Tampering

3. Statt auf Standard-Kryptografie zurück zu greifen, haben Sie sich selbst einen Mechanismus zur Gewährleistung von Integrät oder für den Schlüsselaustausch ausgedacht. Ein Angreifer kann sich dies zunutze machen.

4. Ihr Code trifft Entscheidungen zur Zugangskontrolle an vielen unterschiedlichen Stellen, anstatt diese Funktion an zentraler Stelle (in einem Security Kernel) zu implementieren.

5. Ein Angreifer kann unbemerkt bereits übermittelte Daten er- neut übertragen, weil Ihr Code keine Zeitstempel, Sequenznummern oder ähnliches nutzt, um dies zu verhindern oder zu erkennen.

6. Ein Angreifer kann Daten an Speicherorten schreiben, an denen Ihr Code liegt oder die durch Ihren Code interpretiert werden.

7. Ein Angreifer kann Berechtigungen umgehen, weil Sie Namen nicht kanonisieren (normalisieren), bevor Zugriffsrechte geprüft werden.

8. Ein Angreifer kann Daten manipulieren, die per Netzwerk übertragen werden, weil Ihr Code keine Integritätssicherung vorsieht.

Fortsetzung umseitig

Tampering

Tampering cont.

9. Ein Angreifer kann Statusinformationen beeinflussen.

10. Ein Angreifer kann gespeicherte Daten verändern, weil die Berechtigungen (ACLs) zu wenig restriktiv sind oder eine Gruppe verwendet wird, die letztlich jedem Nutzer Zugriff gewährt.

J. Ein Angreifer kann auf eine Ressource schreiben, weil es keine ACLs gibt, oder weil jeder berechtigt ist (world writable).

Q. Ein Angreifer kann Parameter über eine Trust Boundary hinweg ändern, nachdem sie validiert wurden (z.B. in einem HTML hidden field, oder einem Pointer an eine kritische Speicherstelle im RAM übergeben).

K. Ein Angreifer kann Code mithilfe eines Extension Points einbinden.

A. Sie haben einen neuen Tampering Angriff erfunden.