

Prevent Web Site Defacement

Defacement to your Web site could have extremely damaging consequences for your e-business. Here's how to prevent this growing problem.

By Dr. Yona Hollander

TECHNOLOGIES

- Application access control
- OS level protection
- HTTP attack protection
- Web server resources protection

Web defacement occurs when an intruder maliciously alters a Web page by inserting or substituting provocative and frequently offending data. The defacement of an organization's Web site exposes visitors to misleading information until the unauthorized change is discovered and corrected.

Web page defacement is surprisingly common: 19 percent of the respondents in the 2000 CSI/FBI survey reported their Web site had suffered unauthorized access or misuse, with an additional 32 percent stating they didn't really know whether they had been subject to tampering. The most common type of misuse reported in the survey was Web site vandalism (64 percent of the attacked sites), far exceeding proprietary information theft (8 percent). It's very reasonable to assume that most Web site vandalism involves site defacement incidents.

To witness the real-life impact of Web site defacement, you can visit:

<http://www.attrition.org/mirror/attrition/>

which closely tracks incidents and keeps a 'mirror' of defaced Web sites. Statistics show that the number of incidents on average is about 10 per day. This means that you can access this site every single day and find new organizations whose Web presence was defaced during the previous 24 hours.

At first glance, the popularity of Web defacement is mystifying. Why do hackers make such an effort to deface a Web site? It would seem much more logical for them to look for credit card numbers and other valuable proprietary information. It's unclear what the reward is for hackers, other than the obvious credibility in the hacking community and, in some high-profile cases, 15 minutes of fame through media coverage of the incident.



Dr. Yona Hollander is vice president of strategy for Enterscept Security Technologies, a leading provider of proactive anti-hacking solutions for e-business. Hollander is responsible for identifying security market trends and engaging Enterscept's product development resources to engineer new and innovative anti-hacking solutions. <http://www.enterscept.com>.

Web defacement is a significant and major threat to businesses developing an online presence. Defacement of a Web site can detrimentally affect the credibility and reputation of the organization as a whole. Unlike other attack cases where the hacker hides his activities, in defacement incidents, the major goal of the hacker is to gain publicity by demonstrating the weakness of the existing security measures. The damage from a Web defacement incident can be disproportionate. Damage can range from loss of customer trust to loss of revenue. An e-retailer can lose considerable patronage if its customers feel its e-business is insecure. Financial institutions, which emphasize security and credibility, may experience significant loss of business and integrity, due to security breaches in their Web site. Along the spectrum, consumer confidence and loyalty in these organizations can have serious negative implications.

There's an overwhelming need for a solution that eliminates compromises to the Web server, especially Web page defacement. Ideally, it would prevent the hacker from making any modifications, thereby precluding any possibility of attracting attention. The Web server would never present a defaced page to a user. Equally important, a proactive solution would eliminate any after-the-fact need for recovery and fixes, and be transparent to standard operations.

How a hacker defaces Web pages

Although it sounds simplistic, obtaining usernames and passwords is a very popular and effective technique used by hackers to break into a site and deface it. To retrieve this information, hackers use the following: information-gathering techniques, which exploit vulnerabilities in the system (e.g., read Web pages such as 'global.asa' that are supposedly not viewable from the outside), making use of publicly

available information (e.g., domain registration records), or using 'social engineering' tactics (e.g., calling an employee and posing as a system administrator). If the hacker has a username, he can try to guess the password by going through a list of popular or default choices, or by using intelligent guesses. Social engineering helps here, too—birth dates, names of family members, etc., are all prime candidates. In an amazingly large number of cases, these techniques lead to success: authenticated access to the system.

After the hacker is logged on to the system, he tries to escalate his privileges, i.e., obtain system administrator privileges. Both Windows NT and UNIX provide a "superuser" account (administrator in NT, root in UNIX); as this account has full access rights to all system resources, it's the ultimate goal of any hacker to own it. At this stage, the hacker does some additional information gathering to find out useful tidbits: the exact version and patch levels of the operating system, the versions of software packages installed on the machine, and services and processes enabled. Using this information, he accesses well-known Web sites and easily locates hacks that exploit vulnerabilities existing in the software installed. When these exploits are executed on the machine, the hacker ends up gaining privileged access rights, and actually controls the machine. At this stage, if he's interested in defacing the Web site, he simply modifies the content of the pages. To the system, it's business as usual, as the intruder works in the security context of a privileged entity.

Sechole

An example of a privilege escalation exploit on Windows NT4 is Sechole. The attack modifies the instructions in memory of the OpenProcess API call so it can successfully attach to a privileged process, regardless of whether it has the permission to do so. After it's attached, it runs code within that process that adds the user to the Administrators group. This technique works if the code runs locally on the machine. However, if Microsoft's Internet Information Server (IIS) Web server is present and certain other conditions are met, Sechole can be launched from a remote location, adding the IIS Internet User Account, IUSR_machine_name, to the Administrators or the Domain Admins group. After this is done, this account has privileged access rights and can help the hacker bypass all existing security measures.

Another approach is to exploit vulnerabilities in Internet servers that are listening to open ports. In this case the hacker typically doesn't need to log on to the server at all. By exploiting a weakness in the server, he can remotely execute malicious code over an open legitimate connection. One possible outcome of the code execution is the modification of a Web page. A popular hacking technique is to start an ftp client on the victim machine that simply downloads 'replacement' Web pages from the hacker's ftp server (which, of course, is another box that has been taken over by him, preferably in another country).

IIS Hack

A well-known example for a remote attack on the IIS Web server is IIS Hack. IIS supports a number of file extensions that require server-side processing (e.g., ASP, .HTR, .IDC). When a request is made for one of these file types, the Ism.dll filter processes it. Unfortunately, vulnerability exists in Ism.dll, the library file that handles files with .HTR, .STM, or .IDC extensions. A carefully constructed file request exploits a buffer overflow weakness in Ism.dll that causes arbitrary code to execute in the security context of the System on the server. As a result of the code exe-

cution, the remote intruder can gain administrative privileges, or deface the Web site.

These are just a few examples of how Web sites can be compromised. Of course, there are hundreds more. Detailed information about hacking into Web sites is well documented and available for the average computer user.

Limitations of existing solutions

Firewalls currently are the main perimeter security tool, but they typically can't prevent Web defacement. They capture incoming communication packets and drop them if found to be malicious. For the Web server to operate, TCP/IP port 80 is always left open, as the HTTP protocol is using it. The firewall typically doesn't scan the incoming HTTP packets, and therefore can't identify malicious ones. From its point of view, the communication is legitimate and thus no packets are dropped. As a result, HTTP attacks (such as IIS Hack) aren't detected by the firewall.

Network-based Intrusion Detection systems (NIDS) vendors claim to detect some HTTP-based attacks. However, NIDS can't really prevent attacks on the spot. The NIDS listens to packets on the wire, but doesn't block the transfer of the packet. In many cases, the packet reaches its destination and is processed prior to its interpretation by the NIDS. As a result, it's common for an attack to be successful before the NIDS identifies it.

Host-based intrusion detection (HIDS) shares the same problem. Attack detection is delayed in virtually all cases until after the attack has been completed. HIDS scans logs periodically, creating a time interval in which there's no processing, and in which intrusion attempts go undetected, even though they've been logged by a system component.

Another approach that helps sites recover from Web defacement incidents is integrity assessment. According to this scheme, the system keeps a hash code (similar to a checksum) for any Web page reflecting the page's content. Periodically, the system compares a freshly computed hash code based on current Web page content with the saved hash code. If the page has been modified, as in defacement, the hash codes won't match. After such a discrepancy has been detected, the integrity system alerts the operator and, in some cases, it's able to restore the original page.

This approach provides a possible solution to Web defacement but needs to overcome some obstacles to be effective. First, the frequency of the hash code comparisons needs to be high; otherwise it can take a long time to identify any defacement. During this time, the defaced page will be served to users, and the calamity will be openly exposed to the public. This frequent comparison isn't necessarily trivial to do, though. When the site includes hundreds or thousands of pages, the performance penalty can be significant.

A case where integrity assessment tools have an even harder time is when the pages are generated dynamically (currently very frequent). In this case, there's no starting point from which the software can create a reference. In these cases the whole integrity assessment scheme collapses.

An effective solution

A hacker can apply multiple and different techniques to deface a Web site. He can exploit vulnerabilities in the operating system, the Web server, or within other Internet servers to break into the Web server machine. The diversity of the attack techniques and their different targets requires a multi-layered protection system that provides the following functionality:

PREVENT WEB SITE DEFAACEMENT

On-the-spot prevention—The attack should be identified at the service request level, probably at the system call or API call invocation. At this stage, the request hasn't executed yet. This is the perfect time since changes to the page have not yet been made. An effective technique is to use system call and API call interception. The interception routine is transparently activated prior to the execution of the request. It checks if the initiator is allowed to perform the request and whether the request is legitimate, i.e., not part of an attack. If the request is found to be legitimate, execution resumes with no further delay. If, however, the request is malicious, the call is failed and the attack is thwarted.

Administrator (root) resistant—Most hackers first gain privileged rights and then try to deface the site. Therefore, it's good practice to restrict the privileges of the Administrator account on a Web server machine. Instead of the 'Administrator' account, only a specific predefined user (the Web master) should be allowed to modify the Web site content and configuration. The system should enforce this rule and fail malicious use of the Administrator privileges.

Application access control—It makes no sense for an arbitrary application such as a text editor to modify a Web page (even if the user has the adequate privileges). A single predefined program should be used to edit and/or create Web pages. An effective solution should enforce this rule by making sure that access to Web pages can be done only by using this predefined program.

OS level protection—Many hackers exploit vulnerabilities in the operating system in an attempt to break into the Web server machine. The solution should be able to identify and prevent such attempts. In particular, buffer overflow attacks, which are very popular, should be prevented.

HTTP attack protection—There are many attacks that use the HTTP protocol to break into Web servers and the OS. A protection module, which scans incoming HTTP requests for malicious requests should be used. The module should be effective also when the communication is encrypted.

Web server resources protection—Hackers typically need access to Web server resources for them to succeed in their attempts. They may want to kill the Web server process, modify configuration settings, and manipulate the Web server user properties (see the Sechole case). The resources that must be protected include:

- Executables
- Configuration files (including the Registry in NT)
- Data files
- Web server process

The access to these resources should be restricted to a predefined set of users and to a predefined set of applications.

Other Internet server attack protection—Internet servers such as Bind (a DNS server), Sendmail (an SMTP server), and others are known to have many vulnerabilities that let a hacker gain administrative privileges. The solution should be able to pre-

vent such attacks by parsing the incoming communication stream and identifying malicious requests.

Conclusion-type header

Web defacement is a major threat to companies that own a Web server. The number of Web defacement incidents is large and the damage to the companies operating the Web sites is significant. Existing security products provide only a partial solution. In most cases, the solution isn't satisfactory since the attack is exposed to external users and recovery takes significant time, effort, and costs.

Protection requires an effective, multi-layered solution that prevents Web defacement before altered pages are exposed to the public. The solution must be based on System call and API interception so it monitors the activities at the request level before any damage occurs. This scheme must cover different software layers through which hackers break into the system including the operating system, Web applications, Web servers, and the HTTP layers. The solution shouldn't let an intruder who gained superuser privileges tamper with the system. Only predefined users with adequate privileges (that includes a predefined specific tool) should be allowed to modify the Web site content or configuration. Access to different Web server resources such as executables, processes, data files, and configuration files (including the relevant registry keys in Windows NT) should be monitored and protected from unprivileged users. **ADVISOR**