From Defect to Doctrine

Security bug case studies

Fraser Tweedale
@hackuador

April 4, 2019

Dogtag PKI

Access control - definition

```
certServer.ca.authorities
  :create,modify
  :allow (list,read) user="anybody";
   allow (create,modify,delete) group="Administrators";
   deny (create,modify,delete) user="mallory"
  :Administrators may create and modify lightweight authorities
```

Access control - evaluation order

```
authz.evaluateOrder=deny,allow
```

Access control - processing (1)

```
if (order.equals("deny,allow"))
    entries = getDenyEntries(acls, op);
else
    entries = getAllowEntries(acls, op);

for (ACLEntry entry : entries) {
    if (evaluate(token, entry.getExpressions()))
        throw new EACLsException("permission␣denied");
}
```

Access control - processing (2)

```
if (order.equals("deny,allow"))
    entries = getAllowEntries(acls, op);
else
    entries = getDenyEntries(acls, op);

boolean result = false;
for (ACLEntry entry : entries) {
    if (evaluate(token, entry.getExpressions()))
        result = true;
}
if (!result)
    throw new EACLsException("permission␣denied");
```

CVE-2018-1080

Principle: avoid booleans; use custom types

Principle: avoid booleans; use custom types

Corollary: use tools that support custom types

- Enumeration types: Java, C#, C++11, TypeScript, Python**
- Algebraic data types: Rust, Haskell, Scala, Swift, F#, C++17

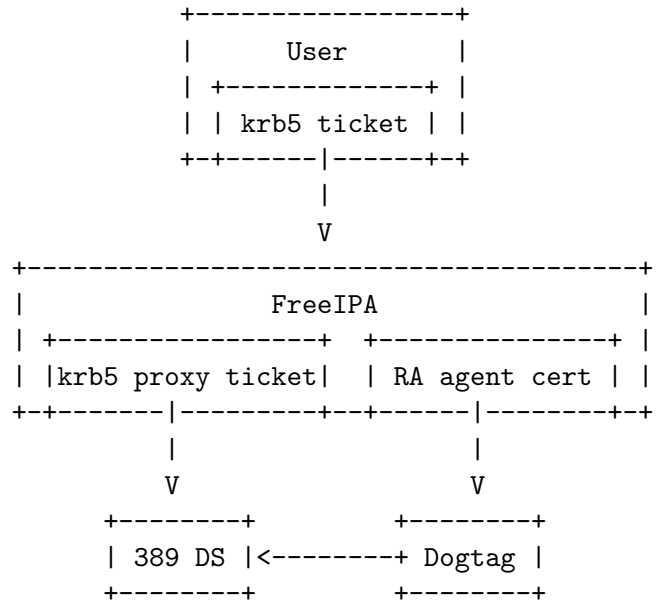
Access control - bools be gone!

```
public enum ACLOrder { DenyAllow , AllowDeny };
public enum ACLEntryType { Allow , Deny };
public enum ACLResult { Allowed , Denied };
```

Access control - fixed

```
if (order == EvaluationOrder.DenyAllow) {
    checkDenyEntries(token, acls, op);  // throws
    result = checkAllowEntries(token, acls, op);
} else {
    result = checkAllowEntries(token, acls, op);
    checkDenyEntries(token, acls, op);  // throws
}
if (result != ACLResult.Allowed)
    throw new EACLsException("permission␣denied");
```

FreeIPA


User
krb5 ticket
FreeIPA
krb5 proxy ticket
RA agent cert
389 DS
Dogtag

CVE-2016-5404

Principle: never cut corners on privilege separation

CVE-2017-2590

Firefox

Object Identifier (OID)

2.5.4.3	`commonName`
2.5.29.14	`subjectKeyIdentifier`
1.2.840.113549.1.1.11	`sha256WithRSAEncryption`

Object Identifier (OID)

1.3.6.1.4.1.311.21.8.8950086.10656446.2706058.12775672.480128.147.13466065.13029902

python-cryptography (bug)

```
buf_len = 80
buf = backend._ffi.new("char[]", buf_len)

res = backend._lib.OBJ_obj2txt(buf, buf_len, obj, 1)
```

```
backend.openssl_assert(res > 0)
return backend._ffi.buffer(buf, res)[:].decode()
```

python-cryptography (fixed)

```
buf_len = 80
buf = backend._ffi.new("char[]", buf_len)

res = backend._lib.OBJ_obj2txt(buf, buf_len, obj, 1)
if res > buf_len - 1:  # account for null terminator
    buf_len = res + 1
    buf = backend._ffi.new("char[]", buf_len)
    res = backend._lib.OBJ_obj2txt(buf, buf_len, obj, 1)
backend.openssl_assert(res > 0)
return backend._ffi.buffer(buf, res)[:].decode()
```

A buffer length of 80 should be more than enough to handle any OID encountered in practice.

— `OBJ_obj2txt(3)`

Object Identifier (OID)

0.8.8950086.10656446.2706058.12775672.480128.147.13466065.130299021.3.6.1.4.1.311.21
.8.8950086.10656446.2706058.12775672.480128.147.13466065.130299021.3.6.1.4.1.311.21
.8.8950086.10656446.2706058.12775672.480128.147.13466065.130299021.3.6.1.4.1.311.21
.8.8950086.10656446.2706058.12775672.480128.147.13466065.130299021.3.6.1.4.1.311.21
.8.8950086.10656446.2706058.12775672.480128.147.13466065.130299021.3.6.1.4.1.311.21
.8.8950086.10656446.2706058.12775672.480128.147.13466065.130299021.3.6.1.4.1.3

CVE-2017-7792

Firefox (bug)

```
char buf[300];
int i, n = 0, first = 1;
unsigned long v = 0;

for (i = 0; i < oid->len; ++i) {
  v = oid->data[i];
  if (first)
    n += snprintf(&buf[n], sizeof(buf) - n, "%lu", v);
  else
    n += snprintf(&buf[n], sizeof(buf) - n, ".%lu", v);
  first = 0;
}
```

Principle: don't rely on assumptions about input

Principle: don't rely on assumptions about input

Corollary: validate everything

Principle: use memory-safe languages

An unnamed online service

authentication - library (v1)

```
def authenticate(user, password):
    ... # talk to the database
    if all_good:
        return True
    else:
        raise AuthenticationError()
```

authentication - application

```
user = request['username']
password = request['password']

try:
    authenticate(user, password)
except AuthenticationError:
    respond_401_unauthorized()

# user is logged in
do_stuff()
```

authentication - library (v2)

```
def authenticate(user, password):
    ... # talk to the database
    if all_good:
        return True
    else:
        return False
```

Principle: avoid using exceptions

Principle: avoid using exceptions

Corollary: return types shall express failure cases

Principle: avoid using exceptions

Corollary: return types shall express failure cases

Corollary: use tools that ensure failure cases are handled

Error handling in Rust

```
enum Result<T, E> { Ok(T), Err(E) }

enum Error { AuthError(), ... }

fn authenticate(user: &str, password: &str)
                    -> Result<String, Error> {
    ... // talk to the database
    if all_good {
        return Ok(String::from(user));
    } else {
        return Err(Error::AuthError());
    }
}
```

Wrapping up

things that weren't covered

- cryptography
- input validation *in general*
- information flow control
- web things (XSS, CSRF, etc)
- so much other stuff...

principles

- avoid booleans; use custom types
- never cut corners on privilege separation
- don't rely on assumptions about input
- use memory-safe languages
- avoid using exceptions

Principle: learn from mistakes

Questions?

https://speakerdeck.com/frasertweedale

@hackuador

Access control - fixed

```
public enum ACLResult { Allowed , Denied };

List<ACLEntry> entries;
if (order == EvaluationOrder.DenyAllow) {
    entries = getDenyEntries(acls, op);
    entries.addAll(getAllowEntries(acls, op));
} else {
    entries = getAllowEntries(acls, op);
    entries.addAll(getDenyEntries(acls, op));
}

for (ACLEntry entry : entries) {
    Optional<ACLResult> result = entry.evaluate(token);
    if (result.isPresent()) return result.get();
}
return ACLResult.Denied;
```

Access control - fixed

```
data RuleOrder = AllowDeny | DenyAllow
data RuleType = Allow | Deny deriving (Eq)
data Result = Allowed | Denied

eval :: RuleOrder -> Token -> Op -> [Rule] -> Result
eval order tok op rules =
 fromMaybe Denied
  (alaf First foldMap (evalRule tok) orderedRules)
 where
  opRules = filter (elem op . aclRulePermissions) rules
  allowRules = filter ((== Allow) . aclRuleType) opRules
  denyRules = filter ((/= Allow) . aclRuleType) opRules
  orderedRules = case order of
   DenyAllow -> denyRules <> allowRules
   AllowDeny -> allowRules <> denyRules
```