What is EAP-FAST?

The first widely deployed EAP method for authenticating wireless networks was Cisco's Lightweight EAP (LEAP), sometimes referred to as EAP-CiscoWireless. LEAP performs mutual authentication by using an MS-CHAP version 1 exchange to authenticate the client to the server and vice versa. MS-CHAP is vulnerable to dictionary attacks. In April 2004, code to implement an on-line dictionary attack against LEAP was released as a tool called *asleap*.

asleap works by forging a message to kick a targeted client off the network, forcing a reauthentication. By capturing the authentication exchange, an attacker has the necessary data to launch a dictionary attack. LEAP users have two options to securing their networks:

- Ensure that the passwords used are strong and, therefore, are not vulnerable to dictionary attacks. In practice, this option will cause user difficulties and is hard to enforce.
- If ensuring that strong passwords are used is not possible or is too difficult, then move to a different EAP method that is not vulnerable to dictionary attacks, such as PEAP.

PEAP is perceived to be difficult because of the requirement for digital certificates to authenticate the server to the client. Many organizations used LEAP instead of a stronger method because it does not require any certificates. To replace LEAP, Cisco proposed a new EAP method, Flexible Authentication via Secure Tunneling (EAP-FAST).

Like PEAP and TTLS, EAP-FAST works in two stages. In the first stage, a TLS tunnel is established. Unlike PEAP and TTLS, however, EAP-FAST's first stage is established by using a pre-shared key called a Protected Authentication Credential (PAC). In the second stage, a series of type/length/value (TLV)-encoded data is used to carry a user authentication.

PACs consist of three components.

- PAC-Key: A 256-bit key used by the client to establish the TLS tunnel. This key maps as the TLS pre-master-secret. The PAC-Key is randomly generated by the AS to produce a strong entropy 32-octet key.
- PAC-Opaque: A variable length field that is sent to the server during the TLS tunnel establishment. The PAC-Opaque can only be interpreted by the server to recover the required information for the server to validate the peer's identity and authentication. For example, the PAC-Opaque may include the PAC-Key and the PAC's peer identity. The PAC-Opaque format and contents are specific to the issuing PAC server.
- PAC-Info: A variable length field used to provide at minimum, the authority identity or PAC issuer. Other useful but not mandatory information, such as the PAC-Key lifetime, may also be conveyed by the AS to the peer during PAC provisioning or refreshment.

PACs are automatically refreshed when they expire as part of the EAP-FAST protocol.

PACs are not stored by the authentication server. Rather than store a PAC for each user, the PACs are generated from a "master key." When a new PAC is created, it consists of a random key plus the PAC-Opaque. The PAC-Opaque field is generated by encrypting the PAC-Key with the master key from the RADIUS server. The client authenticates to the server because the PAC-Opaque it submits must match the PAC-Key it will use to establish the TLS tunnel. The server authenticates to the client by proving that it has the master session key used to link the PAC-Key and PAC-Opaque fields.

Distributing PACs to clients is a key challenge faced by networks built on EAP-FAST. The current draft of EAP-FAST simply states that "…provisioning of the PAC may be achieved using the same mechanisms as the provisioning of any other credential such as certificates or username/password credential types." The initial EAP-FAST draft included a provisioning protocol capable of automatic generation and distribution of PACs to authorized users, but it was removed from later drafts.

The automatic provisioning protocol in the first draft relied on an encrypted tunnel to protect the authentication of the client and the delivery of the PAC to the client. Because no trust relationship existed, the encrypted tunnel relied on an unauthenticated Diffie-Hellman key exchange to establish the tunnel. In theory, an attacker could mount a masquerade or man-in-the-middle (MITM) attack, and capture enough of the authentication exchange to mount a dictionary attack. For added security, a signed Diffie-Hellman key exchange could be used, though this would re-introduce the use of certificates to deployment without significant security benefit when compared to PEAP.