Hunt said, Leaf owners were trying to figure out how to manipulate their vehicles because the app was unreliable and cumbersome. “A number of frustrated app users figured out all they needed to do was feed a URL into a browser to turn on their car’s heat,” Hunt said.

When Hunt originally took his findings to Nissan, he said they were all ears. Then, after the initial private disclosure, Hunt said, Nissan gave him the cold shoulder and still didn’t fix the problem. Weeks later, when Hunt pointed out customers were slowly figuring out the vulnerability on their own, he told Nissan he was going to post his research on his website.

“Nissan was not considering this situation urgent,” Hunt said. “I finally told Nissan I was publishing my findings before the vulnerability became more widely known and abused.” It was only then, on Wednesday, that Nissan shut down the app that was creating the problem.

**TOTAL RECALL: TROY HUNT BREAKS DOWN HIS NISSAN HACK**

by Tom Spring

February 26, 2016, 9:45 am

Last month, when researcher Troy Hunt argued the dangers of insecure APIs at a security workshop, little did he know hours later he would discover an API vulnerability that allowed remote access to onboard computers of 200,000 Nissan Leaf and eNV200 electric automobiles.

“After talking about the way applications can sometimes get APIs wrong, a workshop attendee goes back to his hotel room and 15 minutes later calls to say he has found something fishy with the Nissan Leaf smartphone app,” Hunt said in an interview with Threatpost, speaking about the discovery.

The vulnerability, it turned out, allowed anyone with the right Nissan Leaf and eNV200 vehicle identification number (VIN) to remotely access the car’s climate controls, battery status and GPS logs that included the dates, times and distances the car traveled.
symmetric cryptography

cs642
adam everspaugh
computer security
ace@cs.wisc.edu
Announcement

* Midterm next week: Monday, March 7 (in-class)

* Midterm Review session Friday: March 4 (here, normal class time)
today

* Provable security
* Vernam's one-time-pad
* Shannon's security definition
* Block ciphers (AES)
* Block cipher modes of operation
review
* Symmetric cryptography
  / All parties have access to a shared random string $K$, called the key
  / Short keys
    // AES: 128b, 192b, 256b
  / Fast
  / Good for long messages

* Asymmetric cryptography
  / Each party creates a pair of keys: a public key $pk$ and a secret key $sk$
  / Long keys
    // RSA: 2048b, 4096b, 8192b
  / Slow
  / Good for short messages
primitives

* Encryption
  / confidentiality
  / symmetric + asymmetric versions

* Message authentication codes
  / integrity, authentication
  / symmetric

* Digital signatures
  / integrity, authentication
  / asymmetric

* Key exchange
SSL ver 2.0 designed by Hickman at Netscape

Wagner, Goldberg break SSL ver 2

Freier, Karlton, Kocher design SSL ver 3.0

Bleichenbacher breaks RSA PKCS #1 encryption, used in SSL ver 3

TLS ver 1 released as IETF standard, based on SSL 3, many cryptographers involved

Vaudenay, Klima et al. padding attacks

Rogaway IV re-use insecurity

Brumley, Boneh remote timing attacks

TLS ver 1.1 released as standard

ancient history
* TLS was built via "design-break-redesign-break" iteration

* Some amount iteration is fundamental

* Designing secure protocols is really hard
  / the problems are rarely in the primitives

* Many other tools have similar stories:
  / SSH, IPsec, kerberos, WEP + WPA (WiFi), GSM (cell phone)
Provable security supplements "design-break-redesign-break" iteration with a mathematical approach.

1. Design a cryptographic scheme
2. Provide a proof of its security

[Shannon, 1946]

Formal definitions
- Scheme semantics and assumptions
- Security

Security Proofs
- Security of a scheme cannot be broken if assumptions hold
enigma

* Put yourself in Shannon's place in 1946

* Enigma is state of the art cryptography developed by the Germans

* Broken by the Allies

* There must be a better way…
* Vernam's one-time pad (1917)

* Fix message length L

* Kg: output random bit string K of length L

\[
E(K,M) = M \oplus K = C \quad \text{D}(K,C) = C \oplus K = M
\]
security notion

**Dfn.** A symmetric encryption is *perfectly secret* if for all messages $M, M'$ and ciphertexts $C$

\[
\Pr[ E(K, M) = C ] = \Pr[ E(K, M') = C ]
\]

where probabilities are over choice of $K$.

* Shannon's "perfect secrecy" notion, 1946

* Each message is equally likely to map to a given ciphertext

* Also: seeing a ciphertext leaks nothing about what message was encrypted
otp proof

Dfn. A symmetric encryption is *perfectly secret* if for all messages $M, M'$ and ciphertexts $C$

$$\Pr[ E(K, M) = C ] = \Pr[ E(K, M') = C ]$$

where probabilities are over choice of $K$.

* Thm. OTP is *perfectly secret*.

* For any $C, M$ of length $L$:

  $$\Pr[ E(K, M) = C ] = \frac{1}{2^L}$$
  $$\Pr[ E(K, M') = C ] = \frac{1}{2^L}$$

  $$\Pr[ E(K, M) = C ] = \Pr[ E(K, M') ]$$
K must be as large as M
Reusing K for M, M' leaks M ⊕ M'
Message length is obvious
Mallory can make undetected modifications
Mallory can submit random messages
=> will decrypt to something
provable security

* Cryptography as a computational science
* Use computational intractability as basis for confidence

1. Design a cryptographic scheme
2. Provide a proof that no attacker with bounded computational resources can break it

[Goldwasser, Micali, Blum, 1980s]

Formal definitions
- Scheme semantics and assumption
- Security

Security Proofs (reductions)
- Breaking scheme
- Breaking assumptions
* Provable security yields well-defined assumptions and security goals. Designers (and attackers) can focus on assumptions.

* As long as assumptions hold, we can be confident in the security of a cryptographic scheme.
typical assumptions

* Underlying primitives are hard to break
  / Factoring of large composite numbers is intractable
  / RSA permutation is hard to invert
  / Block ciphers (AES) are good pseudorandom permutations (PRPs)
  / Hash functions are collision resistant

* Confidence in primitives is gained by cryptanalysis, public design competitions
  / design-break-redesign-break over the years
symmetric cryptography
Correctness: $\text{Dec}(K, E(K, M, R)) = M$

with probability 1 over all randomness

**symmetric encryption scheme**
What security goals do we need from symmetric encryption?
1. **Confidentiality** (Mallory cannot read M)
2. **Integrity** (Mallory cannot later M)
3. **Authenticity** (Mallory cannot forge her own messages M)
M

E

C

D

Kg

K

R

Key generation

Implements a family of permutations on $n$ bit strings, one permutation for each $K$

Security goal: $E(K,M)$ is indistinguishable from a random $n$-bit string for anyone that doesn't know $K$
Let $C$ be a string chosen uniformly at random.

Can adversary distinguish between World 0 and World 1?

If this holds for all polynomial time adversaries, then $E$ is called a secure pseudorandom function (PRF).
* Advanced Encryption Standard (AES)

* Current standard for a secure block cipher

* Chosen by public competition, run by NIST, academic cryptographers

* Key sizes: 128b, 192b, 256b

* Block size: 128b
building a block cipher

key k

key expansion

k_1 \quad k_2 \quad k_3 \quad k_n

R(k_1, m) \quad R(k_2, m) \quad R(k_3, m) \quad R(k_n, m)

m \quad \rightarrow \quad \rightarrow \quad \rightarrow \quad \rightarrow \quad c

R(k, m): round function

AES-128 n=10

[slide credit: Dan Boneh, CS155]
Designing good block ciphers is a dark art

Must resist subtle attacks: differential attack, linear attacks, others

Chosen through public design contests

Use build-break-build-break iteration

aes round function
<table>
<thead>
<tr>
<th>Attack</th>
<th>Attack type</th>
<th>Complexity</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bogdanov, Khovratovich, Rechberger</td>
<td>chosen ciphertext, recovers key</td>
<td>$2^{126.1}$ time + some data overheads</td>
<td>2011</td>
</tr>
</tbody>
</table>

- Brute force attack against AES: $2^{128}$
- ~4x speedup
$M = m_1 m_2 m_3 m_4 \ldots m_L$

$C = c_1 c_2 c_3 c_4 \ldots c_L$

Electronic Code Book (ECB) mode

modes of operation
ECB is the\textit{natural way} to implement encryption with block ciphers. But it's\textit{insecure}. Basically $\rightarrow$ it's a complicated substitution cipher.

If $m_i = m_j$ then $E(k, m_i) = E(k, m_j)$
secure modes

CTR, GCM, any randomized mode

secure modes
\[ M = m_1 m_2 m_3 m_4 \ldots m_L \]

\[ IV := \text{rand()} \]

\[ C = c_0 c_1 c_2 c_3 c_4 \ldots c_L \]

**Counter (CTR) mode**

How do we do decryption? think-\textit{pair}-share
Cipher Block Chaining (CBC) mode

$$M = m_1 m_2 m_3 m_4 \ldots m_L$$

$$C = c_0 c_1 c_2 c_3 c_4 \ldots c_L$$

IV := rand()
Can attacker learn $K$ from just $c_0, c_1, c_2$?
Implies attacker can break $E$ (recover block cipher key)

Can attacker $m_1, m_2, m_3$ from $c_0, c_1, c_2$?
Implies attacker can invert block cipher without $K$

Can attacker learn one of $M$ from $c_0, c_1, c_2$?
Implies attacker can break PRF security of $E$

Provably: passive adversaries cannot learn anything about $M$ if $E$ is secure
What about forging messages?
* Provable security

* Vernam's one-time pad
  / Shannon's "perfect secrecy"

* Block ciphers (AES)

* Block cipher modes of operations
  / ECB - obvious, but insecure!!
  / CTR

* Exit slips
  / 1 thing you learned
  / 1 thing you didn't understand