INTERHACK

Interhack
2599 E Main St #512
Columbus, OH 43209

VOX +1 614 545 HACK
FAX +1 614 545 0076
WEB http://web.interhack.com/

# Comments on *Guidelines on Securing Public Web Servers*

Matt Curtin

March 5, 2002

**Abstract**

On February 28, 2002, NIST published a draft of *Guidelines on Securing Public Web Servers* for public comment. This document considers that discussion and makes several recommendations for improvements.

I wish to thank NIST for the opportunity to review and to comment on the document and for its effort to create the document.

## Contents

# 1 General Remarks

Several issues can be raised generally, as they apply throughout the document. With regard to terminology, I have two remarks.

**Definition of HTTPS** The protocol identifier HTTPS is not defined by an IETF standards track document, and therefore several contradictory definitions are in use. However, it is desirable to use an unambiguous definition consistent with early use. "Secure HyperText Transfer Protocol" (S-HTTP) was an early IETF effort separate from Netscape's definition of Secure Sockets Layer (SSL), and the use of HTTP over SSL (HTTPS). Largely because of the widespread success of HTTPS, IETF abandoned its S-HTTP effort. IETF has also standardized SSL as Transport Layer Security (TLS) [2]. Usage of the term that is both unambiguous and in use in the IETF vocabulary is "HTTP over TLS" [5].

**Definition of Hacker** Use of the term "hacker" is also ambiguous in a computing context. As generally seen in the media, the term means someone who attacks without authorization, but dictionary definitions and recent trends in mainstream media reflect another, more historical, usage. The Internet itself is an invention of hackers, according to the more historical definition. The term "cracker" can be employed to resolve ambiguity in the case where an unauthorized attacker is concerned, or use of computer-specific jargon can be avoided altogether. E.g., "Use the server as a distribution point for illegally copied software, criminal tools, or pornography..." is unambiguous and avoids computer-specific jargon with no loss in meaning.

Additionally, there seems to be some missing discussion of URI sensitivity. Specifically, I strongly recommend that URIs never contain any datum that could be considered non-public, as URIs are recorded in numerous locations: web client logs, proxy caching server logs, and in some cases, third-party HTTP referrer logs. In no case should a URI contain data that would allow someone to access something he otherwise should not. For example, simply "hiding" a host by not publishing its existence is ineffective. Another example would be the use of user names and passwords in URIs, which is part of the specification, but bad security practice.

# 2 Improved Discussion of Security Principles

Section 3 seems to jump directly into specifics, enumerating the needs for identification of the components, without providing or citing any basis for the reasons why. The Saltzer-Schroeder design principles [6] are well worth considering in this context, not just from the perspective of implementing systems, e.g., as programmers, but for configuration and operation of systems. Perhaps the end of section 3.1 could include a review and brief explanation of each of these, particularly as they relate to web services. Detailed discussion of these design principles as they apply to web services can be found in my book, *Developing Trust: Online Privacy*

*and Security* [1]. The document encourages many of these practices, but there is little reason behind the practice. For example, passwords of eight characters or more are recommended, but there is no discussion of *work factor*, the real issue at hand, which will allow administrators to decide appropriate password length to fit their own policies.

The principles might be thus summarized:

**Economy of mechanism** Design should be as simple as possible, but no simpler. Complexity is often the root of security problems.

**Fail-safe defaults** The system's default behavior should be the safest, such that when failure occurs, the system will lose functionality, rather than losing security.

**Complete mediation** Rather than providing direct access to information, mediators that enforce access policy should be employed. Examples include file system permissions and error-checking proxies.

**Open design** System security should not depend on the secrecy of the implementation or its components.

**Separation of privilege** Functions (e.g., read, edit, write, delete, supersede) should be separate, providing as much granularity as possible.

**Least privilege** The ability to perform one function should never imply another. Access should be granted as explicitly as possible.

**Least common mechanism** When providing a feature to the system, it's best to have a process or service gain some function without granting the same function to other parts of the system. The ability for the Web server process to access a back-end database, for example, should not also enable other programs on the system to access the back-end database.

**Psychological acceptability** People need to be able to understand what they're using, and need to be presented sensible options that will give them the flexibility and manageability they need to operate systems safely. Giving too many options, bad options, or confusing options, is as bad as providing no options at all.

**Work factor** System operators should understand what it would take to break the system's security features. The amount of work necessary for an attacker to break the system should exceed the value that the attacker would gain from that work.

**Compromise recording** When the system finally will give way, a system that records the compromise will help operators to avoid relying on something that they should not, and can also assist in the identification, prosecution, and litigation of attackers. At least if the system cannot defend against the attack, it should provide its operators some recourse.

# 3 Specific Per-Section Comments

The remainder of this document deals with comments I have on particular sections of the document.

## 3.1 Enforcing Minimal Access by Packet Filtering

In the context of disabling unneeded services, it's also noteworthy that web servers generally have no need to make client connections to the Internet. Packet-filtering routers can be used to enforce that policy. If administrators enforced this policy, Code Red and its progeny would not have become the problems that they were.

## 3.2 Web Robots

Section 4.2's discussion of web robots is very helpful. One explicit warning that is missing, however, is that the Robot Exclusion Standard is *not* followed by ill-intentioned robots like Email Siphon and Cherry Picker. Further, some "email address harvesting" bots will use the robots.txt content to find areas of the site to crawl. Thus, the advice is sound, but it should be pointed out that the spam bots will not follow the directives.

## 3.3 Data Sensitivity

Section 5 falls into a dangerous trap, the identification of data sensitivity by how "personal" it is. This definition does not have the support of information science and sound security policy, but rather that of marketers attempting to avoid public outcry and litigation over their self-serving data handling practices. As an example, the document's definition of what constitutes personal information lacks medical data, yet many HIV-positive web users would find that data intensely personal. The document does include financial data, yet many public employees' salaries are a matter of public record. IP address is cited as "personal information" yet it is necessary for the web server to receive the request for information, and DNS PTR records might even advertise the name of the host. The attempt to classify broadly what is and isn't "personal" and therefore what may and may not be handled without consent is wrong-headed. Instead, it is better to follow the Salter-Schroeder advice of *fail-safe defaults*, not gathering anything except by what is explicitly allowed. It should never be acceptable to collect any data not needed for the transaction or to use it for any other purpose than the servicing of the request by default.

## 3.4 HTTP Cookies

Section 5's discussion of cookies also misses an important point, that sensitive data can be correlated and collected with any type of cookie. The only difference is that in session cookies, the lifetime of the cookie is generally shorter, thus increasing the *work factor* needed to correlate data from session to session. However, the issue with cookies is not one of data sensitivity, but one of nymity. Cookies are most

often used to give users unique identifiers, which are pseudonyms that last for the duration of the cookie. Pseudonymity is very different from anonymity, and the risks presented pseudonymous users are different from the risks presented properly anonymous users [4].

It should also be noted that cookies, even when marked "secure", should not be used to transfer data that could be used directly by an attacker. In some environments, for example, persistent cookies would be stored on unencrypted network filesystems, such that a cookie received by the browser over a secure channel might be transmitted over the local network in the clear. This could have significant ramifications in the scenario of an attacker that has access to the same network as a legitimate user [3].

## 3.5 Blurring Code and Content

Section 5.3.1's discussion of risks presented by formats like JavaScript, PDF, and ActiveX is good, but a simple metric for determining the level of risk is not presented. I offer the following: the degree to which the line between data and code are blurred. In the lower-risk categories, risk is low because although some code is present, these are primarily data. An ActiveX control, on the other hand, is explicitly code, and code that runs closer to the machine than that which would run through some kind of interpreter. The same metric can be used to determine the risks in server-side dynamic content like ASP and JSP. (Significantly, the greater the risk presented, the higher the degree to which the Saltzer-Schroeder principle of *complete mediation* is violated.)

## 3.6 TLS Weaknesses

Discussion of TLS weaknesses in section 6.5.2 make it sound as though the man-in-the-middle attack is an unsolved problem or that manual verification of the browser's "Location" URI is sufficient protection. In reality, the TLS handshake will pass certificates, and the browser software will check the certificate's signatures. If a trusted signing authority has signed the certificate, the certificate will be accepted. However, the problem present here is that it places all trust in the signing authorities, many of whom the users have not heard, and some of which have been compromised. The weaknesses in TLS are not weaknesses in TLS per se, but are weaknesses inherent in any system requiring a centrally-administered public key infrastructure (PKI).

## 3.7 Triple-DES Ambiguity

Table 6.1 shows 3DES as having a 168-bit key. It isn't clear whether this is an oversight (asserting 3DES always has a 168-bit key) or a specification (of a three-key mode of ANSI X9.52).

### 3.8 Obsolete Citation

RFC1945 is cited in section 6.5.5, but that's an old version of the HTTP specification. The present specification is RFC2616, with updates for TLS coming from RFC2817.

### 3.9 Testing for Vulnerabilities

Security testing discussed in Section 8.4 lacks an important warning: testing the security of a production system can have unintended consequences if the tester checks for weakness by attempting to exploit it. Security vulnerabilities are often the result of some kind of system failure, which can introduce additional side-effects, including instability and data corruption.

### 3.10 Version Control

Recommended practice for handling of content and configuration is generally sound, but it might be worth adding that the use of a revision control system can greatly increase the flexibility and stability of managing content, particularly where many persons are involved in development and operation.

## References

[1] Matt Curtin. *Developing Trust: Online Privacy and Security*. Apress, November 2001.

[2] T. Dierks and C. Allen. RFC 2246: The TLS protocol version 1, January 1999. Status: PROPOSED STANDARD.

[3] Paul Graves and Matt Curtin. Bank one online puts customer account information at risk. Technical report, Interhack Corporation, October 2000.

[4] Josyula R. Rao and Pankaj Rohatgi. Can pseudonymity really guarantee privacy? In *Proceedings of the 9th USENIX Security Symposium*, pages 85–96. IBM T.J. Watson Research Center, USENIX Association, August 2000. [online] http://www.usenix.org/publications/library/proceedings/sec2000/rao.html.

[5] E. Rescorla. RFC 2818: HTTP over TLS, May 2000.

[6] Jerome H. Saltzer and Michael D. Schroeder. The protection of information in computer systems. In *Proceedings of the IEEE*, volume 63, pages 1278–1308, September 1975.