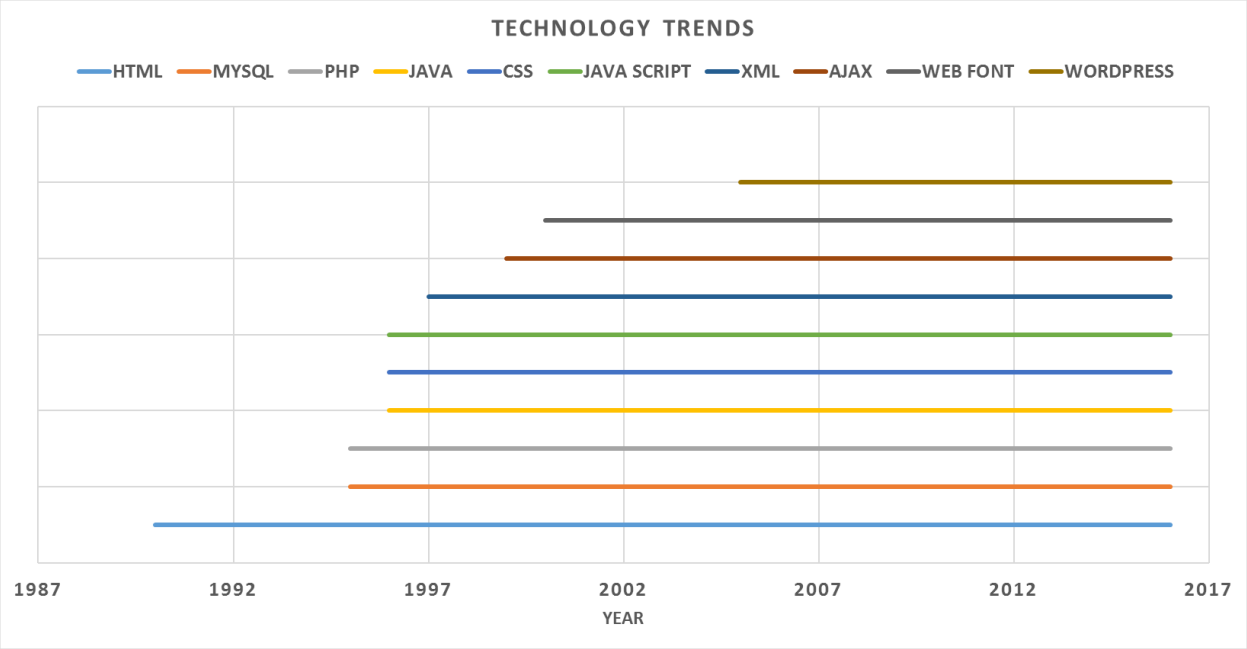
Source: Email from Gate.com technical support to Grandon Gill

Exhibit 2: Grandon.com



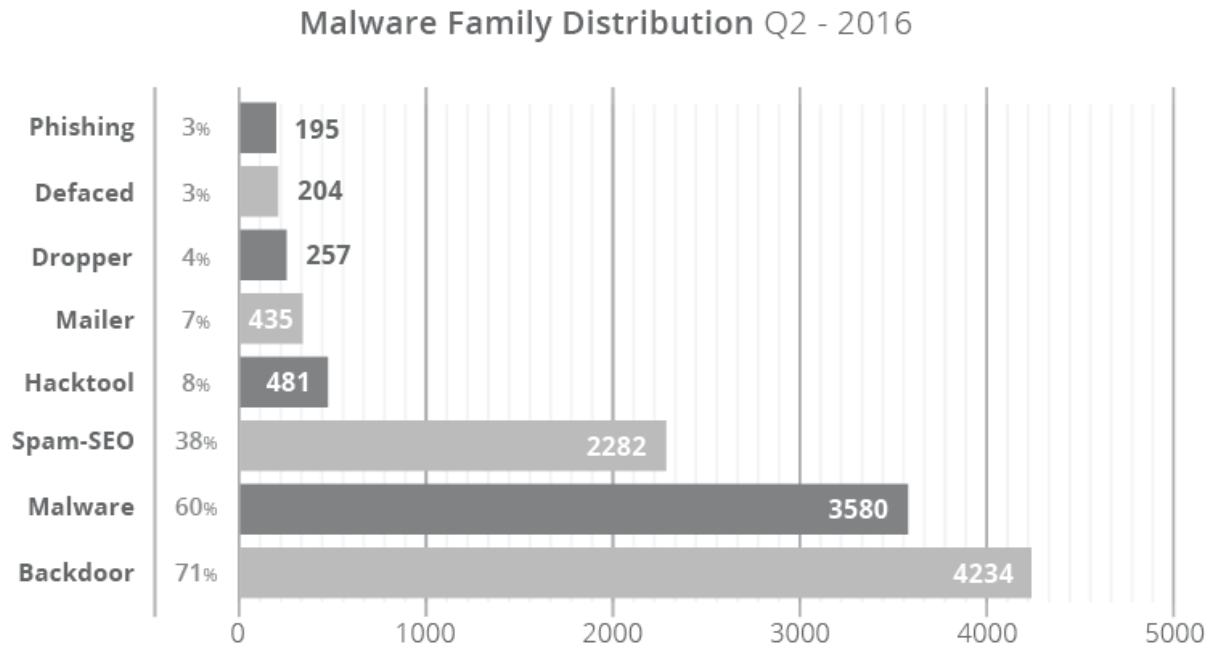
Source: Grandon.com web site

Exhibit 3: Technology Trends



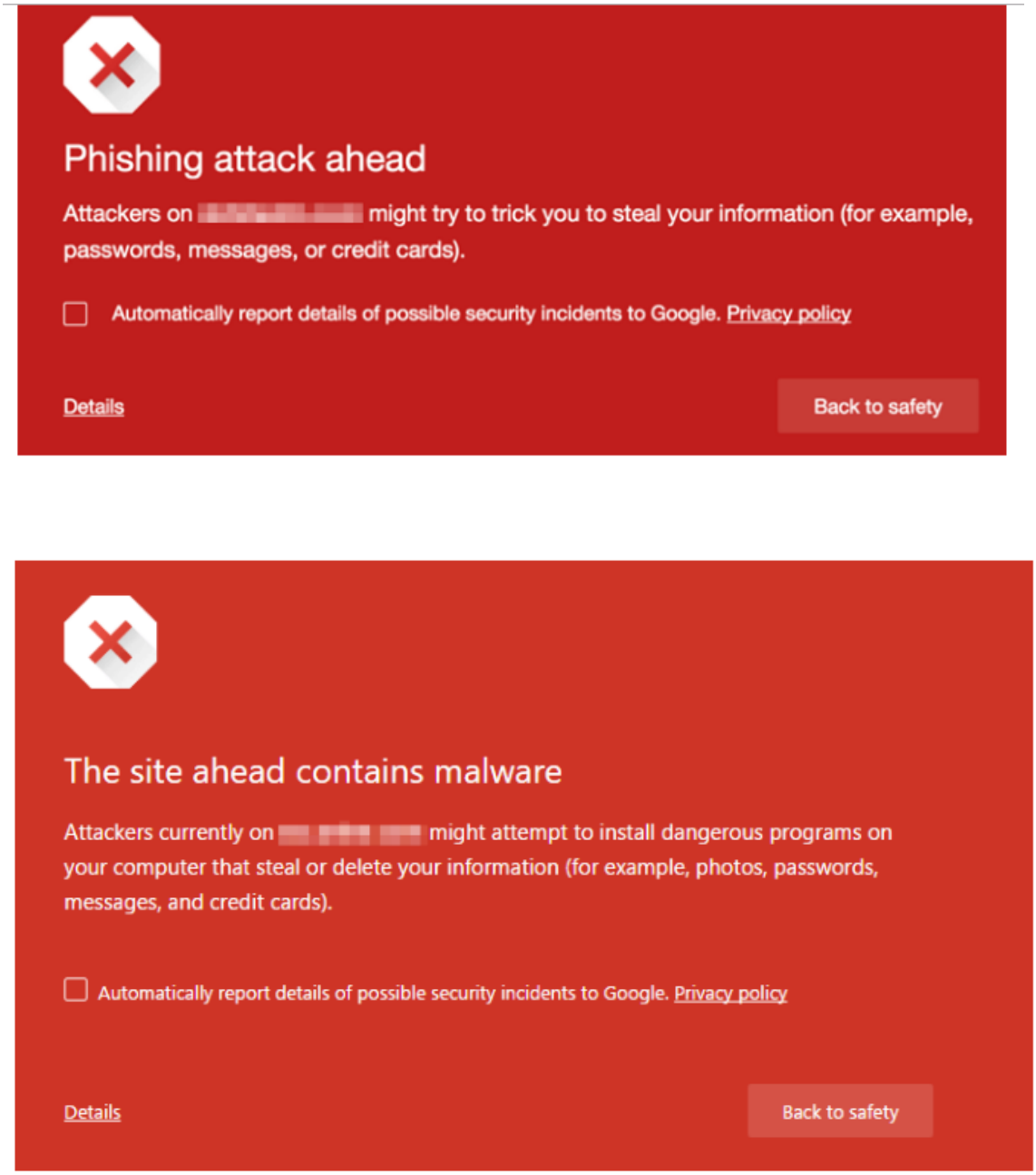
Source: http://www.evolutionoftheweb.com/

Exhibit 4: Distribution of Hacking Attacks in 2016



Source: https://sucuri.net/website-security/hacked-reports/2016-q2-hacked-website-report

Exhibit 5: Warnings from a Web Browser (Google Chrome) on Visiting a Hacked Website



Source: https://www.wordfence.com/learn/has-my-site-been-hacked/

Exhibit 6: Website Vulnerabilities that Lead to Hacking

A hacking attack could exploit the following user, application or system vulnerabilities.

Use of Reusable Code: Writing code from scratch can be time-consuming. Web developers therefore often included code files from other open source outlets. The improper naming of these open source files could inadvertently expose PHP code or other security-critical information to the hacker. As an example, the PHP code is processed by the web server before presenting the web page, but due to improper file extensions, the server may display the web page (along with any usernames or passwords within) without processing. Slight awareness while developing the website such as use of the correct extension (e.g., PHP for PHP scripts in place of say .inc) and securing the sensitive included files in a folder other than published web root can deter hacking attempts. The access to a folder containing security critical files can also be restricted by using .htaccess file in the folder of the code (Davis & Phillips, 2007).

Definition of Global Variables: The data exchange between a website visitor and web server was facilitated by GET and POST operations. The older versions of server scripting languages such as PHP did not require the variables to be predefined. By default, these variables were named and were accessed from GET or POST operations as global variables. A hacker can call the PHP script with either a GET or POST operation and name the returned value exactly the same as some other critical task variable. As an example, the value returned can be named exactly as the variable that was used to indicate if the password matched or not. By arbitrarily assigning the required value to this task-critical variable, a hacker can bypass the authentication and access the data on the server. Similarly, the functionality of a program can be altered by the introduction of a false parameter by the hacker.

This security loophole can be fixed by manually initializing the variables generated by the GET or POST operations. There was also a setting called register global in the PHP.ini configuration file which can be turned off and the data passed through GET or POST operations would be visible as the global variables. Some of the open source code scripts writing in older PHP versions by default may have assumed that this option was enabled and therefore manual initialization of the variables would require modifying all the dependencies.

***Unverified User Inputs to the Database*:** The user inputs accepted by the website should be checked and verified before entering into the database. For example, it should be checked for the special characters, such as single and double quotes or an escape character. The RDBMS also provides certain functions such as mysql_real_escape_string to validate user inputs. User inputs embedded with the HTML scripts were often used for cross-site scripting or SQL injection attacks. Functions such as HTML entities can be applied to the user inputs to reject HTML elements within the use inputs. The key sources of user data in any website included: GET and POST operations, cookies, sessions data, and global server variables. Further, using default PHP settings on a shared server could be dangerous because a user's sessions data was stored in a temporary directory which was shared by all users. A hacker can access the session id of a user and can gather private information related to the session. Therefore, on a shared server, the session's data should be stored in a separate directory with restricted access to other applications hosted on the server.

Availability of Database Information: The information related to the database such as its location on the web server, IP address, or database port number should be hidden from the hackers. Often, when there were problems connecting to the database, the MySQL error code revealed the IP address of the host. The database information could be utilized by the hacker to gain control over it. A PHP error control function could be used to display a different error message. Similarly, the database port information would enable an external host to take the services of the hacked database. The access to the database port for the

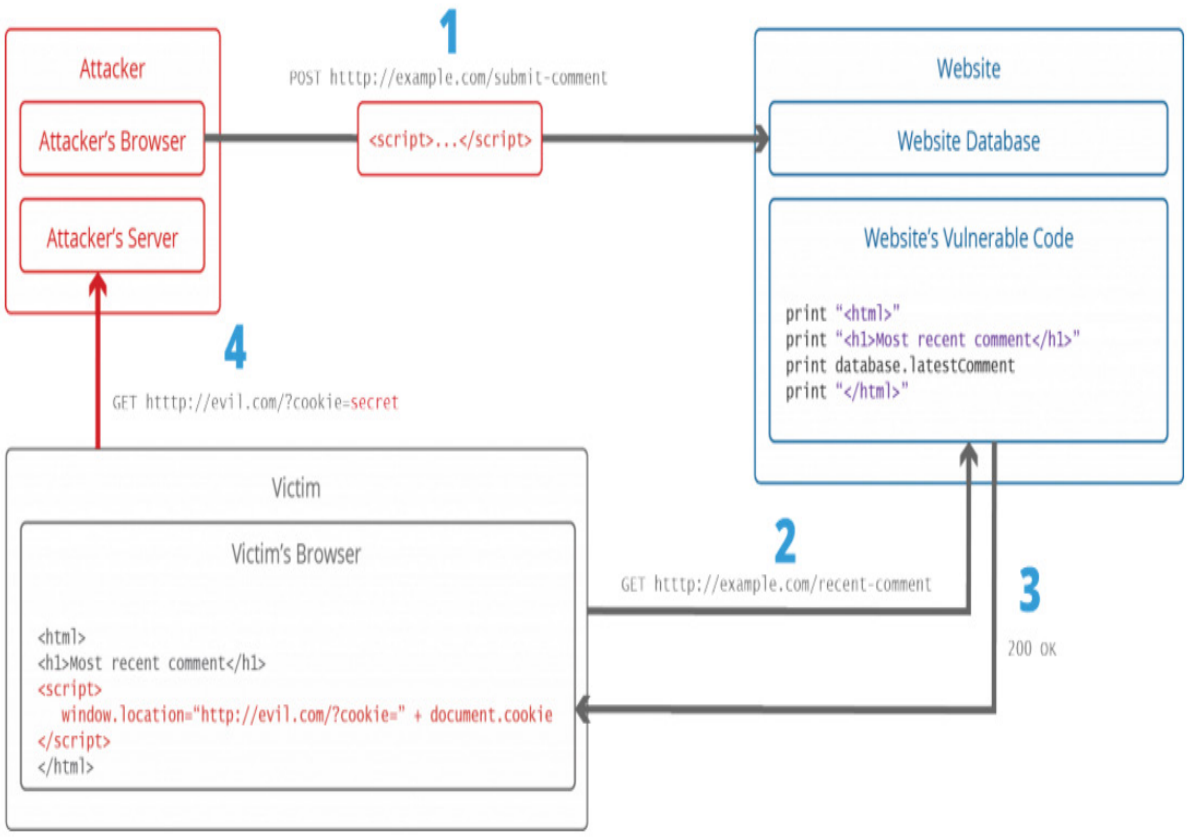
external applications should, therefore, be restricted using firewall software. Further, if more than one application were running on the server, then a separate database for each application should be set up on the RDBMS (e.g., MySQL). By doing so, the data related to an application could be protected from the security breaches on another application.

Hosted Server: A shared web hosting plan was attractive for its reduced cost and maintenance. Multiple websites hosted on a shared server had a separate domain, content, and applications, but they often competed for the limited server resources. In a shared hosting environment, an insecure script associated with a website could give access to the complete server to the hackers. If all the websites hosted shared the same IP address, then a denial of service attack on one website may put down all the sites hosted on the server. Similarly, if a website was placed on the spam blacklist, then all the websites with the similar IP address on the shared server would follow the same fate.

WordPress Application: Being an open source project, anyone could contribute to it by writing patches, plugins, themes or offering technical support. Therefore, the out of the box WordPress installation was pretty secure, but the use of plugins, themes, scripts from other sources increased vulnerability to security risks. Incompatibility of the WordPress installation with the web server features may also make the use of WordPress on the website insecure.

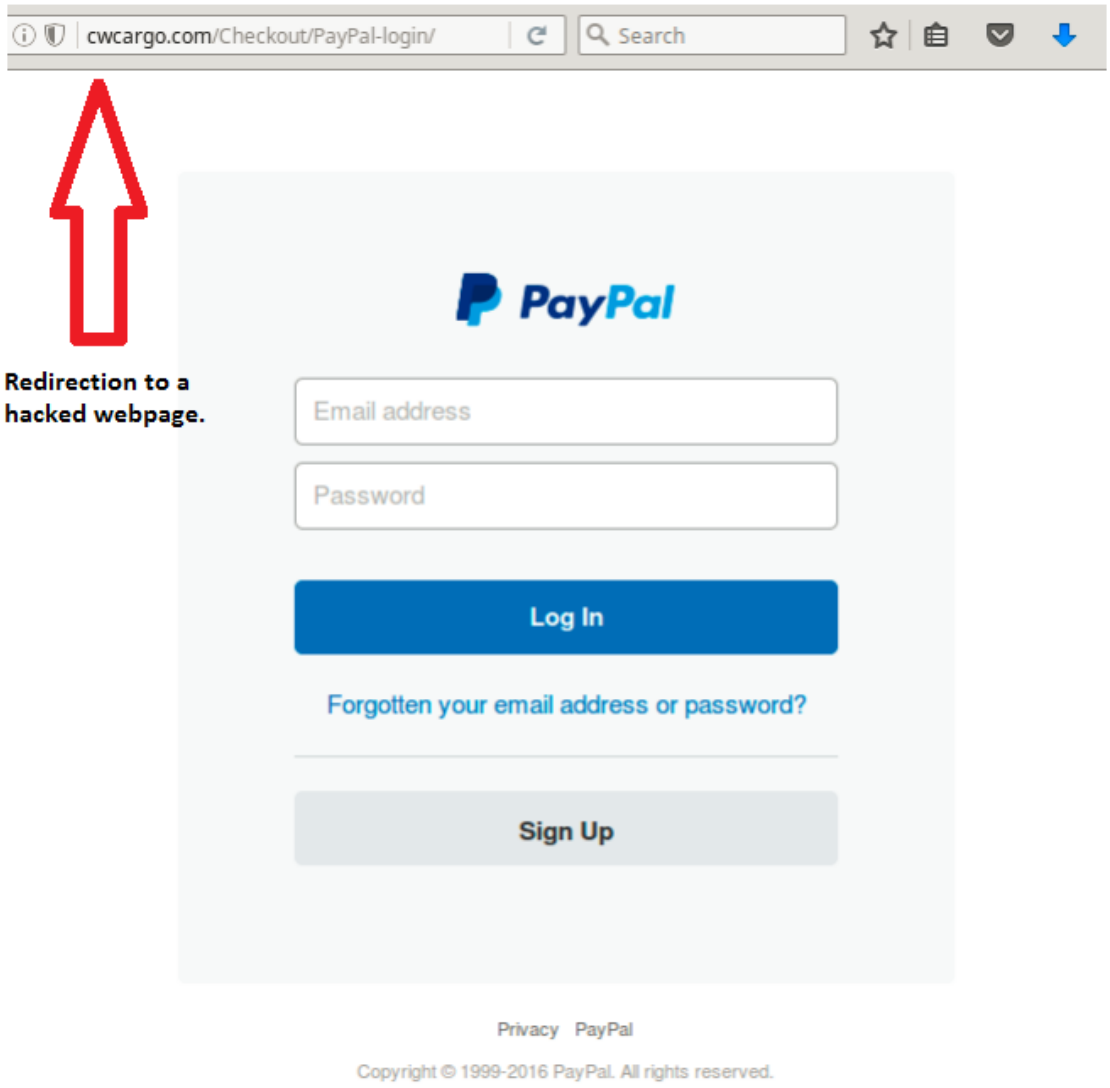
Source: Developed by case writers

Exhibit 7: Cross Site Scripting Attack Mechanism



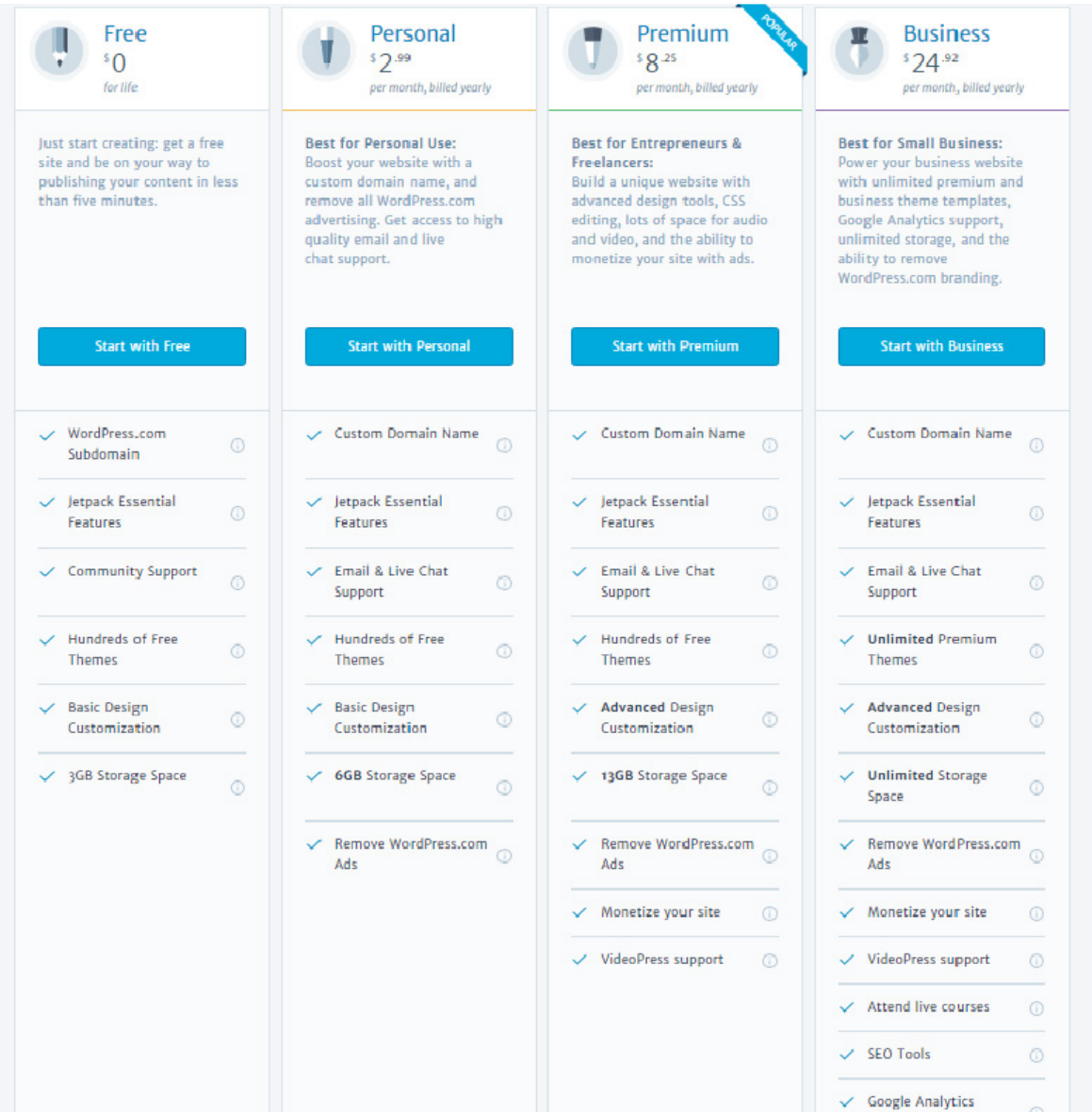
Source: https://www.acunetix.com/websitesecurity/cross-site-scripting/

Exhibit 8: Phishing Attack



Source: Prepared by case writers

Exhibit 9: WordPress Paid Plans



Source: https://wordpress.com/pricing/

Exhibit 10: Ransomware

Source: https://arstechnica.com/security/2016/02/mysterious-spike-in-wordpress-hacks-silently-delivers-ransomware-to-visitors/