

Search blog posts



Posts by Topic

Bug Bounty Management Bugcrowd News Bug Hunter Methodology Company Resources Conferences & Events Cybersecurity News
Guest Blogs Product Updates Program Launches Program Management Program Updates Researcher Resources Researcher Spotlight
Success Stories Thought Leadership Vulnerabilities Vulnerability Disclosure Winner's Circle



WATCH
DEMO



CONTACT
US

Posts By Author

Abigail Nguy Alyssa Habing Ariana Ling Ashish Gupta Barnett Klane Bugcrowd Bugcrowd Product Marketing Casey Ellis Chloe Brown
Daniel Trauner David Baker Grant McCracken Guest Post Jason Haddix JP Villanueva Kaila Pollart Kaushik Srinivas Keith Hoodlet
Lauren Craigie Michael Chung Michelle Dailey Omar Carmical Parag Baxi Pawel Lesniewski Rick Beattie Ryan Black Sam Houston
Shpend Kurtishaj Tim Sandberg

BY [SAM HOUSTON](#) | NOV 9, 2017

HOW-TO: FIND IDOR (INSECURE DIRECT OBJECT REFERENCE) VULNERABILITIES FOR LARGE BOUNTY REWARDS



TIPS FROM TOP HACKERS

How to find IDOR vulnerabilities
(they pay well!)

bugcrowd

The following is a guest blog post from [Mert](#) & [Evren](#), two talented researchers from Turkey. IDOR vulnerabilities are of the higher impact and higher paying vulnerabilities for web bug bounties. This article explores what IDORs are and how to find them.

What is authorization in web/mobile applications?

Web/mobile applications' session management is very important for end users. Session management consists of two important parts, authentication and authorization. The authentication part is the answer of the "WHO AM I?" question and the authorization part is the answer of the "WHAT CAN I DO?" question.

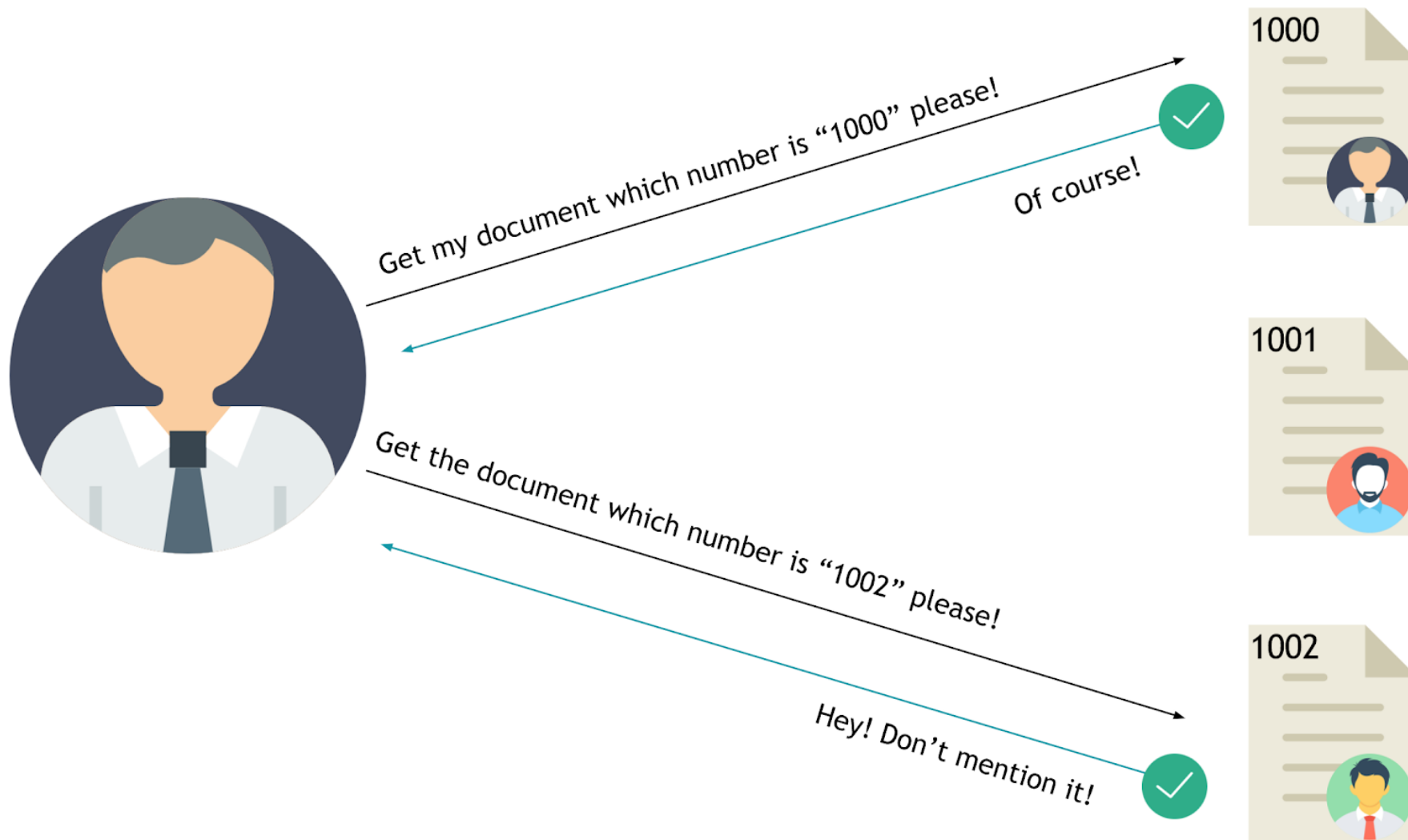
If we explain with the simplest approach, we are observing the authorization stages on Linux operating system file rights specified as write, read and execute. If a user wants to write a file, that the user must be given the "w" authority. If not used correctly, there are some privacy violations.

What is an IDOR vulnerability?

There can be many variables in the application such as "id", "pid", "uid". Although these values are often seen as HTTP parameters, they can be found in headers and cookies. The attacker can access, edit or delete any of other users' objects by changing the values. This vulnerability is called IDOR.

First, it needs to understand the application flow developed by the software developers. All the modules functions and their sub-modules functions need to be understood when the logged-in user into the web/mobile application. It is also important to remember that this vulnerability is as severe as XSS, CSRF in security testing and as a type of vulnerability that is not easily discovered (automatized testing or manual testing).

The IDOR vulnerability is illustrated in the following image between user and server.



In this article, the following topics will be addressed:

- How to find the injection point that IDOR vulnerabilities can be found?
- Some IDOR vulnerability tips that seem to be quite simple and the best experiences we have encountered.
- Precautions to taking into consideration while IDOR vulnerability is testing.
- How to provide basic authorization controls?

Effective & fast IDOR vulnerability test

You can use the browser's secret tab to quickly practically test IDOR vulnerabilities. So, when you use the regular tab as a normal user, you can use the secret tab as an attacker. This will ensure that you don't logout.

You can use Burp Suite's HTTP History tab for checking all of requests. The HTTP History feature that shows all the traffic between the device (browser, phone, tablet) and the application's server. Also, you can use Burp Suite's scope feature for fast testing. Because the scope feature can be useful to make a target list and the scope feature allows showing only relevant data for your testing scope.

For example; the company "Bugcrowd" that we tested and the scope is only given as "bugcrowd.com" on the scope page. In this case, you can add the relevant scope by right-clicking on a request.

Target

Proxy

Spider

Scanner

Intruder

Repeater

Sequencer

Decoder

Com

Intercept

HTTP history

WebSockets history

Options

Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params
154	https://cdn.heapanalytics.com	GET	/js/heap-351079185.js	<input type="checkbox"/>
152	https://api.segment.io	POST	/v1/p	<input checked="" type="checkbox"/>
147	https://dev.visualwebsiteopt...	GET	/j.php?a=39390&u=https%3...	<input checked="" type="checkbox"/>
146	https://assets.locomotive.w...	GET	/_app/sign_in	<input type="checkbox"/>
144	https://cdn.segment.com	GET	/analytics.js/v1/7iC2Ms9O4T...	<input type="checkbox"/>
142	https://assets.bugcrowduse...	GET	/assets/public-ae627fba943...	<input type="checkbox"/>
141	https://cdn.trackjs.com	GET	/releases/current/tracker.js	<input type="checkbox"/>
140	https://assets.locomotive.w...	GET	/sites/57d86805a2f4221de1...	<input type="checkbox"/>
137	https://bugcrowd.com	GET	/user/sign_up	<input type="checkbox"/>
136	https://api-iam.inte		https://bugcrowd.com/user/sign_up	<input checked="" type="checkbox"/>
135	https://js.intercom		Add to scope	<input type="checkbox"/>
131	https://js.intercom		Spider from here	<input type="checkbox"/>
124	https://widget.inter		Do an active scan	<input type="checkbox"/>
123	https://cdn.heapan...			<input type="checkbox"/>

You can edit this added scope value according to the given scope as follows



Target Scope



Define the in-scope targets for your current work. This configuration affects the behavior of tool context menus in the site map to include or exclude URL paths.

Include in scope

Add	Enabled	Protocol	Host / IP range	Port	File
	<input checked="" type="checkbox"/>	HTTPS	^bugcrowd\.com\$	^443\$	^.*
Edit					

Re

Pas

Lo

Exclud

Edit URL to include in scope

?

Specify a regular expression to match each URL component, or leave blank to match any item. An IP range can be specified instead of a hostname.

Protocol:

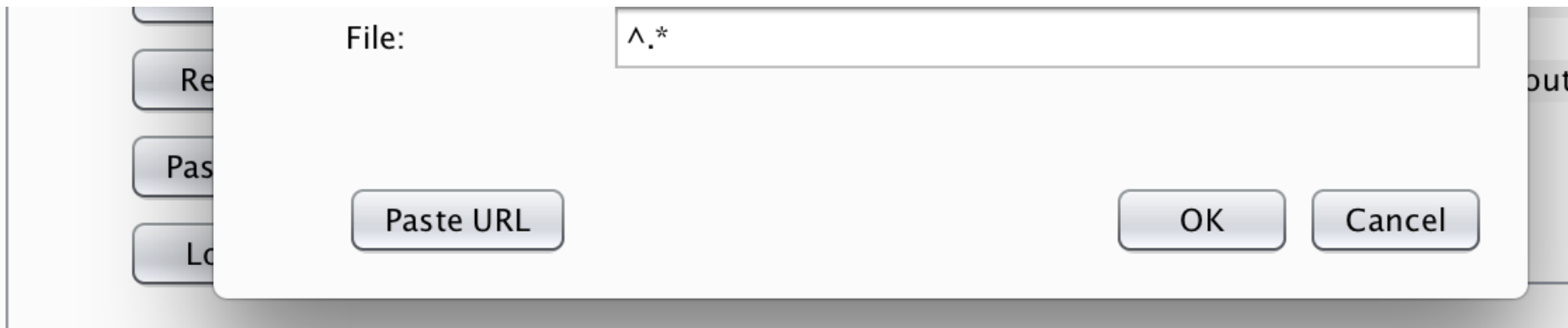
HTTPS

Host or IP range:



^bugcrowd\.com\$|

Port:

^443\$



Finally, you should do the following filtering in the HTTP History tab by selecting “Show only in-scope items”.

Target	Proxy	Spider	Scanner	Intruder	Repeater	Sequencer	Decod
Intercept	HTTP history	WebSockets history	Options				
Filter: Hiding out of scope items; hiding CSS, image and general binary content							
<div><div></div><div><div>Filter by request type</div><div><input checked="" type="checkbox"/> Show only in-scope items <input type="checkbox"/> Hide items without responses <input type="checkbox"/> Show only parameterized requests</div></div><div><div>Filter by MIME type</div><div><div><input checked="" type="checkbox"/> HTML <input checked="" type="checkbox"/> Script <input checked="" type="checkbox"/> XML <input type="checkbox"/> CSS</div><div><input checked="" type="checkbox"/> Other text <input type="checkbox"/> Images <input checked="" type="checkbox"/> Flash <input type="checkbox"/> Other bina</div></div></div></div>							

These will help you to understand roles like readonly, normal, super etc in the application better.

Capture all requests!

When IDOR vulnerability testing, basically, you need to perform make all the requests that the web/mobile application should be create. Because if you changed something in the application, can create other requests using this case. If you have all API requests of the application like WSDL file, Swagger page etc. and it's working regularly, you're in luck! You can use this and it'll give you convenience for IDOR testing.

An example is encountered in a private program. Credit cards are added when made the purchase in the mobile application. After the requests are tested, it can be thought that there is not any vulnerability. But when a made second purchase, credit card selection screen is seen and the IDOR vulnerability is at this point. When selecting a credit card on here, application will send the credit card id to server in a request and this request did provide to access other users' credit cards data changing the credit card id.

In another private program, the web application included an in-app messaging system. The user could send messages to other users and add other users to own messages. When the user tries to access one of own messages, a request went to “/messages/5955” and own message id seems to be “5955”. Likewise, when trying to access another user's message by making a request to “/messages/5955”, the message was not accessed. When the user wanted to add another user to own message, arises a request like the one below.

```
POST /messages/5955/invite HTTP/1.1Host: example.comUser-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:52.0) Gecko/20100101 Firefox/52.0Accept: */*X-Requested-With: XMLHttpRequestCookie: my_cookiesConnection: closeuser=testaccount2
```

And when this request is examined, the user can add itself to other users' messages and access all messages.

In addition, roles in the application must be well understood in order to identify IDOR vulnerability. If you know what that a role should do or should not do, it will be very useful during the phase of weakness

detection. So firstly, you should understand the application deeply!

How to find injection points?

As before say, you can find many requests for IDOR vulnerability testing using all features of application. When API endpoints are not provided in IDOR vulnerability tests, .html source code or .js files are useful. These files include interesting things and ajax requests usually. IDOR vulnerability testing can be performed using presented requests in these files. This can be requests made earlier by the application, and possible future requests.

If you are lucky, you can see only the requests that an authorized, admin user should see in javascript files. Because of this, source code and especially javascript files should analyze well.

Also, you can search web application's old version on "archive.org" and you may can find useful requests in old javascript files or you can search requests in search engines using dorks.

In some cases, id values aren't unique like 1, 2, 3, 100, 1000 etc, these id values can be encoded or hashed value. If you face an encoded value, you can test the IDOR vulnerability with decoding the encoded value. If you face a hashed value, you should test whether the hash value is an accessible or predictable value. In another case, you can access hashed value in "Referrer" header, so these scenarios can be replicated.

For example, you can't access the another users' objects but you can find object's hashed id value in the object page's source code, you can find the object's hashed id into an in-app message from victim user (this

will decrease the impact of bug). So you can create 2 test accounts as X and Y, then try to X's hashed id value in Y's requests in Burp History.

If we will touch another topic, some applications' requests may scare you. For example, the SmartSheet's request that contains more than one parameter appears to be too complex.

Request

Raw

Params

Headers

Hex

```
POST /b/home?formName=ajax&formAction=fa_loadShareCount&ss_v=47.1.7 HTTP/1.1
Host: app.smartsheet.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:50.0) Gecko/20100101
Firefox/50.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Smartsheet-nep: formName,formAction,sk,to,param1,param2
Referer: https://app.smartsheet.com/b/home
Content-Length: 272
Cookie: S3S=cbe7488b8bd14124a4c3f421ec84cble; lgc=smrP65sBxZw; spiff=1;
optimizelyEndUserId=oeul481838578331r0.3270162196044816;
optimizelySegments=%7B%22750602483%22%3A%22ff%22%2C%22752322120%22%3A%22none%22%2C%2
2753390924%22%3A%22direct%22%2C%22755750651%22%3A%22false%22%2C%22180750605%22%3A%2
2true%22%2C%225856500238%22%3A%22true%22%7D;
optimizelyBuckets=%7B%227700970687%22%3A%227698791599%22%7D;
_ga=GA1.2.1743107951.1481838592; _dc_gtm_UA-315244-6=1; _gat_UA-315244-6=1;
tinfo=r%3Dhttps%253A%252F%252Fwww.smartsheet.com%252F%26ss_lc%3Den_US%26email%3Dtest
erxxx%2540yopmail.com%26slp%3Dprovconfirm;
_mkto_trk=id:464-ONM-149&token:_mch-smartsheet.com-1481838613395-56394;
__insp_wid=644356321; __insp_slim=1481838614156; __insp_nv=true;
__insp_targlpu=https%3A%2F%2Fwww.smartsheet.com%2Fprovconfirm%3Fss_lc%3Den_US%26emai
l%3Dtesterxxx%40yopmail.com;
```

```
__insp_targetpt=Your%20account%20activation%20e-mail%20is%20on%20its%20way...%20%7C%20SmartSheet; __insp_norec_sess=true; user_id=16993324; plan_type=Trial;
_dc_gtm_UA-315244-7=1
Connection: close

formName=ajax&formAction=fa_shareDashboard&sk=BPDA5WUHRBY2RLH4DBUC3YNC2SFPD5YS&to=68000&parm1=134369648&parm2=%5B%22%5Cu005b%5Cu002216469032%5Cu0022%5Cu005d%22%2C%22%5Cu005b%5Cu005d%22%5D&parm3=10&parm4=%5B%22%20-%20Invitation%20to%20View%22%2C%22%22%2Cfalse%2Ctrue%5D
```

If you wanna find the inject point in this request, you can use Burp Suite's compare tool. You should right-click on the request and choose "Send to Comparer" option. Then you can create the same request for using another object and send to comparer.

When you visit to the comparer tool and click on the "Words" button, you will be presented with a window where the changing points.

Word compare of #1 and #2 (1 difference)

Length: 1,835 ☒ Text ☐ Hex

Length: 1,835 ☒ Text ☐ Hex

```
96044816;  
2%2C%22755750651%22%3A%22false%22%2C%222180750605%22%3A  
=1;_gat_UA-315244-6=1;  
ovconfirm;_mkto_trk=id:464-ONM-149&token:_mch-smartsheet.com-  
%26email%3Dtesterxxx%40yopmail.com;  
_id=16993324;plan_type=Trial;_dc_gtm_UA-315244-7=1  
  
2=%5B%22%5Cu005b%5Cu002216469032%5Cu0022%5Cu005d%22%2C
```

Key: Modified Deleted Added

☒ Sync views

You can use same method for HTTP responses and you can examine their differences.

Interesting cases for IDOR bugs

MANIPULATE THE CREATE REQUESTS

Some applications create an id on client-side and then send the in create request to server. This id value can be number such as “-1”, “0” or anything. The existing id values change with the previously created objects’ id. So you can delete / edit other users’ objects using IDOR vulnerability.

If you do not see a parameters like “id”, “user_id”, “value”, “pid”, “post_id” while creating an object, you should add it and test it yourself. You can find the parameter key name by deleting or editing any object on app.

BLIND IDOR

In another case, you can find an IDOR vulnerability but you may couldn’t realize of that. For example, if you change the object’s informations in app, you’ll get an email that includes the object’s information. So if you try to change another user’s informations of object, you can’t access anything in HTTP response but you can access the informations of object with an email. You can call it “Blind IDOR”. 😊

COMBINE THEM!

IDOR bugs' impacts are changeable and we'll touch on that. In some cases, IDOR vulnerabilities can help you by triggering other vulnerabilities that can not be exploited. If you find an unimportant IDOR vulnerability such as editing users non-public & unimportant filename and you wanna raise the impact of your bug, you can use self-XSS bug. The self-XSS vulnerability that you found while the web application testing is generally out of scope and not rewarded. However, you can combine self-XSS vulnerability with another IDOR vulnerability and you can submit report as "IDOR + Stored XSS". In this way you can achieve a vulnerability of P2 level.

CRITICAL IDORS

IDOR vulnerability allows us to access an account at some time, rather than to edit or delete it. These critical bugs appear in fields such as password reset, password change, account recovery. So firstly, you should double check the link in your email and parameters in it. Then, you can capture the password reset request and examine the parameters with any proxy tool. We have seen many times the "user id" value in these requests and we could takeover to another user's account easily.

At the same time, it's an important thing that's account takeover by header values sent in the request. It is seen that some header values such as "X-User-ID", "X-UID" from the test and debug environments are changed. So that the user can act like any user and was able to account takeover successfully.

HPP BUG

In rare cases, you can test the HPP (HTTP Parameter Pollution) vulnerability for IDOR testing by adding the same parameter one more time in your request. An example of this: <https://www.youtube.com/watch?v=kIVefiDrWUw>

Create valid request

You should ensure that the request sent to the server is correct. If you try to send a user's request with another user, you must ensure that this request's "CSRF-Token" value is valid. So you should put the other user's "CSRF-Token" into the request. Otherwise, you will get an error because the token values do not match. This can mislead you.

Likewise, if your tested request is XHR (XML HTTP Request), you must check the validation of "Content-Type" header parameter in your request. Also, app's requests may have custom headers like "W-User-Id", "X-User-Id", "User-Token" etc. If you want to do a correct and perfect test, you must send all headers that the app uses correctly.

Useful tools

As we mentioned before, you can use Burp Suite features. Also, you can use Burp Suite plugins for IDOR vulnerability testing, such as “Authz”, “AuthMatrix” and “Authorize”.

The Authz plugin provides to see response of requests for another users. So you can send X user’s request to Authz and try to access response of it as Y user. Also, you can add custom header for the testing IDOR vulnerability like “X-CSRF-Token”. You can get it from the BApp Store or in this address.

The AuthMatrix plugin allows you to perform authorization checks by registering cookie values or header values for roles in the application. You can get it from the BApp Store and if you want more information for this nice plugin, [go here](#).

If you have API requests, you can use Wsdler plugin for Burp Suite, SoapUI, Postman etc. You can try all GET, POST, PUT, DELETE, PATCH requests and success and fast API test using the tools.

Impact of IDOR vulnerabilities

IDOR vulnerabilities seems as “VARIES DEPENDING ON IMPACT” in [Bugcrowd VRT](#) because of their impact totally depend your submitted bug.

But we have created a list about IDOR vulnerabilities’ impacts based on our experience as follows.

P1 – Account takeover, Access very important data (such as credit card)

P2 – Change / delete another users' public data, Access private / public important data (such as tickets, invoice, payment information)

P3 – Access / delete / change private data (limited personal info: name, address etc.)

P4 – Access any unimportant data

IDOR vulnerabilities' impact depends on the discretion of the program manager.

How to prevent IDOR vulnerabilities?

First, you should control all normal, ajax and API requests when creating an app. For example, can read-only user write anything in app? Or can non-admin user access and create API token that only created by admin user? So, for the test all of IDOR vulnerabilities, you should think like a hacker.

You can provide permission on your application for all endpoints. If your “privatesection” endpoint includes the API requests such as “/api/privatesection/admins”, “/api/privatesection/console”, “/api/privatesection/tokens”, you can block the endpoint for non-admin users.

Also, to make the attacker's job harder and even sometimes even to prevent it, you can use hash function and use hashed values instead of normal number or strings.

~ Mert Tasci, [@merttasci_](#)

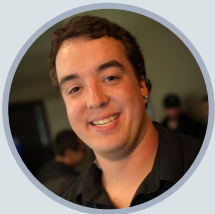
~ Evren Yalcin, [@evrnyalcin](#)

Tags:

Topics:

Researcher Resources

Bug Hunter Methodology



Sam Houston

Senior Community Manager at Bugcrowd. Sam's passionate about working to foster the best researcher community on the web. Prior to joining the security industry Sam worked for Couchsurfing, Electronic Arts, Playfish, and gamerDNA.

[RECENT POSTS](#)



SUBSCRIBE FOR UPDATES

Stay current with the latest security trends from
Bugcrowd

PRODUCTS

Explore The Platform

How it Works

The Bugcrowd Difference

Platform Overview

Integrations

Vulnerability Rating Taxonomy

What We Offer

Bug Bounty

Vulnerability Disclosure

Next Gen Pen Test

Bug Bash

SOLUTIONS

Financial Services

Healthcare

Retail

Automotive

Technology

Government

Security

Secure Marketplaces

CUSTOMERS

RESEARCHERS

Hack With Us

Programs

Bug Bounty List

FAQs

Help Wanted

Learn With Us

Overview

Bugcrowd University

Ambassador Program

Forum

Leaderboard

PROGRAMS

RESOURCES

Resource Library

Webinars

Baker's Dozen

Events

Glossary

FAQs

ABOUT

About Us

Blog

Expertise

Leadership
Press
Careers
Partners
Contact Us

Copyright © 2019 Bugcrowd Bugcrowd Security
Website Terms & Conditions Privacy Policy

