Broken Access Control (A01: 2021)

Presentation to Programming management of a major hybrid chain

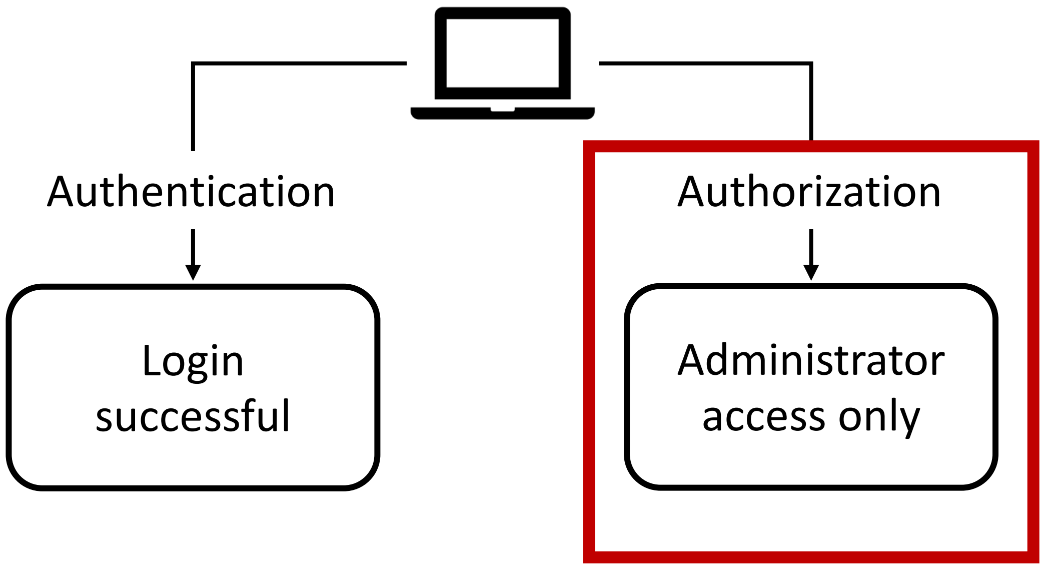
Kwabena Ohene-Bonsu

Outline

- Introduction
- Access Control Models
- Consequences
- Case study
- Prevention of Broken Access Control
- Access Control Principles
- Conclusion

Introduction

- **Access Control:** A security technique that regulates what or who has right to use or view resources in a computing environment. It is an essential concept in security that minimizes risk to an organisation.



Access Control Models

Mandatory access control (MAC)

1. The administrator configures access policies and defines security attributes: confidentiality levels, clearances for accessing different projects and types of resources.
2. The administrator assigns each subject (user or resource that accesses data) and object (file, database, port, etc.) a set of attributes.
3. When a subject attempts to access an object, the operating system examines the subject's security attributes and decides whether access can be granted.

Access Control Models

Discretionary access control (DAC)

1. User 1 creates a file and becomes its owner or obtains access rights to an existing file.
2. User 2 requests access to this file.
3. User 1 grants access at their own discretion. However, user 1 can't grant access rights that exceed their own. For example, if user 1 can only read a document, they can't allow user 2 to edit it.
4. If there's no contradiction between the ACL created by an administrator and the decision made by user 1, access is granted.

Kwabena Ohene-Bonsu

Access Control Models

Role-based access control (RBAC)

- **Main components**
  - User: an individual (with UID) with access to a system
  - Role: a named job function (indicates the level of authority)
  - Permission: equivalent to access rights
  - Session: a mapping between a user and a set of roles to which the user is assigned in the context of a working time
  - Object: a system resource that requires permission to access
  - Operation: any action in the protected network
- **Basic rules**
  - A user can execute an operation only if there is a role assigned to the subject.
  - Identification and authentication are not considered operations.
  - All user activities are carried out through operations.

Access Control Models

Attribute-based access control (ABAC)

- **Attribute – a characteristic of any element in the network. An attribute can define:**
  - **User** characteristics – employee position, department, IP address, clearance level, etc.
  - **Object** characteristics – type, creator, sensitivity, required clearance level, etc.
  - Type of **action** – read, write, edit, copy, paste, etc.
  - **Environment** characteristics – time, day of the week, location, etc.
- **Subject** – any user or resource that can perform actions in the network; a subject is assigned attributes in order to define its clearance level
- **Object** – any data stored in the network; objects are assigned attributes in order to describe and identify them
- **Operation** – any action taken by any subject in the network
- **Policy** – a set of rules allowing or restricting any action in your information retrieval system; rules are “IF/THEN” statements based on attributes of any element (user, resource, environment)

Broken Access Control

- Currently **#1** on OWASP 2021 top 10 list of the most prevalent application security risks
- **94%** of tested applications suffered from this risk with the average incidence rate of **3.81%**
- Over **318K** occurrences of Common Weakness Enumerations (CWEs) from tested applications

Eddie Huang

Broken Access Control

Common broken access control types

- Vertical Privilege Escalation
  - If a non-administrative user can gain access to use or view resource which is only accessible for administrative users.
- Horizontal Privilege Escalation
  - If a user can gain access to resources belonging to another user, instead of their own resources of that type.
- Context-Dependent Privilege Escalation
  - User can do things that should not be possible in context, such as performing tasks out of order, and bypassing the payment step when checking out

Broken Access Control

- Failure to enforce a policy that controls users' actions outside of their intended permissions
- Occurs when application features, use cases, and domain data are not protected properly
- Leads to:
  - Unauthorized information disclosure
  - Modification or destruction of data
  - Performing business functions outside the limits of the user
- Commonly due to:
  - Lack of automated detection
  - Lack of effective functional testing by application developer

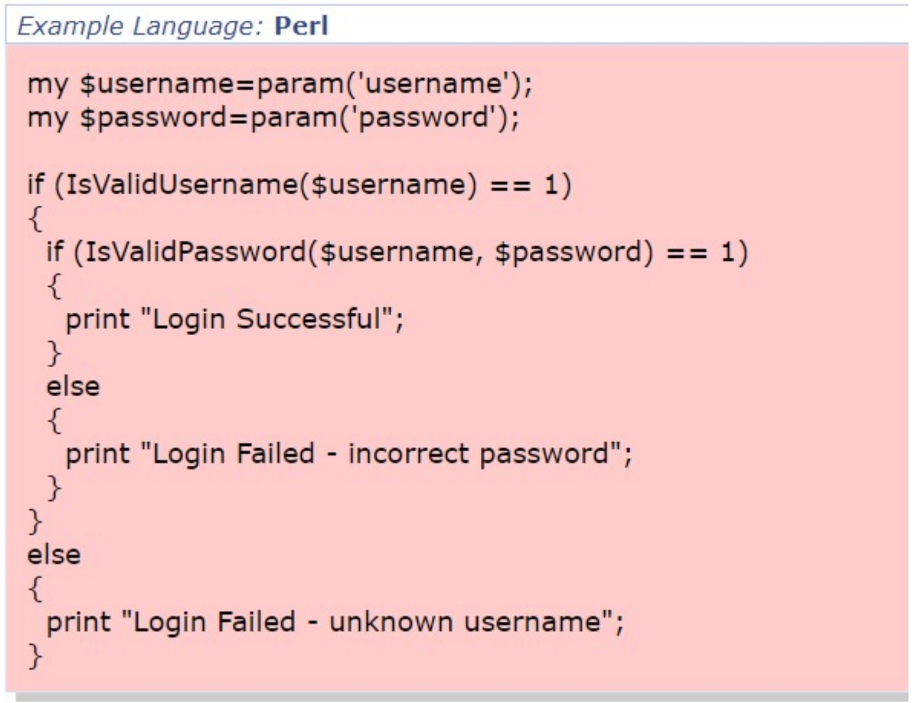
Consequences

Sometimes called Information Leak or Information Disclosure, this is a common weakness in the Broken Access Control category. The Data Protection Act (2018) defines sensitive information and states that anyone responsible for personal data must ensure it is handled in a way that ensures appropriate security. Hence, guarding against this weakness is crucial from a legal standpoint. [UK Government]

The commonality of Broken Access Control attacks in this instance "comes from their ability to bypass legacy security scanning tools, including dynamic application security testing (DAST) that takes a deeper understanding of how data works within an application. [Contrast Security]

Rufus Stretton-Pow

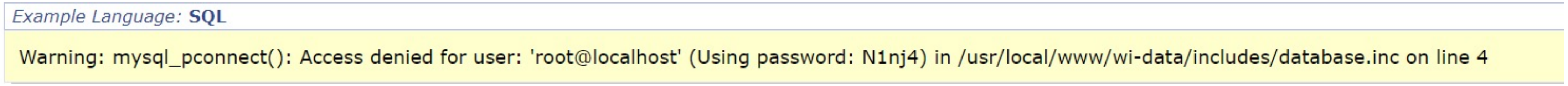
General examples


Example Language: **Perl**

```
my $username=param('username');
my $password=param('password');

if (IsValidUsername($username) == 1)
{
  if (IsValidPassword($username, $password) == 1)
  {
    print "Login Successful";
  }
  else
  {
    print "Login Failed - incorrect password";
  }
}
else
{
  print "Login Failed - unknown username";
}
```

Exposures can happen when "code **explicitly inserts** sensitive information into resources or messages that are intentionally made accessible to unauthorized actors", or "**indirectly inserts** the sensitive information into resources", or "manages resources that intentionally contain sensitive information, but the resources are **unintentionally made accessible** to unauthorized actors". [CWE]


Example Language: **SQL**

```
Warning: mysql_pconnect(): Access denied for user: 'root@localhost' (Using password: N1nj4) in /usr/local/www/wi-data/includes/database.inc on line 4
```

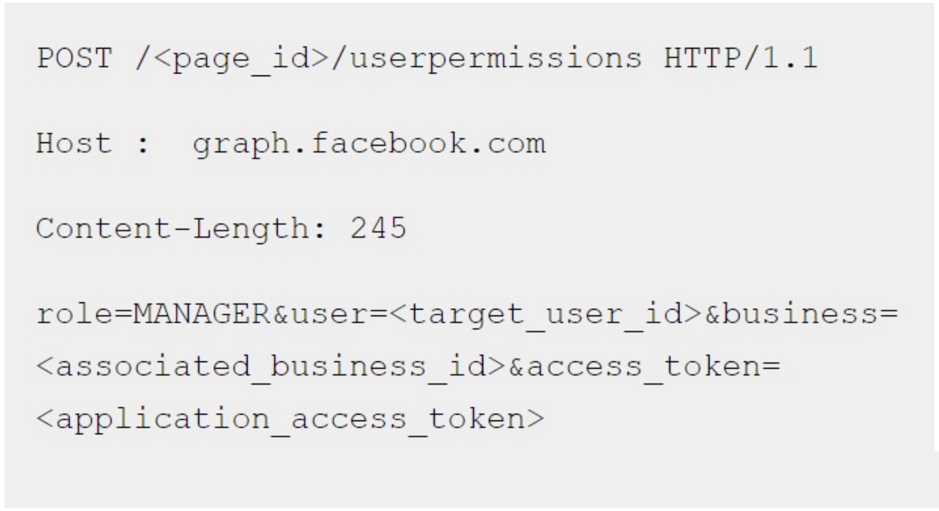
Case Study – Sage Group Data Breach, 2016

The Sage Group, an international Enterprise Software company, were victim of a Data Breach in 2016. An employee had acquired unauthorised access to the personal data of hundreds of employees, and customer data was subsequently compromised. Sage underwent both reputational damage and a 4.3% drop in shares. [Doh]

Hyper socket Software believe the breach was a result of Sage not following the principle of least privilege access. [Hyp]

Case Study – Facebook vulnerability, 2015 [Mut]

Security researcher Laxman Muthiya discovered an access vulnerability in Facebook business manager endpoint that allows a third-party application with limited permissions to add or modify page admin roles. The following POST request to a vulnerable API endpoint would make the target user an admin of any given page (left). The request on the right would remove a victim.

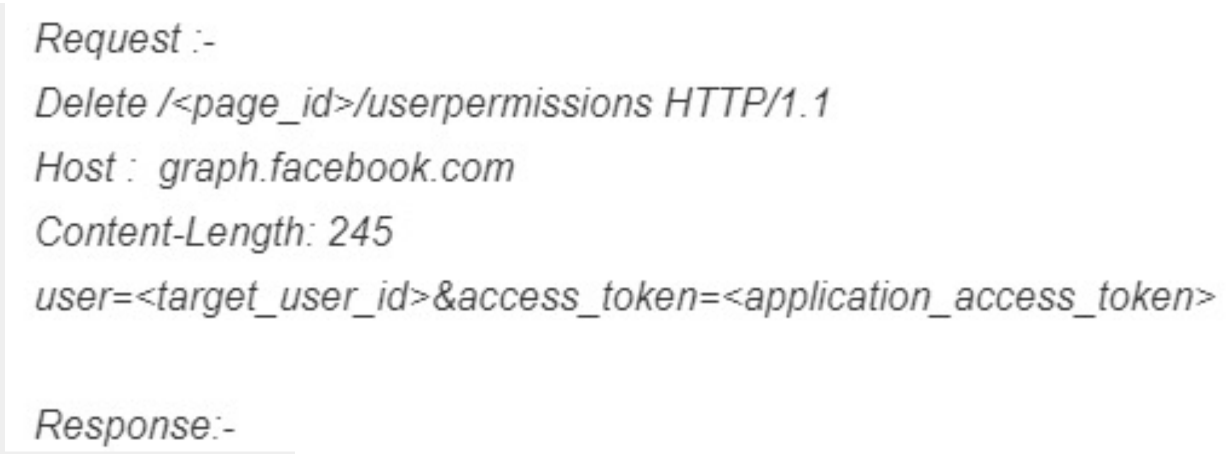


```
POST /<page_id>/userpermissions HTTP/1.1

Host :  graph.facebook.com

Content-Length: 245

role=MANAGER&user=<target_user_id>&business=
<associated_business_id>&access_token=
<application_access_token>
```



```
Request :-
Delete /<page_id>/userpermissions HTTP/1.1
Host :  graph.facebook.com
Content-Length: 245
user=<target_user_id>&access_token=<application_access_token>

Response:-
```

Muthiya reported this vulnerability to Facebook, who paid a bounty and fixed it.

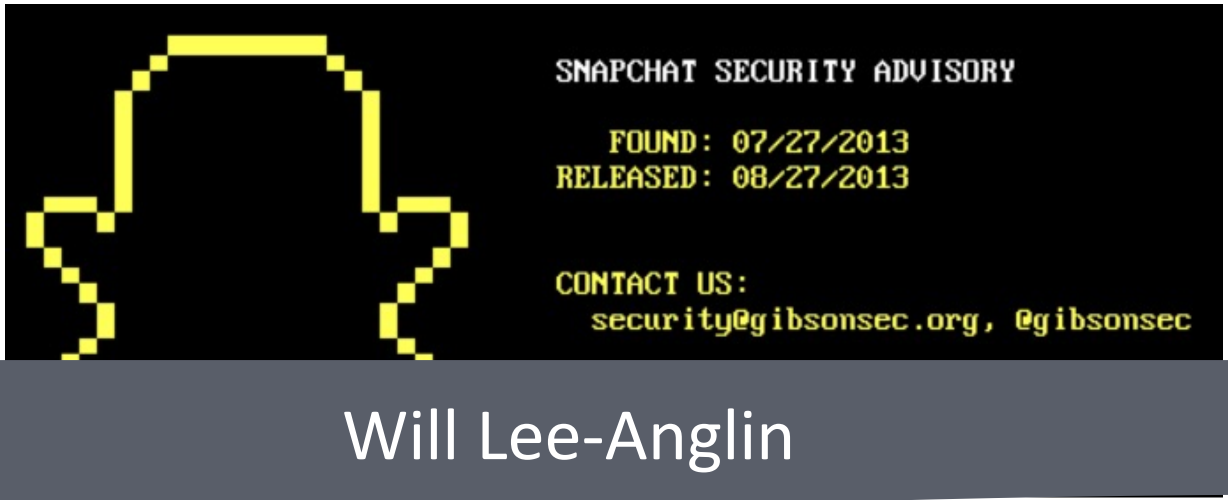
Rufus Stretton-Pow

Case Study – Gibsonsec Snapchat Security Release 2013

Gibson Security discovered multiple vulnerabilities whilst testing Snapchat, a relatively new (at the time) and very popular instant messaging service in mid 2013.

One of the most notable vulnerabilities was a broken access control related vulnerability which allowed any person to create a database of username, alias and phone number combinations.

This was responsibly disclosed by the security company to Snapchat who in turn offered their services to improve Snapchat's security, so such issues don't pop up again in the future.



SNAPCHAT SECURITY ADVISORY

FOUND: 07/27/2013
RELEASED: 08/27/2013

CONTACT US:
security@gibsonsec.org, @gibsonsec

Will Lee-Anglin

Case Study – Gibsonsec – How was this possible?

Snapchat used (still uses) a protocol on top of HTTP.

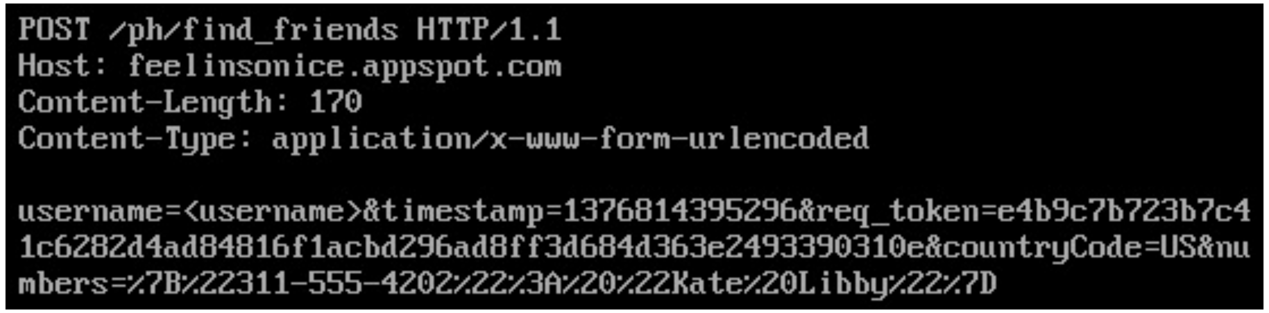
One type of request over this protocol was designed to help users of snapchat find their friends snapchat account through a phone number. This feature would be useful for a user if they had a friend's phone number in their contacts but didn’t have their snapchat account added for example.

This request would result in a reply which contained a list of usernames and aliases for the encrypted phone numbers in the request.

Gibsonsec found that this could be exploited by requesting a single number at a time, easily associating a single phone number with a username and alias (if that phone number had an account associated with it).

There was also no limiting, allowing 75,000 phone number requests at once and being rewarded with thousands of snapchat accounts associated with some of those numbers.

This could be very easily be repeated to generate a database of every snapchat user’s alias, username and phone number.



```
POST /ph/find_friends HTTP/1.1
Host: feelinsonice.appspot.com
Content-Length: 170
Content-Type: application/x-www-form-urlencoded

username=<username>&timestamp=1376814395296&req_token=e4b9c7b723b7c4
1c6282d4ad84816f1acbd296ad8ff3d684d363e2493390310e&countryCode=US&nu
mbers=%7B%22311-555-4202%22%3A%20%22Kate%20Libby%22%7D
```

Preventing Broken Access Control

- Access control is only effective in trusted server-side code or server-less API, where the attacker cannot modify the access control check or metadata.

- Developers and QA staff should include functional access control unit and integration tests.

- SAST and DAST are application security testing methodologies can be used to find security vulnerabilities that can make an application [illegible]ceptible to attack.

Vinay Gupta

Preventing Broken Access Control

- Except for public resources, deny by default.
- Implement access control mechanisms once and re-use them throughout the application, including minimizing Cross-Origin Resource Sharing (CORS) usage.
- Model access controls should enforce record ownership rather than accepting that the user can create, read, update, or delete any record.

Preventing Broken Access Control

- Unique application business limit requirements should be enforced by domain models.
- Disable web server directory listing and ensure file metadata (e.g., .git) and backup files are not present within web roots.
- Log access control failures, alert admins when appropriate (e.g., repeated failures).

Preventing Broken Access Control

- Rate limit API and controller access to minimize the harm from automated attack tooling.
- Stateful session identifiers should be invalidated on the server after logout.
- Stateless JWT(JSON Web Token) tokens should rather be short-lived so that the window of opportunity for an attacker is minimized.
- For longer lived JWT tokens, it's highly recommended to follow the [illegible]th standards to revoke access.

Vinay Gupta

Access Control Design Principles

- Design Access Control Thoroughly Up Front
- Force All Requests to Go Through Access Control Checks
- Deny by Default
- Principle of Least Privilege
- Don’t Hardcode Roles
- Log All Access Control Events

Vishnu Vardhan Reddy Yerragudi

Conclusion

- This is the highest occurring vulnerability in the OWASP TOP 10 2021, and it is quickly becoming a major concern for organizations.
- However, by following the Access Control Design Principles, the Broken Access Control can be prevented from occurring.
- Exploits and attacks are now more prevalent and ubiquitous than ever and ensuring the systems security is an increasingly more complex task.
- Security should be considered as one of the main components during the design and development of an application or system.

Bibliography I

Open Web Application Security Project (OWASP).
The Ten Most Critical Web Application Security Risks (2021).
https://owasp.org/Top10/

Common Weakness Enumeration (CWE).
Weaknesses in OWASP Top Ten (2021).
https://cwe.mitre.org/data/definitions/1344.html

UK Government.
Data Protection Act (2018)
https://www.gov.uk/data-protection

Contrast Security.
Sensitive Data Exposure.
https://www.contrastsecurity.com/knowledge-hub/glossary/sensitive-data-exposure

Doherty.
Real-World Data Breach Examples.
https://www.doherty.co.uk/blog/data-breach-examples-rethink-your-data-strategy

Bibliography II

Hypersocket Software.
Sage data breach.
https://www.prnewswire.com/news-releases/sage-data-breach-highlights-need-for-least-privilege-access-and-two-common-errors-businesses-make-warns-hypersocket-software-590404971.html

Laxman Muthiya.
Hacking Facebook Pages.
https://thezerohack.com/hacking-facebook-pages

Gibsonsec.
Snapchat Security Advisory.
https://gibsonsec.org/snapchat/

Thank you!