Security (In)Dependence
of Mobile and Internet Banking

Tomáš Rosa

crypto.hyperlink.cz

ICT in Financial Institutions, Prague, Feb 26th – 27th, 2013

Memento

If there is somebody who can imagine this then somebody else may already do it.

If somebody did it then somebody else may repeat it.

Part ONE
The Emerging Decline

SMS-Based

Transaction Authentication Number (TAN)

- Very popular authentication method in contemporary banking systems.
  - http://en.wikipedia.org/wiki/Transaction_authentication_number
- Allows a particular kind of the “must have” two-factor authentication.
  - It uses the out-of-band SMS channel to exercise the second authentication factor.
  - Also called mobile TAN – mTAN.

ISDH Behind mTAN

- Independent Security Domain Hypothesis used to be the main argument behind mTAN security.
  - *For any attacker, in any reasonable computing environment, it is infeasible to compromise both the client's workstation and the SMS channel…*
    - *…(with a probability close to one).*

Cross-Platform Attack (CPA)

- *Any dishonest interoperation of several malware components running on different computing platforms.*
    - Inspired by cross-platform computing.
    - That can be viewed as a toolbox for CPA.

Cross-Platform Infection

- *Any way of CPA components spreading to their respective destinations.*
  - CPI is a necessary condition for CPA.

ISDH Would Imply No CPA

- It was assumed earlier that there is especially…
  - …no feasible cross-platform infection that could hit both the client’s workstation and their mobile phone.

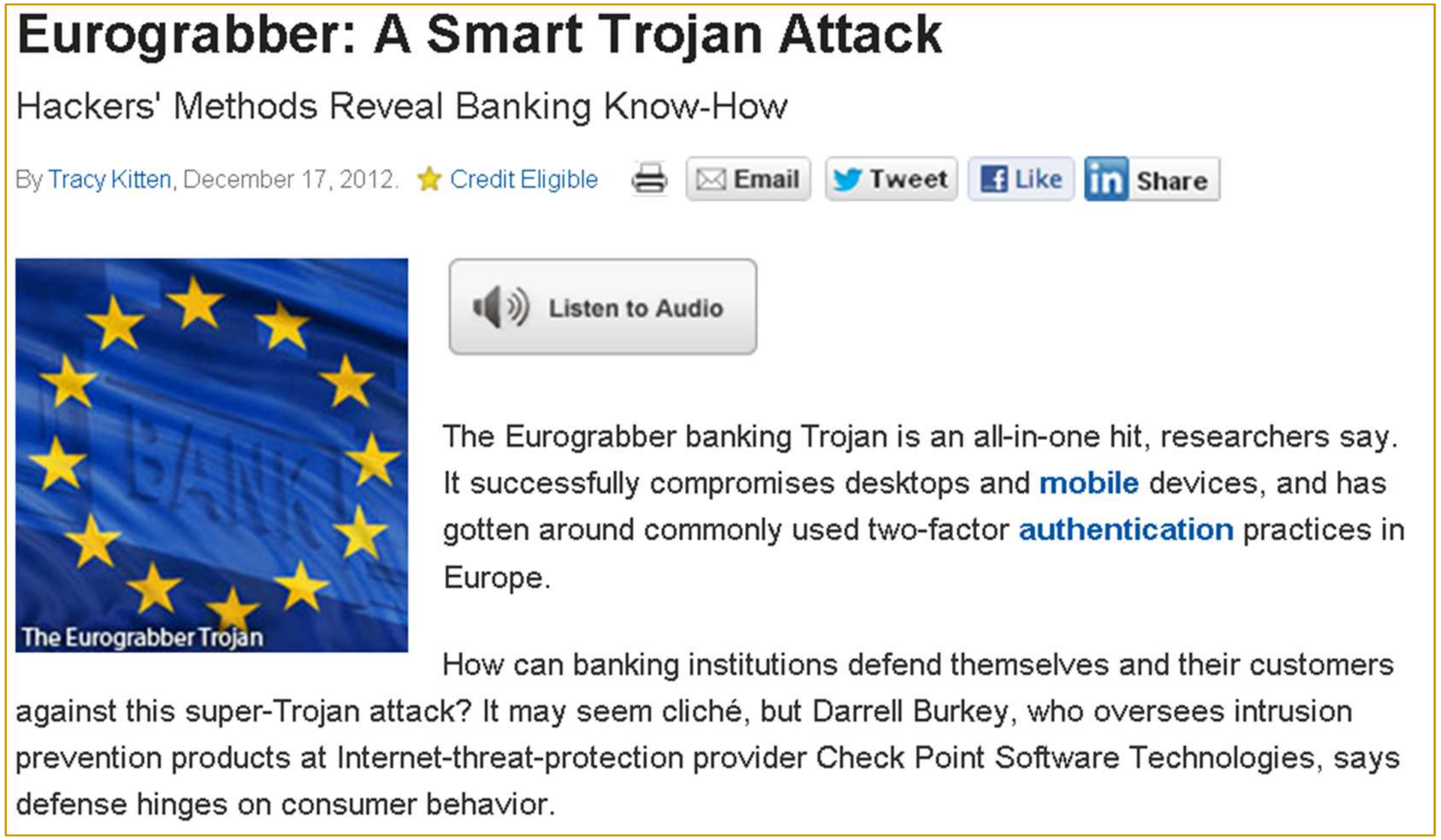
Realistic CPA Does Exist!

- Unfortunately, it was shown experimentally that…
  - …there are several realistic CPAs that can hit both the client's workstation and their mobile phone.
  - In particular, the infection was spreading right through the client's workstation.
    - ZitMo, Spitmo, Gmail link attack, Wi-Fi link attack, USB link attack…

mTAN Becomes Risky

- ISDH is false with a non-negligible probability.
  - In other words, there is non-negligible amount of cases where mTAN security is not guaranteed anymore.
- Furthermore, such attacks usually only get better.
  - We shall be prepared this will get worse.

True Lies



Eurograbber: A Smart Trojan Attack

Hackers' Methods Reveal Banking Know-How

By Tracy Kitten, December 17, 2012.

The Eurograbber banking Trojan is an all-in-one hit, researchers say. It successfully compromises desktops and **mobile** devices, and has gotten around commonly used two-factor **authentication** practices in Europe.

How can banking institutions defend themselves and their customers against this super-Trojan attack? It may seem cliché, but Darrell Burkey, who oversees intrusion prevention products at Internet-threat-protection provider Check Point Software Technologies, says defense hinges on consumer behavior.

http://www.bankinfosecurity.com/eurograbber-smart-trojan-attack-a-5359/op-1

How To Understand

- mTAN risk is <u>not primarily rooted </u>in clients accessing their bank accounts from a web browser on the same mobile platform used for mTAN reception.
    - *<u>The main issue is that cross-platform attacks became feasible anyway!</u>*

Part TWO

Cross-Platform Attacks Examples

No Client Cooperation Required

- Contrary to the pioneering approaches used by ZitMo, Spitmo, and the Eurograbber scenario…
  - … the cross-platform infections reflected hereafter run smoothly with no points of particular cooperation with the client.
  - We can think about generation-2 attacks.

USB Link
Cross-Platform Infection

- Discussed by Stavrou and Wang at BlackHat DC 2011.
  - Exploits USB protocol stack vulnerabilities for infection spreading in both ways (CPI computer $\leftrightarrow$ mobile).
- The original proof-of-concept can be further extended.

Immediate Extensions

- CPI computer $\rightarrow$ iPhone during iTunes synchronization via e.g.:
  - limera1n boot ROM exploit (iPhone 4, 3GS),
    - http://theiphonewiki.com/wiki/Limera1n
  - AFC2 service running on Jailbroken devices.
    - http://theiphonewiki.com/wiki/AFC

Gmail Link Cross-Platform Infection

- Discussed by Rosa in 2011-2012.
  - http://crypto.hyperlink.cz/files/rosa_scforum12_v1.pdf
  - Exploits Android services convergence in the Google portal.
- Based on reverse engineering of certain forensic technique by Cannon and Hoog (2011).
  - Originally named Screen Lock Bypass.
- CPI computer $\rightarrow$ Android device.

Main Idea

- Using stolen Gmail login credentials for an account associated with certain Android device, the attacker can:
  - remotely install any application from Google Play,
  - grant it any user-accessible permission,
  - and make it run!

Requirements

- It requires:
  - Gmail login credentials (captured by the workstation part of the malware)
  - certain malware components exposed on Google Play
- It does NOT require:
  - user cooperation,
  - a-priori knowledge of mobile phone number, etc. (Google Play kindly offers a list…)

Deliberate Fragmentation

- To calm down the automatic malware scan at Google Play…
  - …that part of the malware can be spread around several different applications.
- The attacker is not limited in the number of applications installed during this CPI…
  - Soundcomber proof-of-concept shows nice example of covert channels usage.
    - http://www.cs.indiana.edu/~kapadia/soundcomber-news.html

Wi-Fi Link
Cross-Platform Infection

- Discussed by Dmitrienko et al. at BlackHat AD 2012.
  - Exploits tight interconnection and certain implicit trust of devices in the WLAN of a typical home Wi-Fi.
- CPI computer $\rightarrow$ Android device.
  - This was just a proof-of-concept demo.
  - CPI computer $\rightarrow$ iOS device is also achievable.

Infection Steps

- Let WLAN = {Android, workstation, router, DHCP srv, …}
  1. *Workstation* gets infected.
     - Using conventional techniques.
     - Recall, this assumption was the whole reason for mTAN!
  2. By circumventing *DHCP srv*, the *workstation* is interposed as *default gw* for this WLAN.
     - ARP poisoning would also work…
  3. Wait for Android device joining home Wi-Fi and getting *fake DHCP leas* pointing to the *malicious gw*.
     - Possible variant – do not wait, just force Android to reconnect to the BSS.

Infection Steps (ctd.)

4. Drive-by download Javascript code is injected into web pages visited from Android device.
   - Any GET/POST HTTP transaction can be poisoned.

5. WebKit Use-After-Free vulnerability is exploited to get malicious shellcode running on the embedded Linux core.
   - CVE-2010-1759
   - No ROP, very slight anti-ASLR, just plain ARM code (remember, it was a proof-of-concept setup).
   - Downloads and immediately executes the main malware module.

6. Volume manager deamon (vold) is further exploited to run the main malware module as root.
   - CVE-2011-1823
   - Substitutes system() in place of atoi() in vold GOT.

Infection Steps (ctd.)

7. The main malware module establishes itself in between the radio and application processor Android stack.
   - Besides the others (!), it sees the SMS application-layer traffic.
   - Also, by using low-level configuration files, the malware ensures it survives system restarts.
   - The SMS service is also employed for remote management of the malware module by the attacker.

Lessons Learned

- Remember, we have mostly seen proof-of-concept code.
  - We shall mainly look for general principles.
- Clearly, the coupling of smart devices and client workstations gets tighter and tighter.
  - Apparently, the resulting system cannot withstand cross-platform attacks.
  - Especially, the cross-platform infection became feasible.

New Design Paradigm Required

1. We shall go one step further to have our own code running on the mobile device.
   - It is not only a marketing question.
   - Having a mobile banking application is actually a kind of security countermeasure!
2. We shall admit it is important to keep the “paired” computer safe.
   - We can no longer ignore this issue hoping that the mobile device takes it all!

Part THREE

Here Comes the Smart Device

ATA Scenario

Definition. *Let the After-Theft Attack (ATA) be any attacking scenario that assumes the attacker has unlimited physical access to the user's smart device.*

- Imagine somebody steals your mobile phone…
- Despite being really obvious threat, it is often neglected in many contemporary applications.

- By a robbery, the attacker can even get access to <u>unlocked screen or a synced computer</u>, hence receiving another considerable favor!

Forensic Techniques Lessons

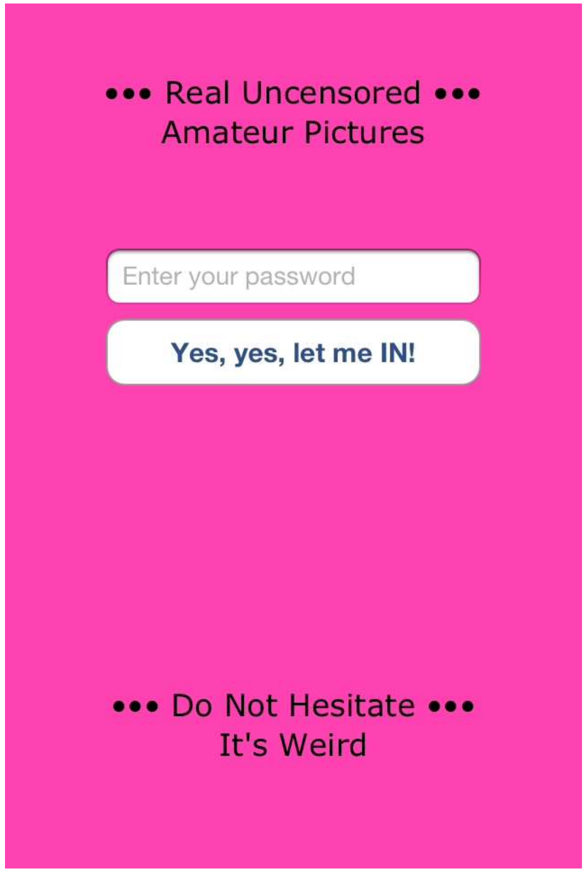
- Hackers conferences are not the only place where to look for an inspiration.
- Forensic experts also publish very interesting results (Breeuwsma, Hoog, Zdziarski, et al.).
  - Actually, they often take hacking techniques and refine them to another level of maturity.
  - The main purpose is to prosecute criminals, of course.
  - But it is just a question of who is holding the gun…
  - Anyway, security experts shall definitely consider looking into forensic publications, at least time to time.

Be Aware of PIN Prints

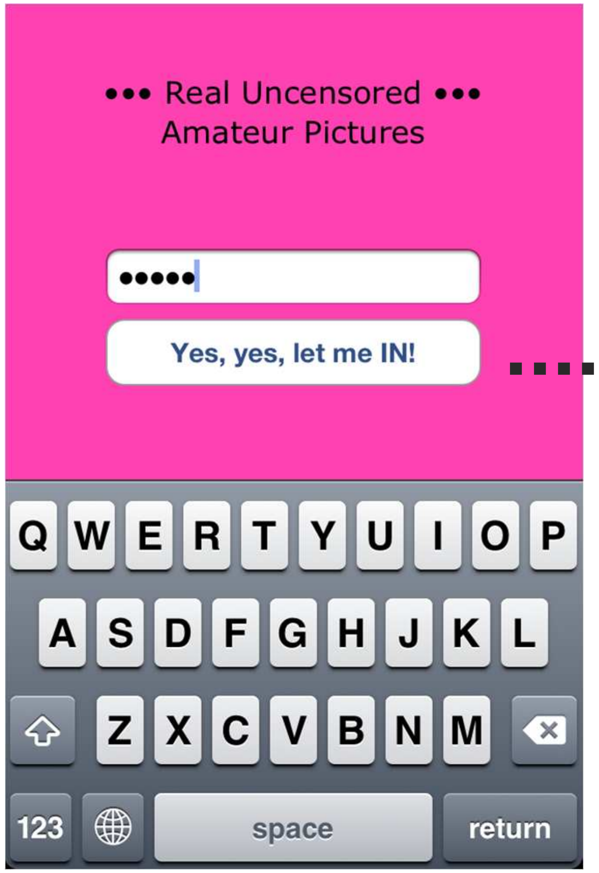
- This can be any direct or indirect function value that:
  - once known to the attacker,
  - can be used for a successful brute force attack on the PIN,
  - under the particular attack scenario.
  - *Rosa: The Decline and Dawn of Two-Factor Authentication on Smart Phones, ISS 2012*
  - *Rosa: Discovering PIN Print in Mobile Applications, Security 2013*
- Principally, the same applies to general passwords, too.
  - However, we can mitigate the risk by enforcing strong password policy here.

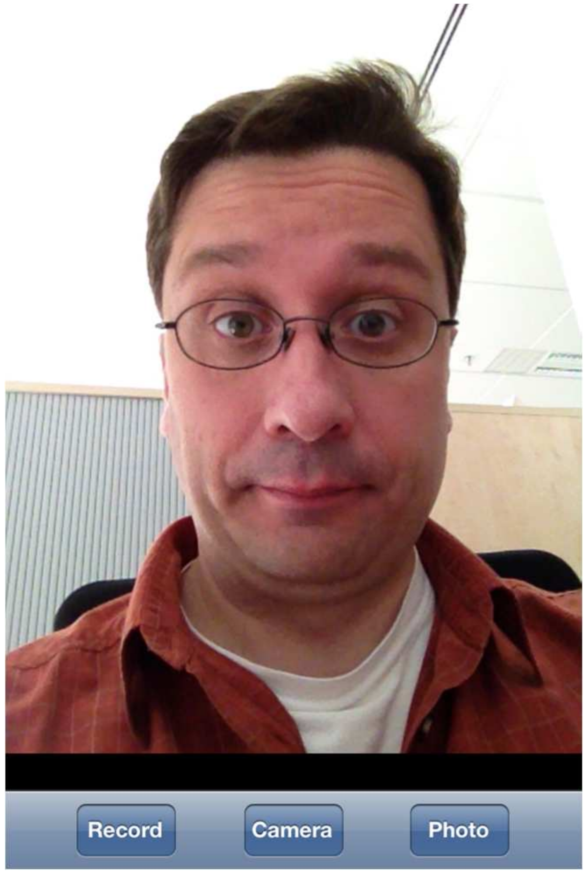
Weird Pictures Demo

- Well, it would not be fair to use real-life applications here.
- We will use a modest iPhone joke that was written especially for this purpose to exhibit all those weaknesses we want to talk about.



Password: “kubrt”

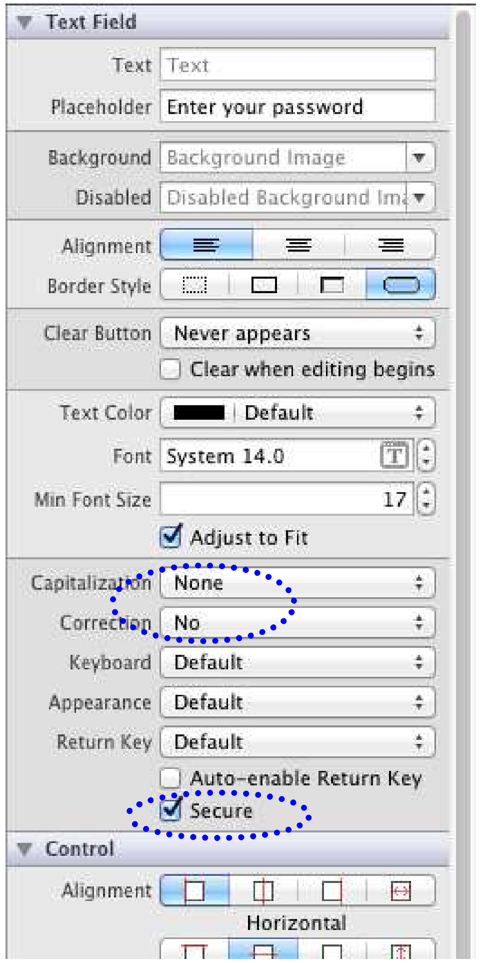




It’s just the front camera in action…

UITextField in Weird Pictures

- We use this control view to let users to type their password.
- Of course, we have marked it “Secure”.
  - But, is it enough?



Text Field

Text	Text
Placeholder	Enter your password
Background	Background Image
Disabled	Disabled Background Ima
Alignment	
Border Style	
Clear Button	Never appears
	☐ Clear when editing begins
Text Color	Default
Font	System 14.0
Min Font Size	17
	☑ Adjust to Fit
Capitalization	None
Correction	No
Keyboard	Default
Appearance	Default
Return Key	Default
	☐ Auto-enable Return Key
	☑ Secure

Control

Alignment	Horizontal

What a Surprise…

- As the user starts typing on the virtual keyboard, we can see:

```
…
Breakpoint 1, 0x324d508a in -[UITextField text] ()
from: 0x3242bb91
0x3242bb91 <-[UITextField _updateAutosizeStyleIfNeeded]+69>…
return: 0x14d750
0x14d750: 0x3f4712c8 <OBJC_CLASS_$___NSCFString>
$2 = (unsigned char *) 0x0

Breakpoint 1, 0x324d508a in -[UITextField text] ()
from: 0x3242bb91
0x3242bb91 <-[UITextField _updateAutosizeStyleIfNeeded]+69>…
return: 0x12f860
0x12f860: 0x3f4712c8 <OBJC_CLASS_$___NSCFString>
$3 = (unsigned char *) 0x35c2c1 "k"
```

...And It Continues...

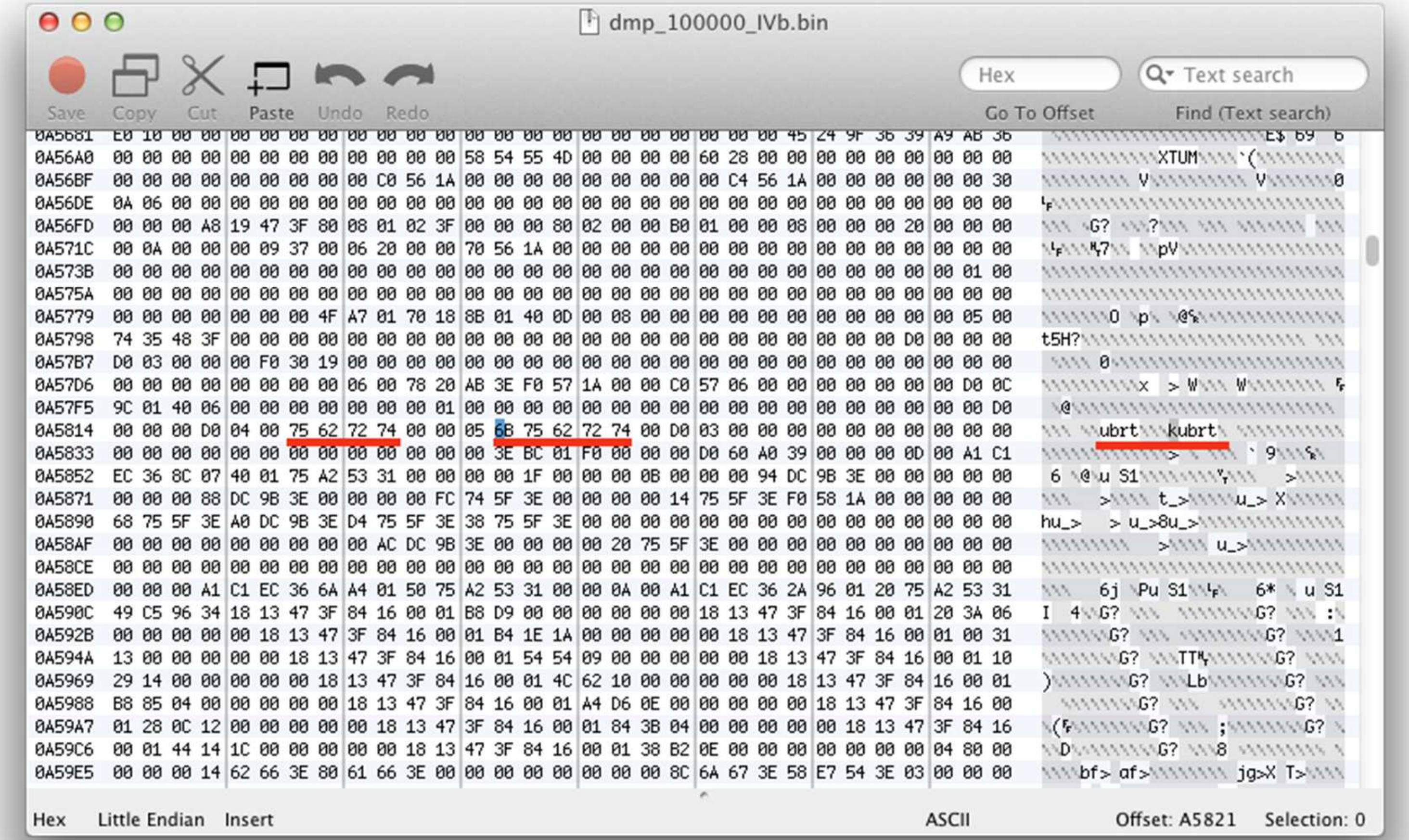
```
Breakpoint 1, 0x324d508a in -[UITextField text] ()
from: 0x3242bb91
0x3242bb91 <-[UITextField _updateAutosizeStyleIfNeeded]+69>:       movw          r6, #5276     ; 0x149c
return: 0x1483f0
0x1483f0:     0x3f4712c8 <OBJC_CLASS_$___NSCFString>
$4 = (unsigned char *) 0x159ae1 "ku"

Breakpoint 1, 0x324d508a in -[UITextField text] ()
from: 0x3242bb91
0x3242bb91 <-[UITextField _updateAutosizeStyleIfNeeded]+69>:       movw          r6, #5276     ; 0x149c
return: 0x3179f0
0x3179f0:     0x3f4712c8 <OBJC_CLASS_$___NSCFString>
$5 = (unsigned char *) 0x35eed1 "kub"

Breakpoint 1, 0x324d508a in -[UITextField text] ()
from: 0x3242bb91
0x3242bb91 <-[UITextField _updateAutosizeStyleIfNeeded]+69>:       movw          r6, #5276     ; 0x149c
return: 0x15a3d0
0x15a3d0:     0x3f4712c8 <OBJC_CLASS_$___NSCFString>
$6 = (unsigned char *) 0x13dca1 "kubr"

Breakpoint 1, 0x324d508a in -[UITextField text] ()
from: 0x3242bb91
0x3242bb91 <-[UITextField _updateAutosizeStyleIfNeeded]+69>:       movw          r6, #5276     ; 0x149c
return: 0x113e40
0x113e40:     0x3f4712c8 <OBJC_CLASS_$___NSCFString>
$7 = (unsigned char *) 0x15a3d1 "kubrt"
```

Illustration of Heap Pollution



```
0A5681  E0 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 45 24 9F 36 39 A9 AB 36
0A56A0  00 00 00 00 00 00 00 00 00 00 00 00 58 54 55 4D 00 00 00 00 60 28 00 00 00 00 00 00 00 00 00
0A56BF  00 00 00 00 00 00 00 00 00 C0 56 1A 00 00 00 00 00 00 00 00 00 C4 56 1A 00 00 00 00 00 00 30
0A56DE  0A 06 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0A56FD  00 00 00 A8 19 47 3F 80 08 01 02 3F 00 00 00 80 02 00 00 B0 01 00 00 08 00 00 00 20 00 00 00
0A571C  00 0A 00 00 00 09 37 00 06 20 00 00 70 56 1A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0A573B  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00
0A575A  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0A5779  00 00 00 00 00 00 00 4F A7 01 70 18 8B 01 40 0D 00 08 00 00 00 00 00 00 00 00 00 00 00 05 00
0A5798  74 35 48 3F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 D0 00 00 00
0A57B7  D0 03 00 00 00 F0 30 19 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0A57D6  00 00 00 00 00 00 00 00 06 00 78 20 AB 3E F0 57 1A 00 00 C0 57 06 00 00 00 00 00 00 00 D0 0C
0A57F5  9C 01 40 06 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 D0
0A5814  00 00 00 D0 04 00 75 62 72 74 00 00 05 6B 75 62 72 74 00 D0 03 00 00 00 00 00 00 00 00 00 00
0A5833  00 00 00 00 00 00 00 00 00 00 00 00 00 3E BC 01 F0 00 00 00 D0 60 A0 39 00 00 00 0D 00 A1 C1
0A5852  EC 36 8C 07 40 01 75 A2 53 31 00 00 00 00 1F 00 00 00 0B 00 00 00 94 DC 9B 3E 00 00 00 00 00
0A5871  00 00 00 88 DC 9B 3E 00 00 00 00 FC 74 5F 3E 00 00 00 00 14 75 5F 3E F0 58 1A 00 00 00 00 00
0A5890  68 75 5F 3E A0 DC 9B 3E D4 75 5F 3E 38 75 5F 3E 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0A58AF  00 00 00 00 00 00 00 00 00 AC DC 9B 3E 00 00 00 00 20 75 5F 3E 00 00 00 00 00 00 00 00 00 00
0A58CE  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0A58ED  00 00 00 A1 C1 EC 36 6A A4 01 50 75 A2 53 31 00 00 0A 00 A1 C1 EC 36 2A 96 01 20 75 A2 53 31
0A590C  49 C5 96 34 18 13 47 3F 84 16 00 01 B8 D9 00 00 00 00 00 00 18 13 47 3F 84 16 00 01 20 3A 06
0A592B  00 00 00 00 00 18 13 47 3F 84 16 00 01 B4 1E 1A 00 00 00 00 00 18 13 47 3F 84 16 00 01 00 31
0A594A  13 00 00 00 00 00 18 13 47 3F 84 16 00 01 54 54 09 00 00 00 00 00 18 13 47 3F 84 16 00 01 10
0A5969  29 14 00 00 00 00 00 18 13 47 3F 84 16 00 01 4C 62 10 00 00 00 00 00 18 13 47 3F 84 16 00 01
0A5988  B8 85 04 00 00 00 00 00 18 13 47 3F 84 16 00 01 A4 D6 0E 00 00 00 00 00 18 13 47 3F 84 16 00
0A59A7  01 28 0C 12 00 00 00 00 00 18 13 47 3F 84 16 00 01 84 3B 04 00 00 00 00 00 18 13 47 3F 84 16
0A59C6  00 01 44 14 1C 00 00 00 00 00 18 13 47 3F 84 16 00 01 38 B2 0E 00 00 00 00 00 00 00 04 80 00
0A59E5  00 00 00 14 62 66 3E 80 61 66 3E 00 00 00 00 00 00 00 00 8C 6A 67 3E 58 E7 54 3E 03 00 00 00
```

ubrt kubrt

Hex   Little Endian   Insert   ASCII   Offset: A5821   Selection: 0

OFA Scenario

Definition. *Let the On-the-Fly Attack (OFA) be any attacking scenario that assumes the attacker is able to launch their privileged code running on the user's smart device transparently during the time the legitimate user performs the authentication procedure.*

- Note that this does not strictly call for having the root account access.
- It is more important to bypass the application sandbox barrier.
  - When we can do that then the "mobile" account on iOS or the respective application UID on Android is usually far enough for the OFA attack.

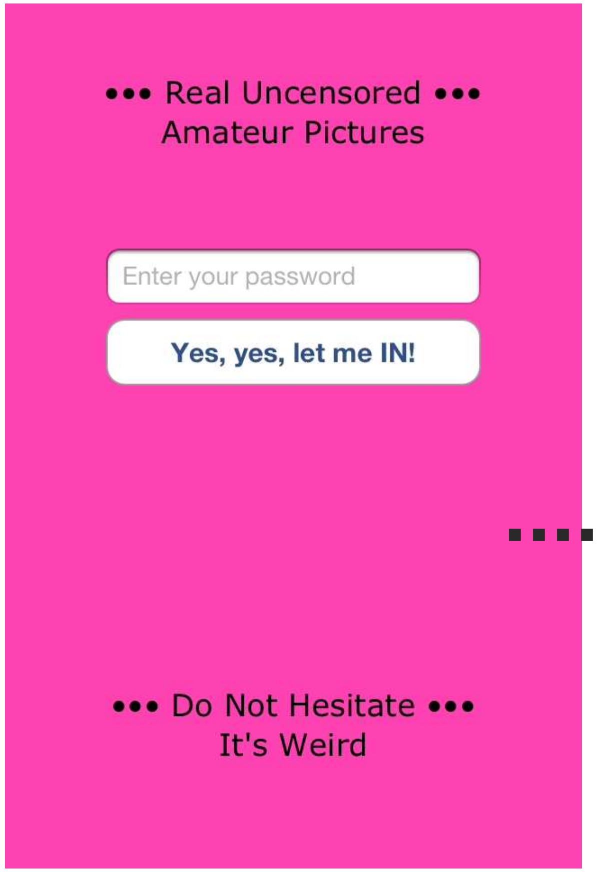
Cycript

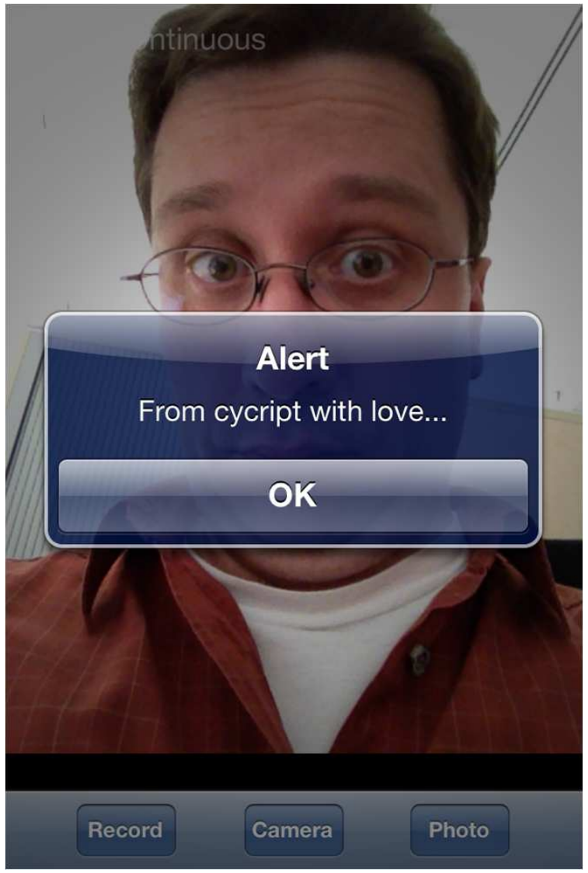
- Delicate combination of JavaScript and Objective-C interpreter running on iOS.
  - Provides REPL (Read-Eval-Print Loop) interface.
- It can attach to an already running process and start commanding its Objective-C runtime.
  - It uses direct process debugging API, now, so it relies on a jailbreak to grant the appropriate *entitlements*.
  - Another injection vector went through MobileSubstrate.
  - Cydia users love installing MobileSubstrate patches for existing applications – they call them *tweaks*.
- Its original purpose probably was not application hacking (in security sense).
  - Anyway, it is an excellent tool for vulnerability research and demonstration.

Consider This (hack1.cy)

```
function AppVC() {
   var window = [UIApp keyWindow];
   this.viewController = [window
   rootViewController];
}
AppVC.prototype.unlock =
   function(animated/*opt*/) {
   [this.viewController
   dismissModalViewControllerAnimated:animated];
   cocoAlert("From cycript with love...");
}
var ac = new AppVC();
ac.unlock();
```

$ cycript -p WeirdPictures hack1.cy





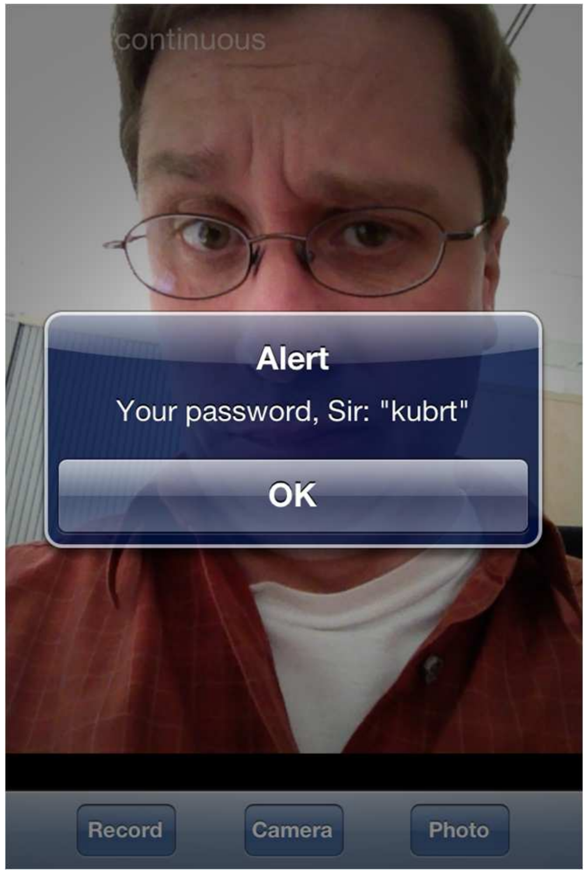
Consider Yet This (hack2.cy)

```
function LoginVC() {
   this.viewController = [WPLoginViewController
   getDefault];
}
LoginVC.prototype.showPwd = function() {
   var pwd = [[this.viewController passwordField] text];
   if (pwd == null)
      cocoAlert("Sorry Sir.");
   else
      cocoAlert("Your password, Sir: \"" +
   pwd.toString() + "\"");
}
var lc = new LoginVC();
lc.showPwd();
```

$ cycript -p WeirdPictures hack2.cy

- We shall consider using one-way derivatives, if we *really* need to keep user secrets in memory for some purpose.
  - Furthermore, it is wise not to expose anything like

```
-(id)passwordField !
```



Alert

Your password, Sir: "kubrt"

Part FOUR

Looking Around

Mobile Payment Application (MPA)

- Runs on the Secure Element (SE).
  - That means on a SIM or a comparable IC.
- Performs client transactions via the EMV contactless protocol.
  - Through the NFC controller, MPA appears as a regular EMV contactless payment card to the terminal.
- The main security focus is usually here.
  - However, MPA has to rely on the Mobile User Application in some cases.

Mobile User Application (MUA)

- Runs on the smart phone application processor.
  - That means under iOS, Android, etc.
- Should mainly provide user interface and network connectivity for MPA.
- Needs to be a trusted code anyway.
  - For instance, it manages entering the PIN (passcode) for MPA.
  - Furthermore, it displays the card details for e.g. internet transactions.

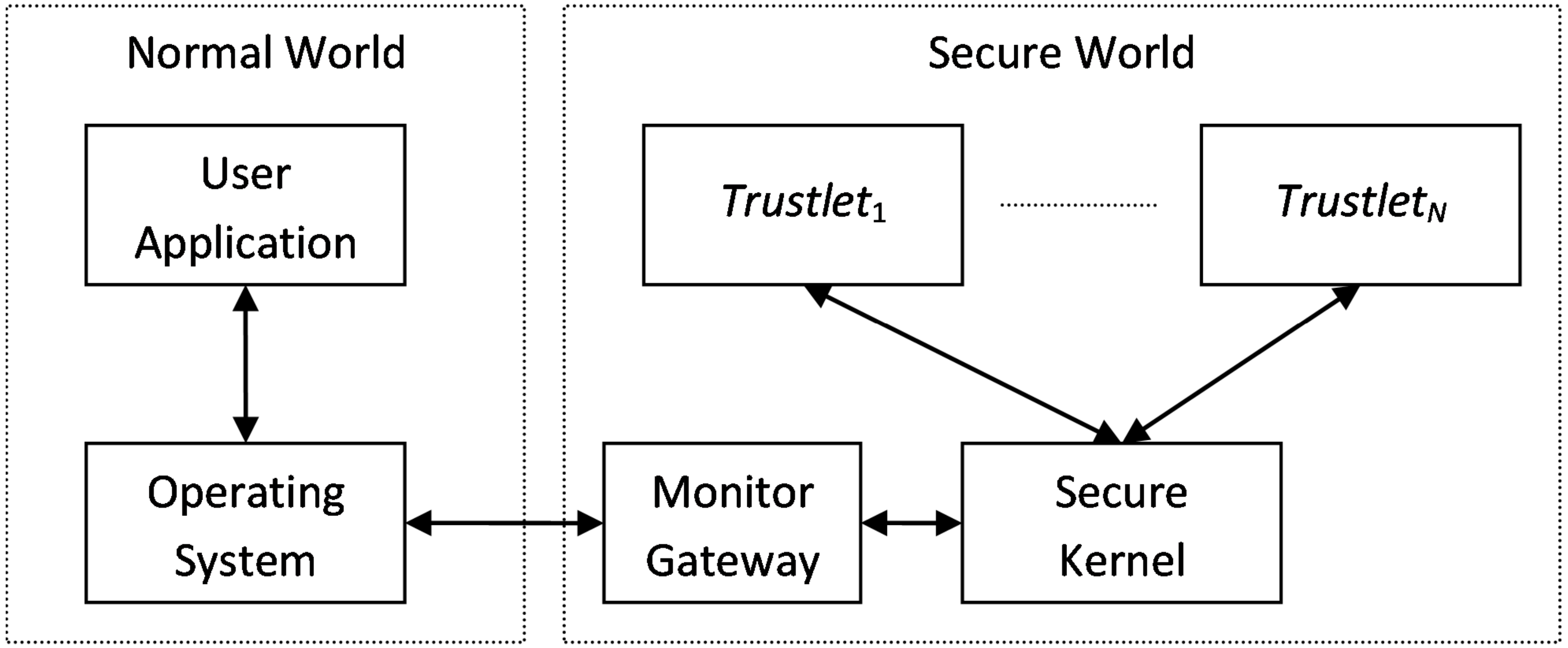
Mobile Cards Wallet (MCW)

- Another smart phone application
  - With possible enhancement on the SE side.
- Solves the problem of having multiple contactless cards “loaded” on the same phone.
- So, it should be independent on the particular bank.
- However, it shall be independent on the particular mobile network providers as well.
  - The smart phone OS is the right place!
  - Apple’s Passbook may serve for an illustration.

TrustZone Basics

- Ready to use HW feature of Cortex-A and higher ARM processors.
  - Offers virtual processor core(s) dedicated to security-critical operations like PIN entry.
- Can reliably defeat OFA threat.
  - Secure environment for OTP/mTAN computation.
- Unfortunately, there is no usable universal operating system support, yet.
  - It is either still unclear on how the procedure of “trustlet” certification would eventually look like.

TrustZone Illustration



ARM Security Technology - Building a Secure System using TrustZone Technology, whitepaper, ARM Limited, 2009

Conclusion

- We shall actively care about smart mobile devices security, now.
  - Not only because of our own mobile applications.
  - We need to secure this platform to have a robust mTAN for online banking anyway!
- Mobile application is not a luxury, it is rather a countermeasure…
  - Having our own code for mTAN support is highly welcome.
- Still not perfect, but sufficient and dynamic enough.
  - Be prepared for future technologies like TrustZone and external “smart tokens”.

Thank You For Attention

Tomáš Rosa, Ph.D.

http://crypto.hyperlink.cz