Stan bezpieczeństwa mobilnych aplikacji bankowych w Polsce

LISTOPAD 2016

PGS SOFTWARE

WSTĘP

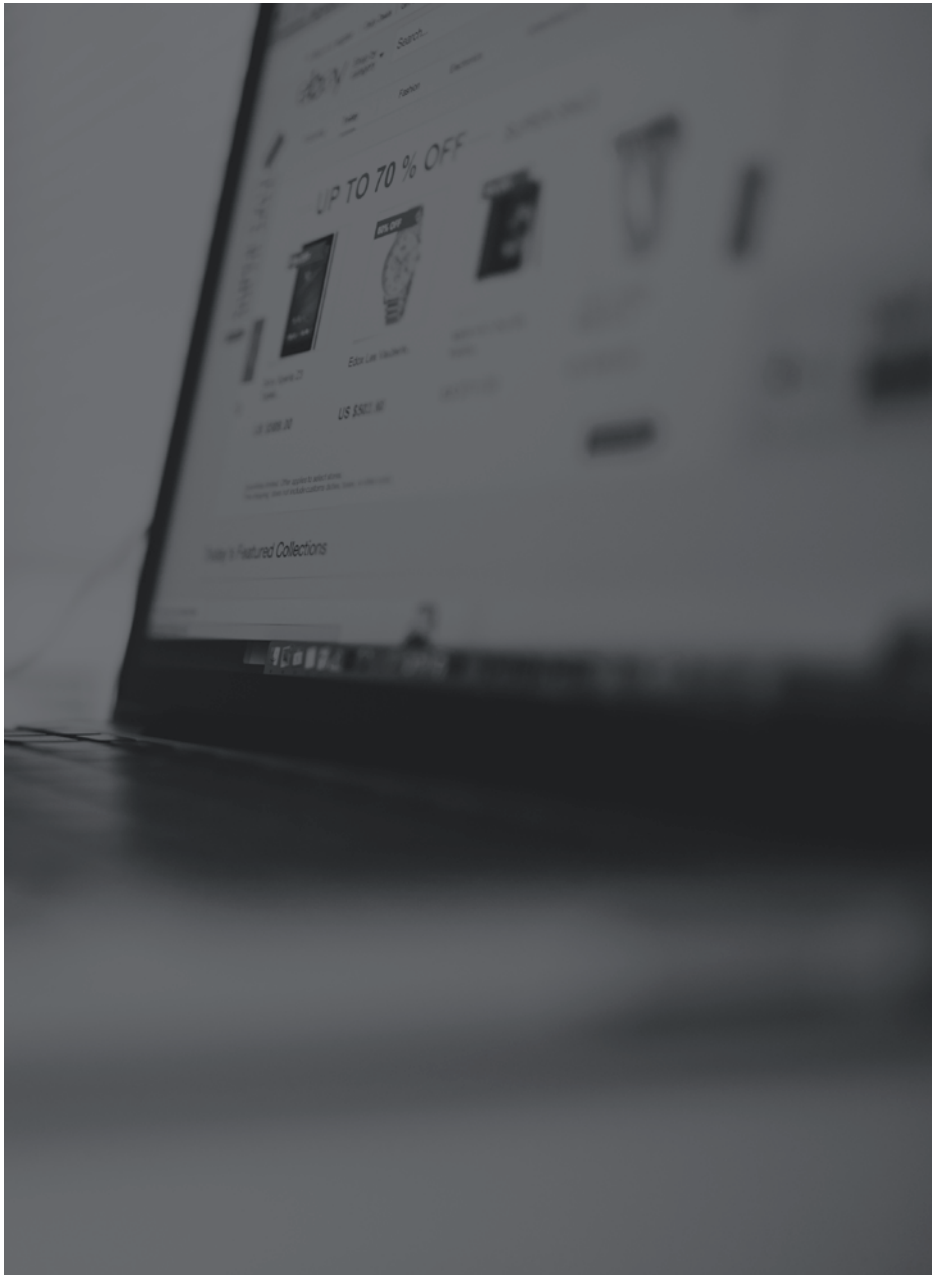
Pytanie „czy mobilna aplikacja bankowa jest bezpieczna" można rozumieć wielorako. Najbardziej oczywista interpretacja dotyczy bezpieczeństwa środków finansowych, jednak po chwili zastanowienia uwzględnimy też np. bezpieczeństwo danych osobowych czy historii transakcji.

W ramach niniejszego raportu przyjrzeliśmy się dziewiętnastu aplikacjom mobilnym dla systemu Android, opublikowanym przez banki detaliczne działające na terenie Polski. W każdym banku założyliśmy najprostszy rachunek, aktywowaliśmy bankowość elektroniczną oraz mobilną, zainstalowaliśmy odpowiednią aplikację i przeprowadziliśmy kilka podstawowych operacji. Jednocześnie staraliśmy się podsłuchać transmisję danych, zapoznawaliśmy się z budową aplikacji, informacjami przez nią składowanymi, ocenialiśmy dobre i złe praktyki, które miały miejsce podczas tworzenia opisywanych programów.

Trzeba podkreślić, że nie tworzymy rankingu bezpieczeństwa i nie wskazujemy, że aplikacja A jest lepsza od aplikacji B. Dlaczego? Każdej pozycji poświęciliśmy raptem kilka godzin – prawie na pewno przeoczyliśmy więc istotną część usterek i podatności. Rzetelne testy penetracyjne wymagałyby wielokrotnie więcej czasu.

Staraliśmy się, aby publikacja była zrozumiała dla przeciętnego użytkownika smartfona. Trudniejsze pojęcia zostały wyjaśnione w ramkach.

Z góry przepraszamy niektóre banki za odstępstwa od używania pełnego brzmienia nazw i marek. Po serii fuzji i przejęć bywają one dość długie i niewygodne w użyciu, więc pozwoliliśmy sobie na skorzystanie z nazw potocznych.



I. BADANIE APLIKACJI

Wnioski

Jakość i bezpieczeństwo wielu przebadanych aplikacji mobilnych pozostawiają bardzo wiele do życzenia. W naszych krótkich testach udało się m.in. doprowadzić do przejęcia sesji zalogowanego użytkownika, wycieku danych osobowych i przejęcia fragmentu maskowanego hasła dostępu. W logach systemowych odnaleźliśmy identyfikatory logowania, tokeny sesji czy dane informujące o liczbie usług określonego typu.

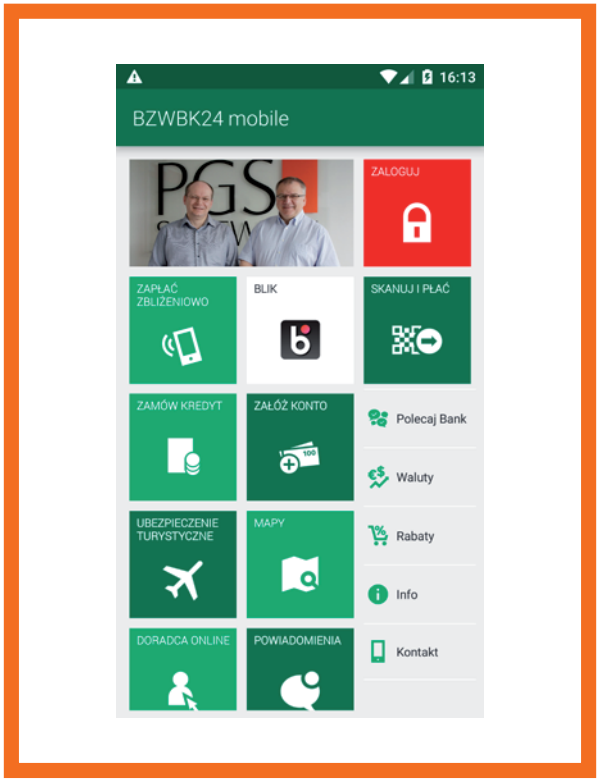
Podsłuchaliśmy transmisję danych wielu aplikacji. W kilku przypadkach wymiana danych miała miejsce otwartym tekstem, przez co może zostać zmodyfikowana w dowolnym miejscu między nadawcą a odbiorcą. Innym razem aplikacjom nie przeszkadzał „lewy" certyfikat bezpieczeństwa, podstawiony celowo i pozwalający na rozszyfrowanie przesyłanych danych.

Spostrzegliśmy, że w wielu przypadkach aplikacje wysyłane do sklepu nie podlegają żadnej kontroli jakości – jak bowiem uwierzyć, że testy przeszła aplikacja, w której zasobach znajduje się plik tekstowy z kompletem testowych danych logowania na środowisku produkcyjnym? Aplikacja z niedziałającymi mapami? Wymuszanie aktualizacji, które można ominąć dwoma kliknięciami?

Odnaleźliśmy wiele przykładów lenistwa lub niekompetencji programistów – na poziomie niedopuszczalnym w aplikacjach finansowych. O ile bowiem kod pozwalający na automatyczne wypełnianie formularzy czy zmianę adresu serwera może być niezbędny w testach, to zgodnie z zasadami sztuki nie powinien trafiać do wersji „sklepowej".

Produkcja części opisywanych aplikacji została zlecona przez banki na zewnątrz, co jest sygnałem, że po pierwsze nie traktuje się ich jak elementu kluczowej infrastruktury; po drugie zaś, że bank może nie mieć kompetencji, aby ocenić jakość odbieranego produktu. Dodatkowo bank, który traktuje aplikację mobilną po macoszemu, będzie miał problem z zatrudnieniem specjalistów od tej technologii, co utrwali niekorzystne praktyki.

Przyszłość mobilnej bankowości nie jest jeszcze określona. Na razie wygrywa wygoda dostępu do rachunku i płatności bezgotówkowych (jak **BLIK** czy wirtualizowane karty płatnicze). Można jednak wyobrazić sobie sytuację, w której kradzież środków przy użyciu dziesiątek czy setek tysięcy telefonów spowoduje całkowite porzucenie aplikacji bankowych. Bankom powinno zależeć, aby taki scenariusz pozostał fikcją.



Tak mogło wyglądać menu startowe aplikacji **BZ WBK**, jeśli użyliśmy jej w zasięgu sieci wifi PGS Software.

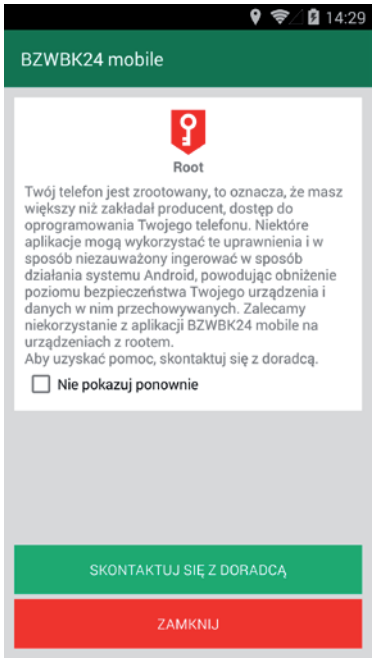
Android a bezpieczeństwo

Zanim przejdziemy do analizy konkretnych przypadków, krótka notka dotycząca Androida. Na rynku dostępne są setki modeli urządzeń z tym systemem, oferowane przez dziesiątki producentów. Spora część z nich sprzedawana jest za pośrednictwem operatorów komórkowych, którzy preinstalują własny software, przez co liczba możliwych konfiguracji oprogramowania sięga tysięcy.

Tymczasem aktualizacje oprogramowania, łatające luki w systemie operacyjnym, udostępniane są z reguły tylko modelom z najwyższej półki, a i to jedynie przez rok – dwa od wprowadzenia na rynek. Sprawia to, że w niektórych szacunkach[1] nawet 80% urządzeń z Androidem jest podatnych na atak z zewnątrz. W niektórych przypadkach użytkownik musi zainstalować złośliwą aplikację lub odwiedzić stronę internetową, jednak w przypadku usterki znanej pod nazwą ***Stagefright***[2] atakujący może uruchomić dowolny kod poprzez wysłanie wiadomości MMS ze specjalnie spreparowaną treścią. Uruchomienie złośliwego kodu odbywa się natychmiast.

Istotne błędy spotykane są też w konkretnych modelach lub seriach urządzeń. W 2015 roku ujawniono podatność Samsungów S4, S5 i S6 (oraz modeli pokrewnych) na atak przy użyciu sfałszowanej aktualizacji pakietu z klawiaturą systemową – także tu wykonanie złośliwego kodu było możliwe bez żadnej akcji użytkownika. Wystarczyło aktywne połączenie z internetem[3].

Podczas przygotowania niniejszego tekstu głośno zrobiło się o podatnościach ***Rowhammer*** (modyfikacja zawartości pamięci) oraz ***Dirty COW*** (podniesienie uprawnień procesu), każda z nich może zostać użyta w programie nie wymagającym szczególnych uprawnień.



Aplikacja **BZ WBK** ostrzega przed zrootowanym urządzeniem.

Na terytorium wroga

Aplikacje bankowe muszą być projektowane z myślą o tym, jak poradzą sobie na skompromitowanym urządzeniu. Na przykład blokada funkcji zrzutu ekranu, choć uniemożliwi wysłanie obrazka z potwierdzeniem wykonanego przelewu, zapobiegnie też podejrzeniu PIN-u. Wykorzystanie mechanizmu przypinania certyfikatu (certificate pinning), choć teoretycznie nadmiarowe, zapobiegnie podsłuchowi danych, gdy w systemie znalazł się podstawiony przez atakującego wrogi certyfikat. Będziemy o tym pisać w kolejnych rozdziałach.

Można przyjąć, że złośliwe oprogramowanie relatywnie często będzie mogło odczytywać logi oraz modyfikować pliki robocze tworzone przez aplikacje. Oznacza to, że dane użytkownika powinny być składowane w postaci zaszyfrowanej, a dane statyczne opatrzone sumą kontrolną lub podpisem cyfrowym, by wykryć nieautoryzowane modyfikacje. Skala takiego ataku może być bardzo duża.

Modyfikacja samej aplikacji bankowej przez atakującego będzie znacznie trudniejsza. Należy się raczej obawiać programowych nakładek (overlay), które wyświetlą formularz wyłudzający dane nad uruchamianą właśnie oryginalną apką – takie ataki mają już miejsce.

Przechwycenie ruchu sieciowego, również trudne, wydaje się bardziej prawdopodobne w przypadku ataków ukierunkowanych na konkretnego użytkownika, firmę lub sieć. Nie spodziewamy się tu ataków o dużej skali, za to pojedynczy atak może nieść większe szkody.

Sytuacja bieżąca w Polsce

Skoro jest tak źle, to dlaczego nie słyszy się dotąd o wielkoskalowych atakach na bankowość mobilną w Polsce? Składa się na to kilka powodów. Po pierwsze, dla obecnych na rynku tysięcy konfiguracji oprogramowania koniecznych byłoby kilkadziesiąt wariantów ofensywnego kodu, których nie da się próbować sekwencyjnie (bo np. pierwszy chybiony sposób zawiesi telefon). Po drugie – aplikacji bankowych do zaatakowania też jest przynajmniej kilkanaście, co zmusza do poszukiwania słabości w każdej z nich osobno. Po trzecie – o ile złośliwe oprogramowanie może mieć łatwy dostęp do plików czy logów systemowych, o tyle wyciąganie danych z pamięci uruchomionej aplikacji czy przechwytywanie komunikacji sieciowej jest już bardzo trudne (tym bardziej, że atak musi przebiegać autonomicznie). Po czwarte zaś – kanał

1. http://androidvulnerabilities.org/
2. https://www.kb.cert.org/vuls/id/924951
3. https://www.nowsecure.com/blog/2015/06/16/remote-code-execution-as-system-user-on-samsung-phones/

mobilny często nie pozwala na zerwanie lokaty, czy wzięcie kredytu, przez co złodziej może wyprowadzić co najwyżej środki z rachunku bieżącego.

Póki więc autorzy złośliwego oprogramowania osiągają wyższą stopę zwrotu na tworzeniu globalnych botnetów do wynajęcia lub szyfrowaniu danych i pobieraniu za nie okupu (*ransomware*), bankowość mobilna średniej wielkości kraju w Europie nie będzie podstawowym celem. Szczególnie, gdy próba ataku ma być podjęta z zagranicy, a wymaga użycia bezbłędnych tekstów w języku polskim.

Nie znaczy to jednak, że banki mogą spać spokojnie. Na myśl przychodzi stary dowcip – *uciekając przed lwem nie trzeba być pierwszym, wystarczy nie być ostatnim*. Analogicznie, bez stałego doskonalenia aplikacja spadnie na koniec stawki i – jako najbardziej podatna – skupi na sobie większość ataków.

Poważny atak nowej generacji wykorzysta wszystkie funkcje, które dziś tak cieszą marketingowców. Przykład? W świecie geolokalizacji i analizy zachowań konsumentów niepotrzebne będzie zakładanie konta *na słupa*. Wystarczy, że kupisz bilet do kina i wyłączysz telefon – od tej chwili przez dwie godziny twój rachunek może być używany przez złodziei jako stacja przesiadkowa dla serii szybkich przelewów, o których w najlepszym razie dowiesz się z SMSów odebranych po seansie.

A'propos SMSów – można w ciemno zakładać, że malware (złośliwe oprogramowanie) jest w stanie odczytać, wyciszyć i odesłać osobie trzeciej każdą wiadomość tekstową z kodem uwierzytelniającym. Dlatego całkiem możliwe, że za jakiś czas przeprosimy się z papierowymi listami kodów jednorazowych.

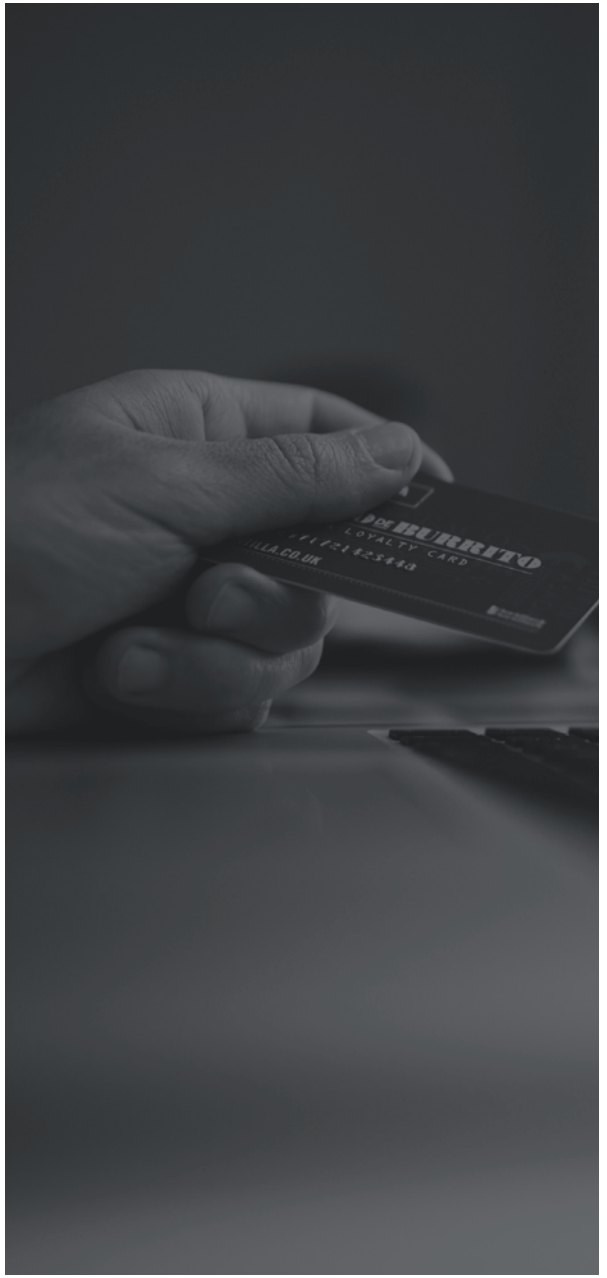
Testowane wersje aplikacji

Wszystkie aplikacje przetestowaliśmy w wersji dostępnej 1 lipca 2016 oraz w wersji aktualnej podczas testu (między lipcem a październikiem). Opis dostrzeżonych błędów i podatności wysłaliśmy do każdego banku z prośbą o ich poprawienie oraz informacją, że zostaną wspomniane w niniejszym raporcie.

Nie wszystkie usterki zostały poprawione. Niektóre, choć piszemy o nich w czasie przeszłym, są nadal obecne w momencie zamknięcia tego artykułu.

Podatności krytyczne

Mianem krytycznych określamy te usterki, które mogą doprowadzić do utraty środków lub wycieku danych na *zdrowym* urządzeniu. W naszych badaniach spostrzegliśmy ich co najmniej kilka. Dołączyliśmy do nich dodatkowo dwa błędy demonstrujące wyraźne lekceważenie dobrych praktyk przez banki.



BZ WBK

wyciek danych osobowych

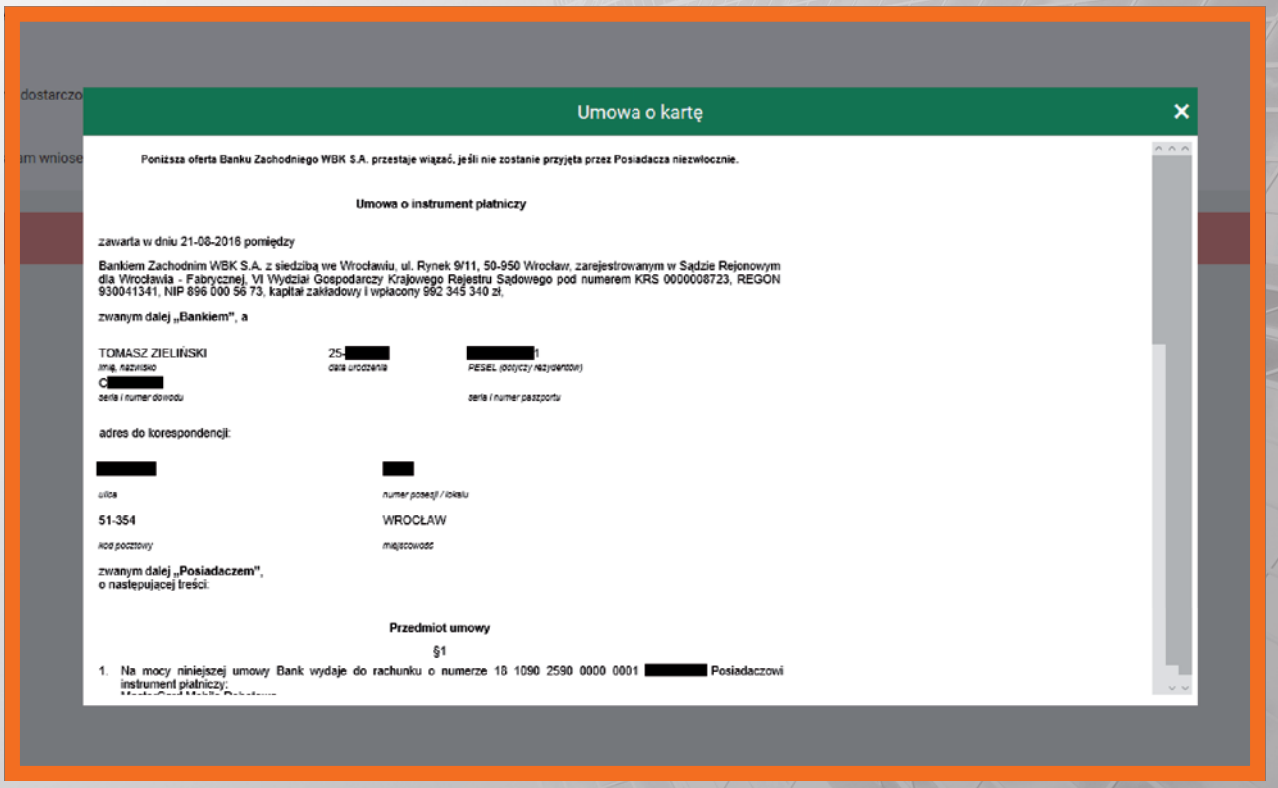
Gdy użytkownik nie dysponuje kartą płatniczą, wybranie opcji płatności zbliżeniowych otwiera formularz z wnioskiem o wydanie nowej karty. Formularz ten nie stanowi jednak części programu, lecz jest wyświetlany w przeglądarce internetowej osadzonej w aplikacji. Problem: adres formularza, zawierający jednorazowy token i identyfikator użytkownika trafiał do logów, skąd mógł go wykraść atakujący. W efekcie możliwe było zamówienie w imieniu użytkownika niechcianej usługi oraz poznanie danych osobowych (imię, nazwisko, PESEL, adres, numer dowodu) wyświetlanych we wniosku.

Co gorsza, formularz – mimo wylogowania z aplikacji mobilnej – pozostawał aktywny ponad pół godziny, co umożliwiało atak odsunięty w czasie. Można go było otworzyć na pececie pracującym w zupełnie innej sieci niż telefon (taka zmiana powinna zostać wykryta i uznana za atak).

Notka techniczna – logi systemowe

Logcat to w systemie Android strumień danych, do którego mogą pisać aplikacje. Najbardziej typowe zastosowanie to diagnostyka – do logów można wysyłać wpisy o zdarzeniach i stanie programu, trafiają tam też informacje o przyczynach awarii aplikacji. To, co jest pomocne dla programisty, może zawierać wskazówki dla atakującego. Dobrą praktyką jest, aby opublikowane wersje aplikacji w ogóle nie korzystały z tego mechanizmu.

W Androidzie do wersji 4.0 aplikacje z odpowiednim uprawnieniem mogły podsłuchiwać cały strumień logów. Od wersji 4.1 wzwyż możliwość ta została usunięta. Bufor systemu logowania przechowuje do kilku tysięcy ostatnich pozycji, więc można znaleźć tam wpisy sprzed nawet kilku godzin.



Formularz wykradziony z komórki można otworzyć na komputerze.

ING

przejęcie sesji

Błąd krytyczny w aplikacji **ING** też spowodowany był wyciekiem linku do logów systemowych, ale tu winne okazało się... zabezpieczenie webowego serwisu transakcyjnego przed atakiem typu self-XSS. W ramach tego zabezpieczenia strona wyświetlała w konsoli Javascriptu ostrzeżenie, by użytkownik nie wklejał tam żadnych treści. W systemie Android przeglądarka nie ma konsoli, więc alert trafiał do logów systemowych, przy czym przeglądarka doklejała do niego adres strony źródłowej.

Skąd jednak brał się sam link? Starsza wersja aplikacji mobilnej (przemianowana w trakcie testów na *ING dla przedsiębiorców*) została wyposażona w przycisk, który jednym kliknięciem przenosi zalogowanego użytkownika do aplikacji webowej, bez konieczności ponownego logowania (tzw. Single Sign-On, SSO). Ten właśnie link, generowany w aplikacji i przekazywany do mobilnej przeglądarki, trafiał w opisany wyżej sposób do logów.

Okazuje się, że natychmiastowe przesłanie adresu znalezionego w logach do komputera i otworzenie go w kilku oknach przeglądarki sprawiało, że pecet wygrywał wyścig i sesja zalogowanego użytkownika zostawała wykradziona z komórki, co dawało atakującemu dostęp do serwisu transakcyjnego ofiary.

To ostrzeżenie widoczne w konsoli przeglądarki było przyczyną problemów na urządzeniu mobilnym.

Notka techniczna - Self-XSS

Błąd typu XSS polega na wykonaniu w przeglądarce niepożądanego kodu wstrzykniętego przez atakującego. Efektem może być np. zakupienie towaru na aukcji internetowej bez żadnej akcji użytkownika. Wielokrotnie spotykane były np. ataki XSS na Facebooka, których ofiary automatycznie publikowały coś na wallu swoim lub wszystkich znajomych (infekując ich przy okazji).

Atak self-XSS to wariant rozpowszechniany metodą socjotechniczną – ofiarę należy przekonać, aby odwiedziła wybraną stronę, otworzyła konsolę Javascriptu (przydatną jedynie programistom) oraz wkleiła tam fragment kodu. Użytkownicy Facebooka byli np. kuszeni możliwością podglądu prywatnych wiadomości lub przejęcia konta innego użytkownika, a w efekcie sami stawali się ofiarami ataku.

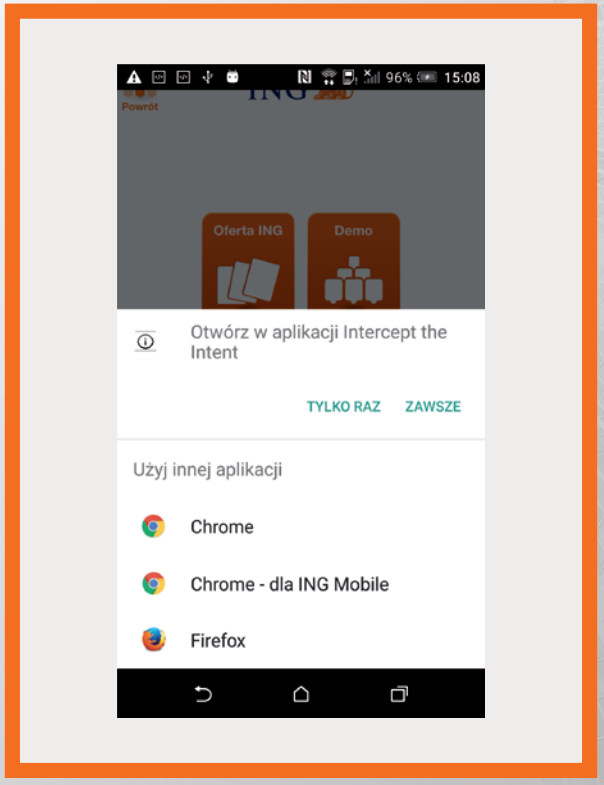
ING

użycie implicit intentów pozwala na przejęcie linku SSO

Z opisanym wyżej mechanizmem otworzenia zalogowanej sesji w przeglądarce wiąże się jeszcze jeden problem – aplikacja korzystała z systemowego mechanizmu *intentów*, czyli możliwości wybrania, który program zrealizuje rozpoczętą akcję.

Sam mechanizm jest dużą przewagą konkurencyjną Androida. Pozwala użytkownikom wygodnie przesyłać dane między aplikacjami, a programistom definiować, do których akcji i jakich typów danych rejestrować aplikację. W efekcie akcja ***Udostępnij tekst*** zaproponuje przekazanie tekstu do SMSa, e-maila, Skype'a, Evernote'a i innych programów potrafiących skorzystać z danych tekstowych.

Problem pojawia się, gdy chcemy przekazać w ten sposób sekret (tu: adres webowy SSO). Atakujący może zarejestrować się pod akcję ***Przeglądaj HTTPS***, podając jako filtr odnośniki do strony ***ingbank.pl***. Jego akcja może być wówczas widoczna w menu np. jako ***Chrome dla ING***. Jej wybranie poskutkuje przejęciem sesji lub – co znacznie gorsze – podstawieniem treści tak, by użytkownik jak najdłużej miał wrażenie, że korzysta z normalnej aplikacji webowej.



Chrome dla ING to fałszywka, tym razem podstawiona przez nas.

PEKAO

wyciek hasła maskowanego

Aplikacja **Pekao** wrzucała do logów kopię całej komunikacji sieciowej (zła praktyka). W ramach tej komunikacji poszczególne litery hasła maskowanego przesyłane były otwartym tekstem (zła praktyka, transmitowany powinien być wynik jednokierunkowej funkcji skrótu czyli tzw. hash). Aplikacja nie blokowała też komunikacji z użyciem podstawionego certyfikatu (zła praktyka).

Jeśli więc używasz telefonu w pracy, gdzie w celu korzystania z Wi-fi musiałeś zainstalować firmowy certyfikat, lepiej dbaj o dobre relacje z adminem. Po kilku logowaniach do aplikacji **Pekao** może on znać całe twoje hasło.

Aplikacja **Pekao** – kopia ruchu sieciowego trafiała do logów systemowych.

IDEA BANK

odsyłanie do aplikacji kompletu danych osobowych

O technicznym aspekcie ochrony kanału transmisyjnego będziemy jeszcze pisać. Tu wspomnimy tylko, że aplikacja **IdeaBanku** nie blokowała możliwości podsłuchu danych przy użyciu proxy i podstawionego certyfikatu, co w przypadku mobilnych aplikacji finansowych jest niedotrzymaniem dobrych praktyk.

Gdy zajrzymy do danych transmitowanych z serwera do aplikacji mobilnej, po plecach może przejść dreszcz. Oprócz imienia, nazwiska, adresu, numeru PESEL mieliśmy tam numer telefonu, nazwisko panieńskie matki i numer dowodu tożsamości (oraz dane przetwarzane wewnętrznie przez bank, np. piszącemu niniejsze słowa przypisano poziom ryzyka *podwyższony*).

Co do zasady informacje, które nie są przetwarzane przez aplikację mobilną, nie powinny do niej w ogóle trafiać. Udane przechwycenie danych z aplikacji **IdeaBanku** w większości telefonicznych BOK-ów wystarczy do zresetowania hasła i całkowitego przejęcia danych logowania. Może być więc pierwszym krokiem obejmującego wiele instytucji ataku na konkretnego człowieka.

```
rozne okresy w lokacie;47507073;Solaris86;1005871
57178520;57178520;123456Qq;none
lokatowywow;lokatowywow;123456Qq;1006962
49884385;49884385;Solaris86;none
Synchro1;Synchro1;123456Qq;none
22718740;22718740;Solaris86;none
Solarissss;Solarissss;Solaris86;none
edycja blad;36620706;123456Qq;none
user1;17345296;Marcin12;2964
user1a;86704303;Marcin12;1000565
user2;52160468;Marcin12;1000568
user3;56436841;123456Qq;1002921
user4;27251376;123456Qq;1002942
user5;31274110;123456Qq;1002922
user6;12083370;5169389543;1003053
user8;24843014;Przemek1;1002307
user9;44281517;123456Qq;1003462
user10;70045088;Marcin12;1000897
user12;10773883;123456Qq;1003793
user13;96659482;Marcin12;5593
user14;35767912;Kanapka6;1001703
user15;66723493;Przemek1;1002299
user16;97366064;123456Qq;1005394
user17;43279804;Marta123;1003822
user18;88309903;Goosip2008;7491
user19;60207105;123456Qq;1005442
user20;46626671;123456Qq;1005598
user21;97404421;123456Qq;1005603
AnnaMaria;AnnaMaria;Marta123;none
Historia;17261962;123456Qq;none
Struktury;47403899;Marta123;1008314
Lokata STX;45072643;123456Qq;1010763
TylkoLokata;94972961;123456Qq;none
BezKont;94603329;123456Qq;none
BezKont;93401487;123456Qq;none
BezKont;92549661;123456Qq;none
BezKontPerson;38580461;123456Qq;none
BezKontFirma;85607457;123456Qq;none
PersonOnly;50093075;Solaris86;none
WnioskiLokaty;74805641;Solaris86;none
user28532;123456Qq;123456Qq;1005966
user28530;71453469;123456Qq;1005965
user28528;66994323;123456Qq;1005964
```

Loginy i hasła znalezione w aplikacji **IdeaBank.**

hasła do kont testowych w zasobach aplikacji

Aplikacja **IdeaBanku** zawierała w zasobach plik tekstowy z przeszło setką testowych loginów i haseł. Część z nich działała w środowisku produkcyjnym. W ten sposób dowiedzieliśmy się, że najpopularniejsze hasło testerów to "123456Qq", o kilka długości wyprzedzające "Kanapka6" i "Solaris86". Teraz chcielibyśmy o tym jak najszybciej zapomnieć.

Bezpieczeństwo komunikacji

Notka techniczna

Skąd wiemy, że połączenie z serwerem jest bezpieczne? Użytkownicy bankowości internetowej przywykli do sprawdzania zielonej kłódki w pasku adresu – jej obecność oznacza, że przeglądarka na pewno łączy się z serwerami wybranego banku, i że łączność ta jest realizowana w bezpieczny sposób. Od strony technicznej, podczas nawiązywania połączenia następuje weryfikacja niepodrabialnego klucza publicznego, opatrzonego podpisem jednego z zaufanych urzędów certyfikacji (w Polsce jest to np. Krajowa Izba Rozliczeniowa lub Polska Wytwórnia Papierów Wartościowych).

Każdy komputer i telefon komórkowy zawierają listę od kilkudziesięciu do kilkuset zaufanych urzędów certyfikacji. Problem pojawia się, gdy atakujący zdoła dodać do tej listy siebie (wystarczy mu minuta sam na sam z klawiaturą). Od tej chwili może wystawić podrobiony certyfikat, sfałszować zieloną kłódkę i podsłuchiwać oraz modyfikować cały ruch sieciowy do i z banku. To właśnie dlatego banki cały czas przypominają, że do bezpiecznego korzystania z bankowości mobilnej oraz internetowej nigdy nie potrzeba instalacji pluginów, wtyczek, dodatków, "bezpieczniejszych" certyfikatów, "dodatkowych zabezpieczeń" ani tym podobnych.

Jeśli lista zaufanych urzędów certyfikacji zostanie "skażona" przez atakującego, do dyspozycji jest jeszcze jedna, ostatnia linia obrony. Autorzy aplikacji mogą umieścić wzorcowy certyfikat wewnątrz aplikacji, aby możliwe było potwierdzenie, że łączymy się z docelowym serwerem, bez żadnych niepożądanych pośredników. Jest to tzw. "przypinanie certyfikatów" (certificate pinning). Należy ono do dobrych praktyk, które powinny być dotrzymywane przez twórców wszystkich aplikacji finansowych.

Aplikacja mobilna musi komunikować się serwerami banku. Z nich pobierze informacje o aktywnych usługach, historii rachunków, wysokości salda; w drugą stronę odeśle zlecenia przelewu, założenia lokaty czy zamówienia karty płatniczej. Atakującemu będzie zależało zarówno na przechwyceniu takich danych (historia rachunku i operacje na koncie mogą zawierać informacje poufne) jak i modyfikacji transmisji (np. w celu podstawienia numeru rachunku, na który ma trafić przelew). Mechanizmy komunikacji muszą więc zapewnić zarówno poufność, jak i integralność przesłanych komunikatów.

Czy testowane aplikacje spełniły oczekiwania w tym zakresie? Niestety, nie do końca.

Niezrealizowane dobre praktyki

Podczas testów okazało się, że nie wszystkie aplikacje realizują oczekiwane standardy. Wiele poddało się bez walki po przepuszczeniu ruchu sieciowego z komórki przez komputer i uruchomieniu serwera proxy z własnym certyfikatem.

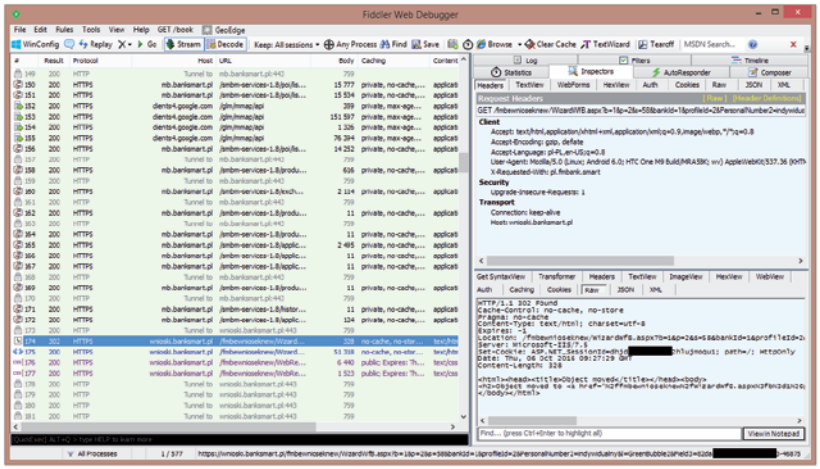
W przypadku aplikacji **mBanku**, **IdeaBanku**, **T-Mobile Usługi Bankowe**, **Orange Finanse** czy **Banku Smart** – kanał transmisyjny stał przed nami otworem. Proxy przechwytujące ruch umożliwiało modyfikację żądań i odpowiedzi. Jak uprzednio wspomnieliśmy, w przypadku **IdeaBanku** już samo podsłuchanie danych mogło mieć fatalne skutki.

Programy banków **Pekao**, **Banku Zachodniego**, **Raiffeisen** i **Citi** także komunikowały się z częścią serwerową, pozwalając na podsłuchanie transmisji dodatkowo szyfrowanej lub opatrywanej podpisem cyfrowym. Daje to adwersarzowi możliwość systematycznego poszukiwania słabości w zabezpieczeniach podsłuchiwanej komunikacji, nawet jeśli nie jest możliwa dowolna modyfikacja treści.

Dodatkowy problem dotyczy aplikacji opartych o technologie webowe, które pobierają moduły wykonawcze z internetu (np. **Citi** lub nowa wersja aplikacji **ING**). Tam jednorazowe udane wstrzyknięcie wrogiego kodu trwale zniweczy wszelkie inne zabezpieczenia.

Decyzję o bezwarunkowym blokowaniu komunikacji podejmują apki **Getin**, **Millennium**, **BGŻ**, **BPH**, **Alior Banku**, **Eurobanku** i **Credit Agricole**. Na szczególne wyróżnienie zasługuje tu bank **Millennium**, który chwilę po uniemożliwieniu próby logowania przesyła do centrali… kopię *lewego* certyfikatu. Zgadujemy, że nasz podstawiony certyfikat opatrzony nazwą *NIE_UFAĆ_CERTYFIKAT_FIDDLERA* (serwera proxy) raczej alarmu nie wzbudził, jednak opisany mechanizm raportowania może w porę ostrzec o rozpoczęciu rzeczywistego ataku.

Pochwaloną przed momentem aplikację **Millennium** musimy jednak zganić za zapisywanie historii działań użytkownika do pliku i przechowywanie tych plików przez długi czas; w katalogu aplikacji znaleźliśmy też cache z plikami pobieranymi z sieci – siedziały tam od kilku miesięcy.



Do podsłuchiwania komunikacji z serwerami banku służy m.in. program Fiddler.

Brak szyfrowania transmisji

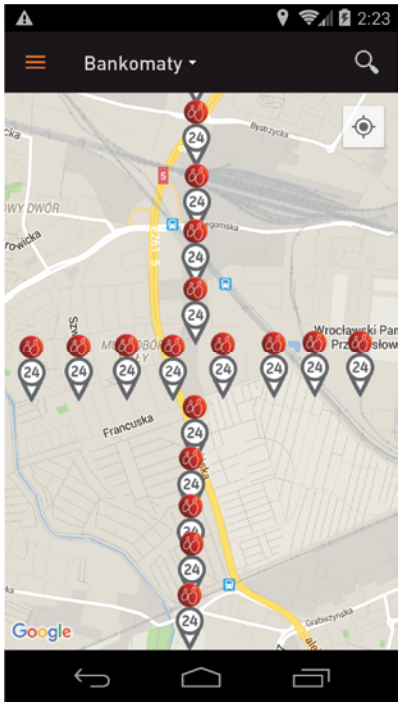
Nie każdy użytkownik internetu orientuje się, że ***zwykłe*** połączenia sieciowe (np. realizowane popularnym protokołem HTTP) mogą być podsłuchane i modyfikowane na każdym urządzeniu, które pośredniczy w transmisji danych. Gdy w Polsce zaczęły pojawiać się pierwsze blokowe i osiedlowe sieci komputerowe, których członkowie dzielili między sobą wysokie wówczas koszty stałego łącza, domorośli admini robili dowcipy polegające np. na lustrzanym odwróceniu wszystkich obrazków na stronach odwiedzanych przez upatrzonego użytkownika. Co bardziej wścibscy przechwytywali i czytali pocztę członków swojej sieci.

Dostęp do internetu realizują dziś profesjonalni operatorzy, których nie trzymają się proste żarty, jednak ryzyko ataku *Man in the Middle* (polegającego na sfałszowaniu zapytania lub odpowiedzi) jest nadal realne – zwłaszcza, jeśli zdarza nam się skorzystać z bezprzewodowej sieci miejskiej czy hotelowej.

Tu autorzy opisywanych programów znów zawiedli – odnośniki do zasobów serwowanych przez niezabezpieczony protokół HTTP dostrzegliśmy w aplikacjach **PKO BP**, **Pekao**, **BZ WBK**, **mBanku**, **ING**, **Getin Banku**, **BGŻ**, **BPH** , **Alior Banku**, **IdeaBanku**, **Eurobanku**, **CreditAgricole**. Czasem były to linki do dokumentów w formacie PDF, gdzie indziej do regulaminów, stron z ofertą czy nawet odnośniki z głównego menu aplikacji.

Bywały jednak przypadki bardziej widowiskowe – aplikacja **BZ WBK** pobierała obrazek do menu głównego przez połączenie nieszyfrowane. Możliwe konsekwencje prezentujemy na stronie 3. Aplikacja **BPH** ściąga w taki sposób listę bankomatów – wstrzyknięta lista może odsyłać klientów **BPH** do maszyn konkurencji . Wstrzyknięte kursy walut mogą przyprawić o krótkotrwałą euforię lub zawał. Politowanie budzi instrukcja bezpiecznego korzystania z aplikacji mobilnej załadowana przez ***gołe*** połączenie HTTP (**Alior Bank**, **BZ WBK**, **PKO BP**).

Dlaczego brak szyfrowania jest niebezpieczny? Wyobraźmy sobie, że atakujący przechwytuje żądanie pobrania regulaminu opłat i odsyła z powrotem dokument z czerwonym napisem „Urządzenie niezaufane, pobranie dokumentu będzie możliwe po instalacji rozszerzonego certyfikatu. Zrób to teraz! Pobierz certyfikat ze strony [...]". Pytanie konkursowe – ile osób własnoręcznie zainstaluje sobie złośliwe oprogramowanie wierząc, że wykonuje polecenia banku?



Mapa bankomatów pobrana przez niezabezpieczone połączenie HTTP.

Problemy z witrynami WWW banków

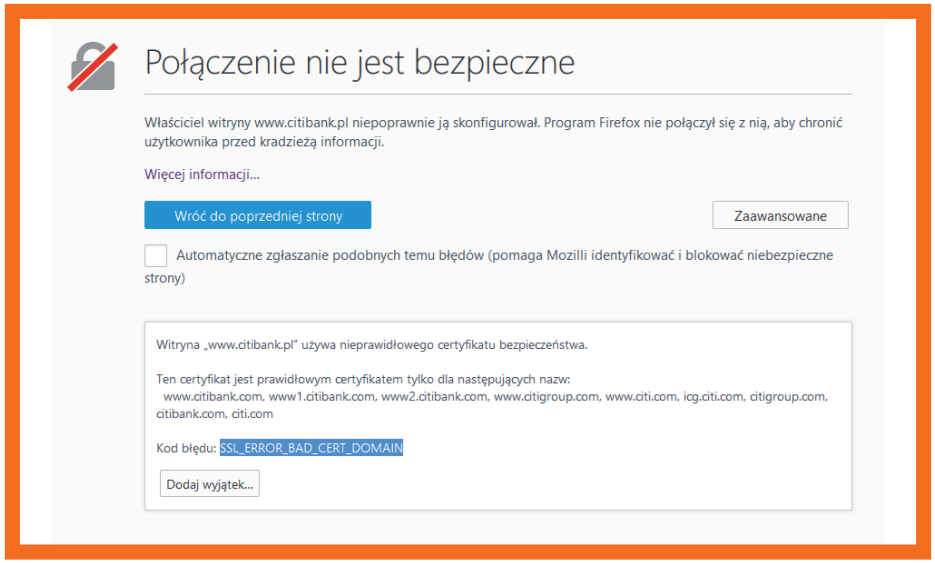
Skoro mowa o komunikacji sieciowej, czas napiętnować praktyki dotyczące witryn internetowych kilku z opisywanych banków.

Chwilę temu wspominaliśmy o niebezpieczeństwach transmisji danych przez niezabezpieczony protokół HTTP. Trudno w to uwierzyć, ale po dziś dzień na rynku istnieją banki, które nie mają strony ofertowej dostępnej przez połączenie szyfrowane. Jeśli użytkownik ręcznie dopisze "https" do adresu witryny **Banku Smart**, zostanie przekierowany na połączenie niezaufane ***http***. To samo osiągniemy w przypadku inteligo.pl oraz iko.pkobp.pl, marek **PKO BP**. Jeszcze podczas wakacji 2016 taki sam efekt oglądaliśmy na stronie pkobp.pl i pekao.com.pl, na szczęście to już przeszłość.

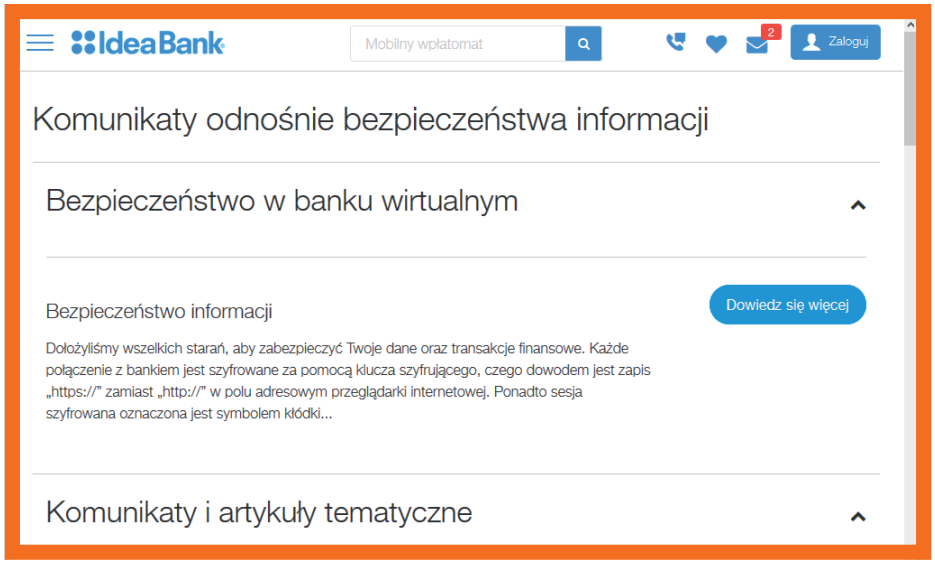
Witryna bph.pl nie ma wariantu bezpiecznego, bo przez ten sam adres z ***https*** serwuje stronę logowania do systemu transakcyjnego. Próba wejścia na https://raiffeisenpolbank.com nigdy się nie powiedzie, połączenie nie zostanie nawiązane.

Co w przypadku banków, które nie potrafią zdecydować się na jedną domenę? Marka **Orange Finanse** korzysta z domen orangefinanse.com, orangefinanse.pl, www.orange.pl oraz orangefinanse.com.pl. Dwie pierwsze nie załadują się przez HTTPS z powodu niezgodności nazwy domeny z certyfikatem, trzecia przekieruje na HTTP, jedynie czwarta zadziała jak należy (jest to jednak strona logowania). Z domen citibank.pl i citihandlowy.pl przez HTTPS nie działa żadna, podobnie jak citibankonline.com (ponownie – niezgodność nazw).

Powiedzmy wprost: osoby odpowiedzialne za stan rzeczy opisany w tym rozdziale powinny spalić się ze wstydu. W dzisiejszych czasach od banków oczekujemy wymuszonego połączenia HTTPS i certyfikatu Extended Validation na każdej stronie ofertowej, nie wspomniawszy o serwisie transakcyjnym. W chwili pisania niniejszych słów wzorcowymi witrynami dysponowały **mBank**, **Millennium**, **Alior Bank**, **Idea Bank**, **Eurobank** i **Credit Agricole**.



Próba otworzenia strony https://www.citibank.pl



Przycisk „Dowiedz się więcej" odsyła do dokumentu PDF serwowanego przez niezabezpieczony protokół HTTP.

Jak znaleźć aplikację banku w sklepie Play?

Pytanie pozornie banalne – należy znaleźć link na stronie banku albo użyć sklepowej wyszukiwarki. Ale czy na pewno? Sklep Play słynie z dość swobodnego podejścia do nadużyć – raczej im nie zapobiega, akcje podejmowane są dopiero po wystąpieniu szkody. Próba logowania do podstawionej aplikacji może skończyć się bardzo źle, więc lepiej trafić za pierwszym razem. Z pomocą może przyjść nazwa pakietu, unikalna w obrębie całego sklepu i widoczna w adresie URL Google Play.

Niektóre domeny odpowiadają bezpośrednio nazwom pakietów aplikacji:

- mbank.pl – pl.mbank,
- eurobank.pl – pl.eurobank.

Czasem jest wystarczająco blisko, np.:

- bzwbk.pl – pl.bzwbk.bzwbk24,
- pkobp.pl – pl.pkobp.iko.

Niestety, czasem nazwa pakietu reklamuje raczej wykonawcę aplikacji niż reprezentuje bank:

- pekao.com.pl – eu.eleader.mobilebanking.pekao,
- bgzbnpparibas.pl – com.comarch.mobile.banking.bnpparibas.

W skrajnych przypadkach nazwa pakietu albo niczego nie powie o banku:

- com.konylabs.cbplpat to CitiHandlowy

albo okaże się wręcz myląca:

- alior.bankingapp.android to... T-Mobile Usługi Bankowe.

W takiej sytuacji nietrudno o fatalną w skutkach pomyłkę przy instalacji.

Screenshoty i nie tylko

Android umożliwia zapisywanie zrzutów ekranu. Z reguły wywołanie tej funkcji ukryte jest pod odpowiednią kombinacją przycisków lub opcją menu systemowego. Zwykłe programy nie mogą uzyskać dostępu do zawartości wyświetlanej przez inne aplikacje. Zapewne z tego powodu aplikacje **PKO BP**, **Pekao**, **mBanku**, **Orange Finanse**, **Raiffeisena**, **Citi**, **BPH**, **Eurobanku**, **ING, Alior Banku, T-Mobile Usługi Finansowe i Banku Smart** nie blokują możliwości robienia zrzutów ekranu podczas logowania, choć ujęcia klawiatury ekranowej jasno pokazują sekwencję wciskanych klawiszy. W przypadku **Idea Banku** nie wycieknie hasło logowania, ale kolejne ekrany ujawnią pozostałe dane widoczne na wyświetlaczu.

Seria zrzutów ekranu zdradzi PIN użytkownika.

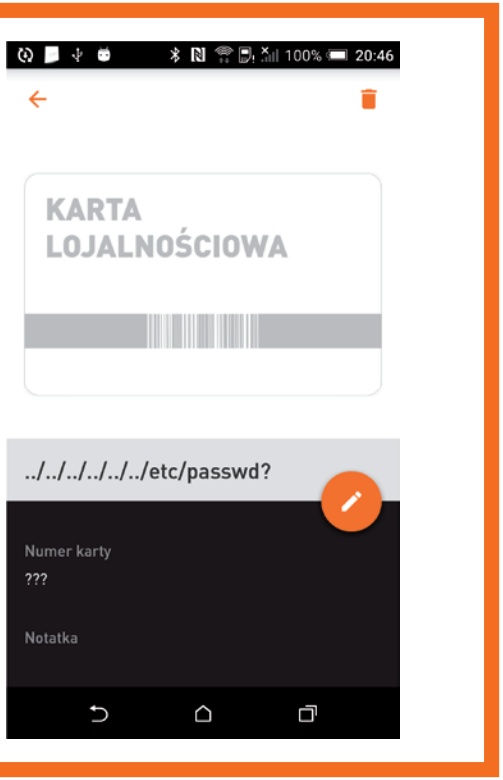
Opcję zrzutu ekranu zablokowano za to w aplikacjach **BZ WBK**, **Getin Banku**, **Millennium**, **BGŻ** i **Credit Agricole**.

Wydawać by się mogło, że jest to przesadna ostrożność, gdyby nie przykład Alcatela A564C, w którego oprogramowaniu producent umieścił (prawdopodobnie omyłkowo) pakiet diagnostyczny Qualcomm SystemAgent[5]. Pozwalał on wszystkim programom na wykonywanie zrzutów ekranu bez żadnych szczególnych uprawnień, o ile tylko programista *fotografowanej* apki aktywnie temu nie przeciwdziałał.

Inną usterką dostrzeżoną w aplikacji **PKO BP** jest deserializacja niezaufanych danych przekazanych wraz z poleceniem uruchomienia aplikacji. Pozwala to atakującemu na wstrzyknięcie obiektu wykorzystywanego potem przez aplikację. Błędy tego typu mogą prowadzić do wykonania niepożądanego kodu.

Aplikacja **BPH** w module kart lojalnościowych używała nazwy podanej przez użytkownika jako fragmentu nazwy pliku, tymczasem możliwe było wpisanie tam np. ciągu znaków *../../../* (przejście po katalogach). Ślepe ufanie użytkownikowi jest złą praktyką. Tu może sprawić, że zamiast zdjęcia karty zostanie odczytany zupełnie inny plik.

W przypadku części aplikacji niepokój budzi duża liczba dołączanych bibliotek i komponentów – nawet ponad dwadzieścia sztuk w jednym produkcie. Komponenty firm trzecich mogą być istotną pomocą dla programistów, jednak sprawia to, że gotowa aplikacja sumuje usterki i podatności własnego kodu oraz wszystkich elementów wchodzących w jej skład.



Aplikacja **BPH** – użycie tekstu wpisanego przez użytkownika jako fragmentu nazwy pliku.

5. https://github.com/rednaga/disclosures/blob/master/QCOMSysAgent.md

Architektura aplikacji

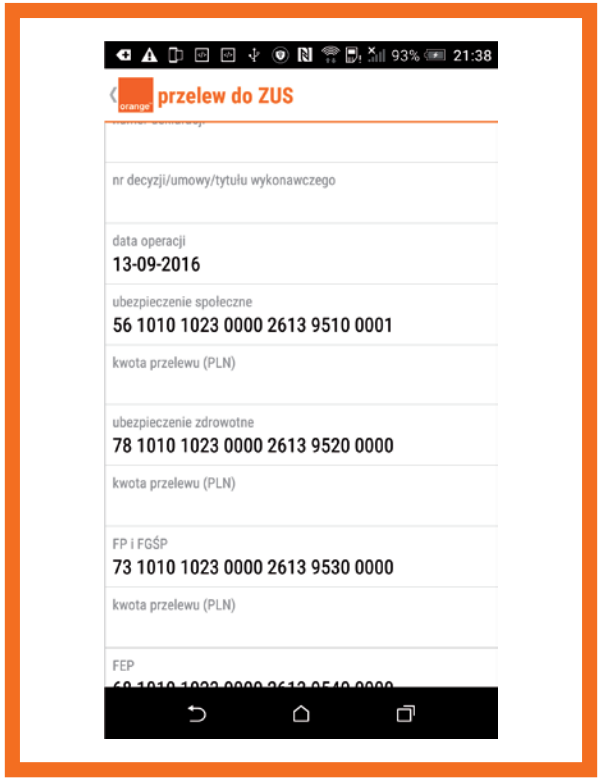
Poważnym błędem w aplikacjach **BGŻ** i **Alior Banku** jest łączenie się z serwisem (komponentem programu działającym w tle) poprzez nazwę, pod którą może podszyć się każdy inny program. Jeśli ów inny program był zainstalowany wcześniej, przekazanie danych na zewnątrz aplikacji jest nieuniknione. Ostrzeżenie o niebezpieczeństwie było przekazywane przez system Android do logów, gdzie zignorowali je tak programiści, jak i testerzy.

```
<v:Envelope xmlns:v="http://schemas.xmlsoap.org/soap/envelope/" xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns:d="http://www.w3.org/2001/XMLSchema"
xmlns:c="http://schemas.xmlsoap.org/soap/encoding/" xmlns="http://mbank.pl/">
  <v:Header>
    <RequestInformation>
      <AppType>AND</AppType>
      <AppVersion>2.7.3</AppVersion>
      <BankId>1</BankId>
      <SessionId>scSunD-9lEy[redacted]6Gdg==</SessionId>
    </RequestInformation>
  </v:Header>
  <v:Body>
    <ZUSRegisterTransfer>
      <ammount1>11.0</ammount1>
      <ammount2>11.0</ammount2>
      <ammount3>11.0</ammount3>
      <ammount4>11.0</ammount4>
      <creditAccount1Number>56101010230000261395100001</creditAccount1Number>
      <creditAccount2Number>78101010230000261395200000</creditAccount2Number>
      <creditAccount3Number>73101010230000261395300000</creditAccount3Number>
      <creditAccount4Number>68101010230000261395400000</creditAccount4Number>
      <decisionNumber></decisionNumber>
      <declaration>072016</declaration>
      <declarationNumber>01</declarationNumber>
      <documentNumber>A[redacted]01</documentNumber>
      <documentType>1</documentType>
      <email1></email1>
      <email2></email2>
      <isItFirstCall>true</isItFirstCall>
      <nipNumber>[redacted]</nipNumber>
      <payType>S</payType>
```

Treść przechwyconego (i zatrzymanego) komunikatu z poleceniem przelewu ZUS.

Nie wszystkie programy kończyły pracę po zadanym okresie bezczynności. Na ekranie pozostały dane finansowe użytkownika **mBanku**, **BGŻ**, **BPH**, **Alior Banku**, **Credit Agricole**, **Orange Finanse**, **Banku Smart**. Większość tych aplikacji reagowała wylogowaniem dopiero po dotknięciu ekranu.

Podczas przyglądania się plikom roboczym programów dostrzegliśmy, że część aplikacji jest parametryzowana danymi z bazy danych lub pliku XML. Niestety, integralność takich zbiorów nie jest chroniona. Słabości odkryliśmy w aplikacjach **Millennium**, **mBanku** i **Orange Finanse**. Zmodyfikowaliśmy tam pliki (symulowany efekt udanego ataku), co zaowocowało wyświetleniem w formularzu ZUS fałszywego numeru konta ubezpieczeń społecznych. Nie był to jedynie problem z prezentacją danych – aplikacje **mBanku** i **Orange Finanse** rzeczywiście próbowały wysłać taki przelew! W tym miejscu musieliśmy niestety przerwać transmisję, by nie narażać części serwerowej na nieuzgodnione testy.



Podstawiony numer konta ubezpieczeń społecznych ZUS w aplikacji **Orange Finanse**.

Niektóre aplikacje komunikują się w ramach jednego banku z różnymi systemami informatycznymi, o różnych adresach, protokołach i poziomach zabezpieczeń. Jest to niezalecane. Cała usługa mobilna jest wówczas tak bezpieczna, jak najsłabszy z elementów składowych.

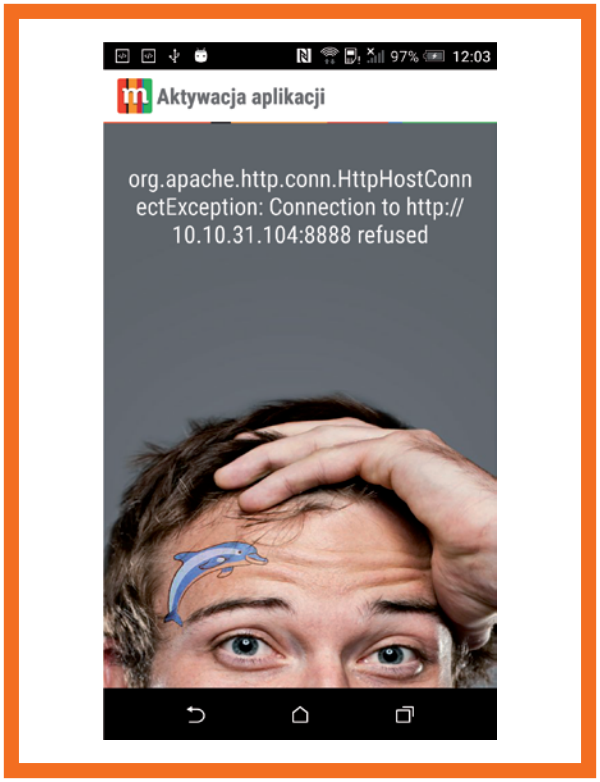
Słaba kontrola jakości aplikacji

Podczas badania aplikacji trafialiśmy na błędy, które powinny zostać wyłapane na wczesnym etapie wewnętrznych testów. Podkreślamy, że podczas zbierania materiałów poświęcaliśmy jednej aplikacji raptem kilka godzin, czyli wielokrotnie mniej, niż powinny trwać testy regresji każdego nowego wydania.

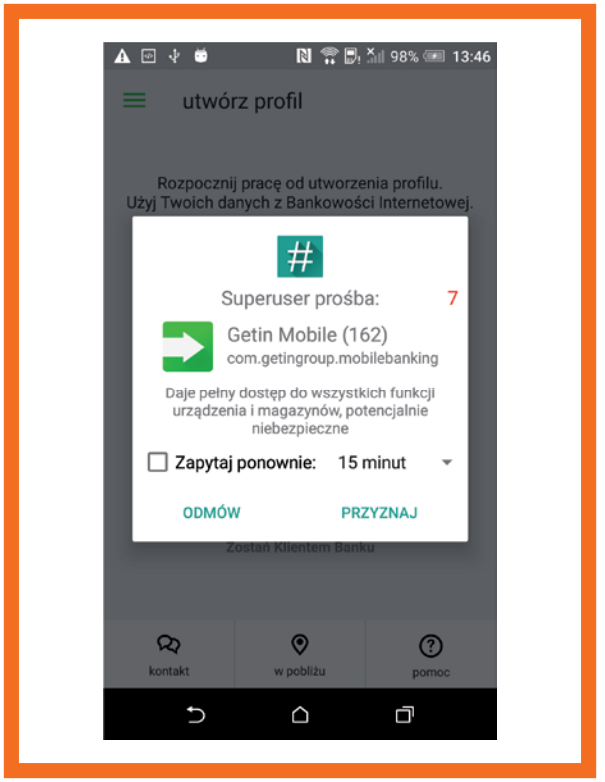
Tak więc:

- aplikacja **BGŻ BNP** po wyjściu z widoku mapy nie cofała subskrypcji lokalizacji GPS, wskutek czego bateria ulegała wyczerpaniu w kilka godzin
- aplikacja **BPH** korzystała z biblioteki do odczytywania numeru ze zdjęcia karty w wersji przestarzałej o ponad rok, wskutek czego funkcja na nowszych telefonach w ogóle nie działała
- starsza wersja aplikacji **BPH** nie potrafiła poinformować o wymuszonej aktualizacji, zamiast tego na ekran trafiał komunikat o błędzie połączenia z serwerem
- wymuszenie aktualizacji aplikacji **Idea Banku** można było ominąć przechodząc do sklepu Play i wracając klawiszem *wstecz*
- aplikacja **Smart Banku** miała problemy z rozmieszczeniem elementów na ekranie, przez co do logów trafiało 150 komunikatów na sekundę, a telefon mocno się grzał
- w przypadku problemów z połączeniem, aplikacja **mBanku** wyświetlała na ekranie anglojęzyczną treść wyjątku, niezrozumiałą dla przeciętnego użytkownika
- aplikacja **Getin Banku** trafiła do sklepu z niedziałającymi mapami
- choć apki **Millennium** i **BGŻ** blokowały połączenia sieciowe przy próbie użycia podstawionego certyfikatu, miały problem ze wznowieniem pracy po przełączeniu na drugą, bezpieczną sieć Wi-fi

Aplikacje **Pekao** i **Raiffeisen** (obie autorstwa tego samego wykonawcy) zawierały komunikaty diagnostyczne z licznymi literówkami, co w normalnym procesie produkcji oprogramowania byłoby wyłapane na etapie przeglądu kodu. Opisy wyjątków w rodzaju „You do something wrong mapper must contains item type for item %sat this place, Maybe You modify your adapter in illegal way !?" mogą oznaczać obniżone standardy pracy z kodem źródłowym.



Niezrozumiały komunikat błędu w aplikacji **mBanku**.



Na zrootowanych urządzeniach aplikacja **Getin Banku** próbuje bez potrzeby podnieść poziom uprzywilejowania.

Kod debugowy

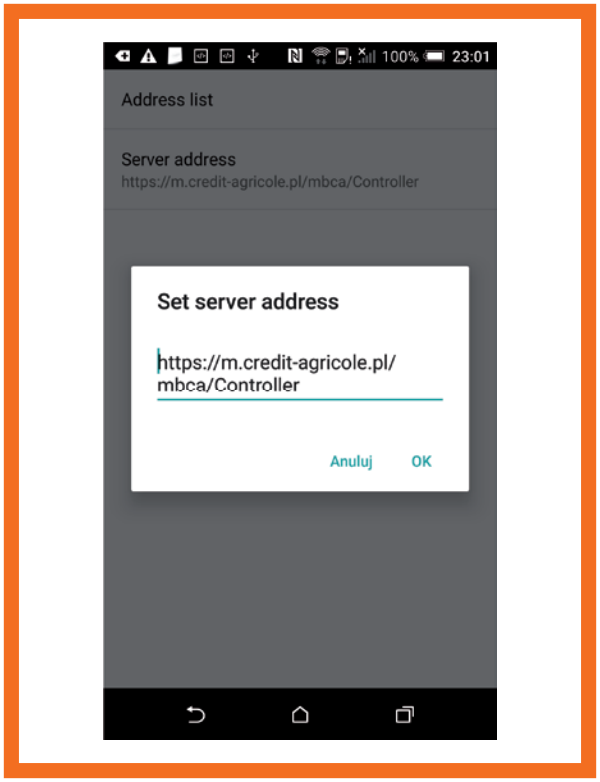
Notka techniczna

Debugowanie – "odpluskwianie", w slangu programistów usuwanie błędów oprogramowania. Kod debugowy to fragmenty programu pomocne w diagnostyce, nie mające zastosowania dla użytkownika końcowego. Dobrą praktyką jest, by kod debugowy nie trafiał do wersji produkcyjnej aplikacji.

Podczas pracy nad aplikacją programiści umieszczają w kodzie źródłowym różne pomocnicze konstrukcje ułatwiające tworzenie lub testowanie aplikacji. Przykładem może być mechanizm wyboru serwera, z którym aplikacja ma się komunikować. Zazwyczaj podczas pisania kodu korzysta się z serwerów testowych, na dalszym etapie ze środowisk do testów integracyjnych i akceptacyjnych, dopiero na końcu z serwerów produkcyjnych – taki przełącznik może być więc przydatny przy diagnozowaniu usterek aplikacji.

Wysłanie do sklepu wersji, która pozwala na przełączenie adresu serwera, potrafi mieć katastrofalne skutki. Atakujący może bowiem wymusić komunikację ze swoją maszyną, pośredniczyć w komunikacji z prawdziwym serwerem, a w krytycznym momencie np. podmienić numer konta i kwotę do przelewu. Taką właśnie możliwość odkryliśmy w aplikacji **Credit Agricole**, która w ciemno akceptowała nowy adres podany w podstawionym pliku XML.

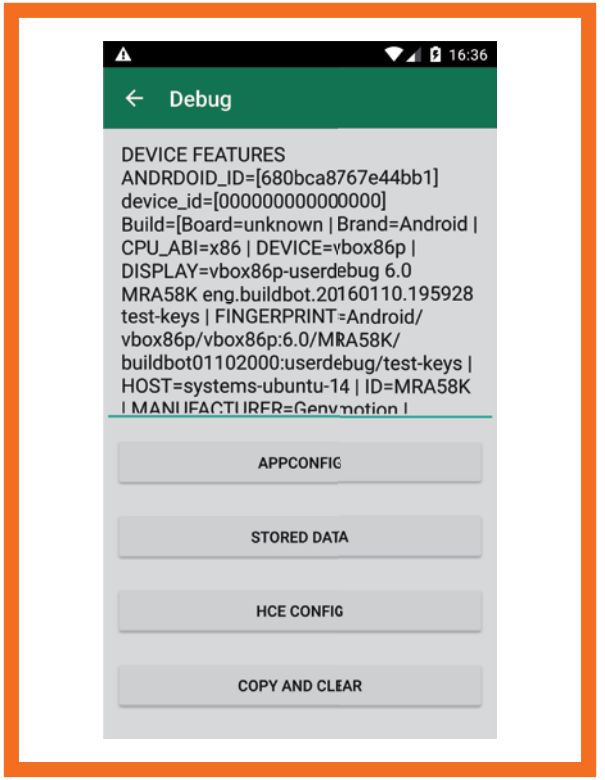
Teoretycznie, by móc utworzyć lub zmodyfikować ustawienia innej aplikacji, atakujący musi najpierw zyskać w atakowanym urządzeniu uprawnienia superużytkownika, jednak uważny czytelnik przypomni sobie opisane wcześniej podatności klawiatury Samsungów bądź wpadkę Alcatela z programem Qualcomm SystemAgent.



Debugowy kod w akcji – zmiana adresu serwera aplikacji **Credit Agricole**.

Dzięki tej ostatniej, każdy program był w stanie nadpisać pliki innych programów, co czyniło atak na aplikację **Credit Agricole** dziecinnie prostym.

Powyższy przykład pokazuje, dlaczego wszystkie mechanizmy pomocnicze muszą zostać usunięte z wersji produkcyjnych. Adresy środowisk testowych (obecne w aplikacjach **Pekao**, **BZ WBK**, **mBank**, **Raiffeisen**) mogą pomóc włamywaczowi w identyfikacji słabych punków infrastruktury. Dane do automatycznego uzupełniania formularzy zdradzą, ile znaków mają hasła testowe (ponownie **Credit Agricole**). Udało nam się też aktywować bibliotekę diagnostyczną Leak Canary będącą częścią aplikacji **Eurobanku**. Autorzy tej biblioteki piszą wprost, że nie powinna ona nigdy trafić do wersji produkcyjnej i podają prosty sposób, jak tego uniknąć. Aplikacja **BZ WBK** zawierała m.in. funkcje diagnostyczne płatności bezstykowych.



Funkcje debugowe aplikacji **BZ WBK**.

Nawet, jeśli kod debugowy nie sprowadza bezpośrednich zagrożeń, to może dostarczyć atakującemu informacji potrzebnych do wykonania ataku innymi ścieżkami.

Zawartość paczki z aplikacją

Plik APK z aplikacją androidową to nic innego, jak archiwum ZIP zawierające pliki rozmieszczone według szczegółowej specyfikacji. Przyjrzeliśmy się zawartości wszystkich aplikacji podejrzewając, że opisany wcześniej przypadek pliku z hasłami nie był unikalny i w zasobach znajdziemy jeszcze jakieś ciekawostki. Intuicja nas nie myliła.

Aplikacja **Banku Smart** zawierała plik *develop.properties.example* ze szczegółowymi opisem wszystkich opcji testowych, które można aktywować podczas kompilacji programu. Niezręcznie robi się, gdy spojrzymy na zamieszczoną tam listę kont testowych, gdzie Hans Kloss i James Blond sąsiadują z... Putinem.

Aplikacja **BPH** wiozła na pokładzie kilkanaście nieużywanych plików KML (znanych np. użytkownikom Google Earth). W aplikacji **Credit Agricole** można było znaleźć logi Jenkinsa (narzędzia do ciągłej integracji), co niesie kilka wskazówek na temat infrastruktury użytej do przygotowania paczki APK. W apce **ING** znaleźliśmy błędnie zintegrowaną bibliotekę Facebooka. Jeden z komponentów aplikacji **PKO BP** wskazuje jednoznacznie nazwisko programisty pracującego nad projektem (i potwierdza jednocześnie zlecanie prac podwykonawcy). W apce **Citi** znajdujemy obrazkowy wzór... chińskiego potwierdzenia przelewu.

Widać wyraźnie, w jak wielu przypadkach pliki APK zostały wrzucone do sklepu Play bez choćby pobieżnej kontroli ich zawartości. Dopisujemy ten punkt do zbiorczego zarzutu o niedostateczne testowanie aplikacji.

Odbiór aplikacji przez użytkowników

Użytkownicy najwyżej oceniają w sklepie Play aplikacje **PKO BP**, **BZ WBK**, **mBanku**, **ING** (starsza wersja), **Millennium**, **Orange Finanse** – wszystkie zebrały ocenę równą lub wyższą od 4.5 gwiazdki na 5 możliwych.

Spektakularną porażkę poniosła udostępniona w drugiej połowie 2016 roku aplikacja **Moje ING**, oceniona na 2.1 gwiazdki (przy dominującej ocenie jednogwiazdkowej). W komentarzach użytkownicy skarżą się na brak możliwości użycia programu na dwóch urządzeniach, brak dostępu do konta prywatnego i firmowego jednocześnie, powolność oraz usterki, które wychodzą na jaw przy słabszym połączeniu internetowym.

II. KONTAKT Z BANKAMI

Wszystkie usterki i podatności opisane w części pierwszej wykryliśmy z pozycji klienta, który założył rachunek, zainstalował aplikację mobilną, a potem zaczął sprawdzać, z jakiej klasy usługą ma do czynienia. Banki nie były informowane o rozpoczęciu badań, nie było więc mowy o prowadzeniu regularnych testów penetracyjnych.

Aby pozostać po jasnej stronie mocy, nie sprawdzaliśmy podatności serwerów i nie modyfikowaliśmy ruchu wychodzącego z aplikacji do serwera. Podobnie było z aplikacjami – sprawdzaliśmy ich odporność na potencjalne wektory ataku (np. modyfikację plików z danymi) oraz badaliśmy ich budowę, lecz nie modyfikowaliśmy kodu wykonywalnego i nie wpływaliśmy na stan pamięci działającego procesu.

Gdy mowa o publikacji wyników badań, trzymaliśmy się niepisanej zasady ***responsible disclosure*** (w wolnym tłumaczeniu ***odpowiedzialne ujawnienie***), czyli poinformowaliśmy banki o wynikach badań z odpowiednim wyprzedzeniem – od 4 do 10 tygodni (z jednym wyjątkiem, gdzie były to 3 tygodnie).

Ujawnianie podatności w praktyce

Notka techniczna

PGP (oraz darmowy wariant GPG) to narzędzia korzystające z tzw. kryptografii klucza publicznego, czyli – w bardzo dużym uproszczeniu – matematycznej metody utajniania wiadomości w taki sposób, że zaszyfrowana pierwszym z pary kluczy daje się odszyfrować jedynie przy użyciu drugiego klucza. Pierwszy można ogłosić publicznie (jest to plik z długim ciągiem znaków), drugi musi pozostać tajemnicą.

Jeśli więc bank ujawni klucz publiczny swojego działu bezpieczeństwa, możemy użyć go do zaszyfrowania wiadomości wysłanej na ogólny adres BOK i mieć pewność, że jedynie pracownik tego działu będzie w stanie zapoznać się z pierwotną treścią.

Choć tylko w kilku przypadkach mieliśmy do czynienia z podatnościami krytycznymi, postanowiliśmy wszystkie zgłoszenia potraktować jako informacje poufne. Zaczęliśmy od telefonów do wszystkich BOK-ów z prośbą o przełączenie do lub kontakt zwrotny z działu bezpieczeństwa. Same biura obsługi klientów (tak telefoniczne, jak i osiągalne e-mailowo lub przez formularz kontaktowy) traktowaliśmy jako stronę niezaufaną. Zdajemy sobie sprawę, że może to być ocena niesprawiedliwa, ale mając na względzie powszechny na tym odcinku outsourcing, dużą rotację, niskie płace i ogólne cięcie kosztów – zdecydowaliśmy się nie zdradzać tam informacji o słabościach bankowych systemów IT.

Metoda telefoniczna zawiodła na całej linii. Nie było ani jednego przypadku sukcesu. BOK-i są całkowicie odizolowane od części biznesowej i nie dysponują możliwością wyeskalowania zgłoszenia do osób odpowiedzialnych za bezpieczeństwo informatyczne. Personel starał się pomóc, jak umiał, ale ustne dyktowanie informacji o podatnościach jako zgłoszenia reklamacyjnego nie jest ścieżką dającą nadzieję na sukces.

W drugim podejściu zdecydowaliśmy się więc na przeszukanie stron internetowych pod kątem procedur zgłaszania usterek i podatności. Znowu pudło! Na żadnej stronie nie było o tym ani słowa. Później okazało się, że na międzynarodowej stronie www.ing.com można odnaleźć klucz PGP, ścieżkę postępowania dla badaczy i informacje o możliwych nagrodach za odpowiedzialne przekazanie informacji.

W trzecim kroku użyliśmy więc konwencjonalnego e-maila (lub formularza online, jeśli brakowało adresu e-mail, przy czym formularz **BGŻ** nie pozwolił wysłać wiadomości bez wyrażenia zgody na otrzymywanie spamu, ech). Oto treść wiadomości:

Chciałbym skontaktować się z działem bezpieczeństwa Państwa banku. Ponieważ nie znalazłem odpowiednich informacji na Państwa stronie internetowej, chciałbym spytać:

- *czy mają Państwo publiczną procedurę kontaktu w sprawie podatności Państwa witryny i aplikacji mobilnych na ataki z zewnątrz?*
- *czy deklarują Państwo czas reakcji, w którym Państwa specjaliści odpowiedzą na taki kontakt?*
- *czy dysponują Państwo publicznym kluczem PGP/GPG, który może posłużyć do poufnego kontaktu z działem bezpieczeństwa?*
- *czy prowadzą Państwo może program typu "bug bounty", w którym osoby zgłaszające podatności w odpowiedzialny sposób mogą zostać nagrodzone?*

Prosiłbym o odpowiedzi oraz wskazanie sposobu, w jaki informacje mogą zostać w sposób poufny przekazane do działu bezpieczeństwa.

Wiadomości tej treści zostały rozesłane w czwartek 30 czerwca około godziny 15:00. Wskazuję godzinę, bo jeszcze tego samego dnia odpowiedź (wraz z kluczami PGP) przysłały **Alior Bank** i **Milennium**. Dzień później odpowiedział **ING**, wskazując instrukcję na międzynarodowej witrynie korporacji. Czwartego lipca, więc już po weekendzie, odpowiedział **mBank**, jego klon **Orange Finanse** oraz **BZ WBK**. Tydzień po wysłaniu pytania otrzymaliśmy klucz publiczny banku **BPH**. Łącznie – mniej niż połowa banków.

Potem zrobiło się trudniej. I nieco dziwniej.

Bardzo proszę oczekiwać na odpowiedź w przedmiotowej sprawie

Poniżej nieprzypadkowo pominęliśmy nazwy banków. Nie chcemy, aby za brak procedur oberwali niewinni pracownicy niższego szczebla.

W jednym z przypadków odpowiedź zawierała rzemieślniczy sposób na poufną korespondencję: przesłanie na adres BOK-u zahasłowanego archiwum ZIP oraz wysłanie hasła SMS-em na podany numer.

Inny z banków zachował milczenie.

Jeszcze inny raz po raz obiecywał odpowiedź specjalisty zajmującego się kwestiami wymagającymi dalszych wyjaśnień. Po czym milczał.

Kolejny pomilczał, po czym zażyczył sobie *próbki celem weryfikacji zgłoszenia*.

Nadzwyczajne sytuacje wymagają nadzwyczajnych środków, więc jako ostatniej deski ratunku użyliśmy serwisu LinkedIn (mail do członka zarządu czyni cuda) oraz rzeczników prasowych (tu cuda następowały wolniej i mniej spektakularnie, ale koniec końców też się działy).

Ostatecznie wszystkie banki otrzymały wiadomość na jeden z kilku sposobów: na adres BOK po zaszyfrowaniu kluczem PGP, poprzez formularz online korporacyjnego systemu poczty lub – przy braku krytycznych podatności – na indywidualny adres e-mailowy pracownika banku, który prowadził sprawę.

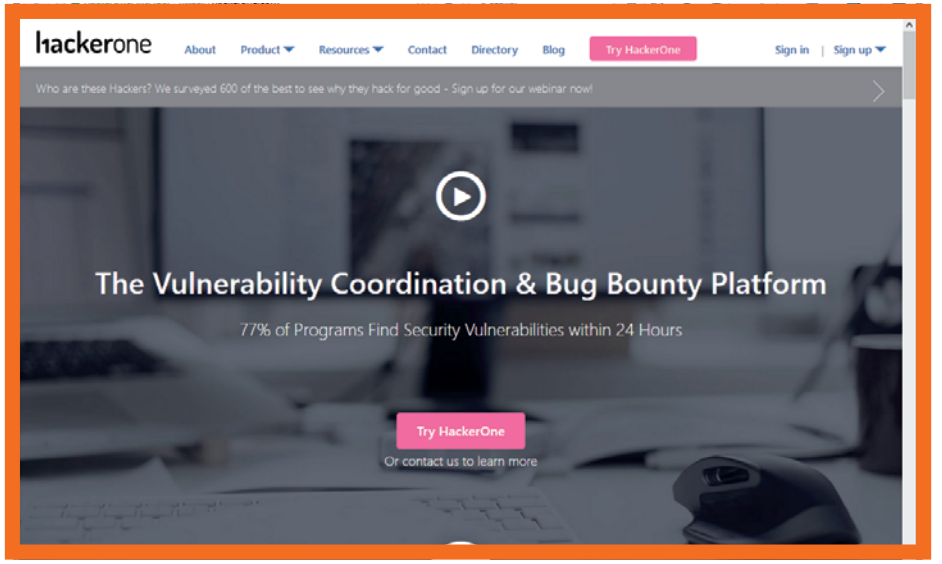
Drogie banki! Musicie usprawnić ten proces!

W Waszym najlepszym interesie jest ułatwienie zgłaszania problemów z bezpieczeństwem. Brak szybkiej ścieżki może się kiedyś boleśnie zemścić. Potrzebujecie procedur, które zapewnią reakcję w kwadrans, niezależnie od dnia tygodnia. Opublikujcie wskazówki na swoich stronach, ujawnijcie alarmowe telefony i adresy e-mailowe, zdefiniujcie ścieżkę pt. ***krytyczna podatność*** dla BOK-u.

Bug bounty

W wielu zagranicznych firmach z branży IT przyjął się zwyczaj dziękowania za odpowiedzialne zgłaszanie podatności – np. publikacja wyrazów wdzięczności na stronie WWW, nagrody rzeczowe, nagroda pieniężna (w przypadku Microsoftu czy Facebooka – nawet ponad 10 000 dolarów).

Spośród osiemnastu banków, którym przekazano informacje dotyczące przetestowanych aplikacji, jedynie **Bank Milennium** wystosował pisemne podziękowania, dołączając upominek w postaci klucza sprzętowego Yubico.



Hackerone – jeden z serwisów, które ułatwiają współpracę firm i instytucji z niezależnymi badaczami.

III. POLSKIE BANKI, AMERYKAŃSKIE USŁUGI

Crashlytics

Podczas testów zwróciliśmy uwagę na fakt, że aplikacje mBanku, Orange Finanse, IdeaBanku oraz Banku Smart korzystają z usługi Crashlytics, której właścicielem jest amerykańska firma Twitter.

Crashlytics to narzędzie do automatycznego raportowania problemów. Jeśli aplikacja *wywróci się*, szczegóły problemu zostaną przesłane na serwery Twittera. Programista będzie mógł zobaczyć w wygodnym panelu nie tylko stos wywołań funkcji, który doprowadził do awarii, ale także statystyki – ile razy wystąpił taki sam błąd, u ilu użytkowników, na jakich urządzeniach, przy jakim obciążeniu pamięci, itd., itp. Jest to świetne narzędzie, ale nas martwi istotny szczegół: POLSKA instytucja finansowa i AMERYKAŃSKI serwer.

Co można znaleźć w takim raporcie błędu? Pozornie niedużo, jednak diabeł tkwi w metadanych. Jeśli aplikacja zgłosi błąd w klasie MortgageLastStepSuccess.java (nazwa sugerująca przyznanie kredytu hipotecznego), Twitter tylko pozornie nie dowiaduje się niczego o użytkowniku telefonu. W praktyce może odnotować, by na ten numer IP wysyłać w przyszłości reklamy dekoratorów wnętrz i sprzętu AGD.

Niespodziankę mogą też sprawić biblioteki zewnętrzne, np. dwa lata temu programiści Google rozszerzyli informacje diagnostyczne w bibliotece GSON. Dawniej do raportu trafiłby mniej więcej komunikat *błąd przetwarzania napisu »100000CHF« na liczbę*, po zmianach ten sam błąd będzie skutkował wpisem *błąd przetwarzania napisu »100000CHF« na liczbę w węźle »Użytkownik › Kredyty › Hipoteczne[2] › Saldo«*. Bum, kolejny wyciek informacji o finansach użytkownika. Ten przykład pokazuje też, że jeśli nawet dziś wszystko jest w porządku, rutynowe podniesienie wersji komponentów zewnętrznych może to znienacka zmienić (nie oszukujmy się, nikt nie robi audytów zmian w bibliotekach).

Gdy po awarii użytkownik zostanie zapytany o zgodę na wysyłkę danych (**mBank** pyta, **Bank Smart** oraz **IdeaBank** nie), z treści pytania nadal nie wynika, że informacje nie trafią bezpośrednio do banku lecz na serwer w USA.

O wypowiedź na ten temat poprosiliśmy dra Pawła Litwińskiego, adwokata specjalizującego się w problematyce ochrony danych osobowych, nowych technologii i prawa telekomunikacyjnego:

Wypada zadać sobie pytanie o zgodność z prawem takiego, a nie innego działania bankowych aplikacji mobilnych korzystających z Crashlytics/Fabric.io. Tym bardziej, że kwestia zgodności z prawem przekazywania danych (osobowych) do USA była w ostatnich miesiącach szeroko komentowana (zob. wyrok z 6 października 2015 r. w sprawie Maksymiliana Schrems, C-362/14).

Przekazywanie danych osobowych do USA co do zasady jest zabronione. Żeby było legalne, podmioty uczestniczące w takim przekazywaniu muszą posłużyć się pewnymi specjalnie zaprojektowanymi na tą okazję instrumentami prawnymi, tzn.:

- *zawrzeć między sobą umowę opartą o tzw. klauzule modelowe, czyli wzór umowy zatwierdzony przez Komisję Europejską, albo*
- *działać wewnątrz jednej grupy kapitałowej w oparciu o zatwierdzone tzw. reguły korporacyjne.*

Odbiorca danych z USA może również obecnie przystąpić do programu Privacy Shield, do którego członków można przekazywać dane z państw UE.

Twitter, Inc. nie przystąpił do Privacy Shield (można to łatwo sprawdzić on-line). Nie jest z ***mBankiem*** *również w jednej grupie kapitałowej – pozostaje więc tylko umowa oparta o klauzule modelowe. Czy taka umowa została zawarta? Nie wiemy i w zasadzie nie mamy sposobu, by się tego dowiedzieć – natomiast, co trzeba podkreślić, jest to jedyny dostępny sposób legalizacji przekazywania danych osobowych do USA w opisywanym przypadku.*

Ale podkreślić warto jeszcze jedną kwestię – my tak naprawdę nie wiemy, czy jakieś dane osobowe są do USA przekazywane. I pytanie, jaką wiedzę w tym zakresie mają wspomniane banki? Pytanie kluczowe, ponieważ administrator danych osobowych, czyli bank w stosunku do danych swoich klientów, powinien dołożyć szczególnej staranności celem zapewnienia ich ochrony (art. 26 ust. 1 ustawy z 29 sierpnia 1997 r. o ochronie danych osobowych). Szczególna staranność wymaga, żeby każdy przypadek, w którym inny podmiot może uzyskać dostęp do danych osobowych, został dokładnie zidentyfikowany, a następnie albo opatrzony odpowiednią podstawą prawną, albo wyeliminowany. Jest to tym bardziej istotne, że informacje o finansach użytkownika aplikacji mobilnej objęte są tajemnicą bankową – a ta wprowadza jeszcze bardziej restrykcyjne zasady ich ujawniania, niż ustawa o ochronie danych osobowych.

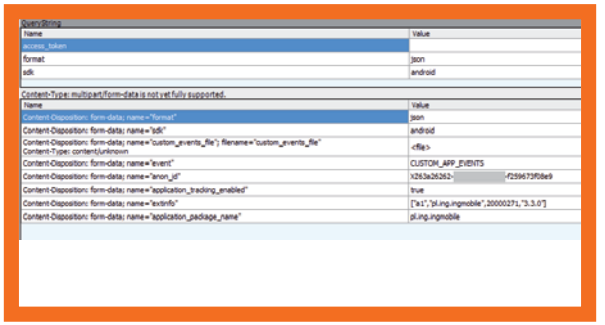
Jak więc – z punktu widzenia prawnika – powinien postąpić podmiot, który zamierza wykorzystywać aplikację mobilną, co do której istnieje ryzyko, że będzie przekazywać innym podmiotom dane osobowe użytkowników tej aplikacji? Po pierwsze, należy zidentyfikować każdy potencjalny przypadek takiego przekazywania danych. Po drugie, jeżeli nie zostanie zidentyfikowana podstawa prawna dla przekazywania danych, taki przypadek należy wyeliminować – tego wymaga ustawa o ochronie danych osobowych.

Facebook i inni

Czy polski bank ma prawo informować amerykańską korporację Facebook Inc. o tym, że klient banku właśnie rozpoczął pracę z aplikacją mobilną? Nie informując klienta, nie dając mu możliwości wyłączenia tej funkcji oraz nie umieszczając marki *Facebook* nigdzie w regulaminach? Dokładnie w taki sposób działa aplikacja **ING** (starsza z dwóch dostępnych), która posłusznie karmi popularnego *fejsa* danymi o tym, kto, skąd i kiedy rozpoczyna sesję z aplikacją.

W innych aplikacjach dostrzegliśmy kod Gemiusa (**mBank**, **Orange Finanse**), Adobe Marketing Cloud (**ING**, **Citi**), Webtrends (**Millennium**), SponsorPay/Fyber (**Bank Smart**), HockeyApp (**BPH**). Żeby było jasne – nie mielibyśmy nic przeciwko osadzeniu w aplikacji modułu raportującego, o ile tylko dane byłyby kierowane bezpośrednio do banku.

Może któryś z czytelników pokusi się o pełniejszą analizę prawną tego zagadnienia?



Dane wysyłane do Facebooka przy starcie aplikacji **ING**.

PODSUMOWANIE

Jak pokazaliśmy w I części niniejszego raportu, bezpieczeństwo i jakość wykonania aplikacji mobilnych polskich banków nie spełniają dzisiejszych standardów. Od instytucji finansowych można i należy oczekiwać więcej. Wraz ze wzrostem popularności bankowości mobilnej zwiększa się przecież liczba metod ataku na infrastrukturę, dane użytkowników i zgromadzone środki pieniężne.

W części II opisaliśmy problemy, na jakie trafiliśmy podczas zgłaszania bankom odnalezionych podatności. Tu do zmian na lepsze naprawdę nie trzeba wiele – wystarczy dodatkowy scenariusz do użycia w BOK, klucz PGP opublikowany na stronie, dedykowany e-mail do przyjmowania zgłoszeń.

Tematyka części III to jedynie skromny przyczynek do złożonego tematu, jakim jest obieg tzw. *danych wrażliwych* między wielkimi graczami współczesnego rynku IT. Banki powinny dołożyć wszelkich starań, by nie karmić systemów big data informacjami o swoich klientach.

Niniejszy raport nie wyczerpuje oczywiście tematu bezpieczeństwa aplikacji mobilnych polskich banków. Całkowicie pominęliśmy system iOS. Zignorowaliśmy obszar płatności bezstykowych NFC/HCE. Przeszliśmy do porządku nad ryzykiem związanym z użyciem wiadomości SMS jako metody uwierzytelniania. Konieczne są dalsze badania i ciągłe patrzenie bankom na ręce, aby oferowane produkty mobilne nie budziły tak wielu zastrzeżeń, jak obecnie.

PRZETESTOWANE APLIKACJE

- PKO Bank Polski (pl.pkobp.iko)
- Pekao SA (eu.eleader.mobilebanking.pekao)
- Bank Zachodni WBK (pl.bzwbk.bzwbk24)
- mBank (pl.mbank)
- ING Bank Śląski (pl.ing.ingmobile oraz pl.ing.mojeing)
- Getin Noble Bank (com.getingroup.mobilebanking)
- Bank Millennium (wit.android.bcpBankingApp.millenniumPL)
- Raiffeisen Polbank (eu.eleader.mobilebanking.raiffeisen)
- Citi Handlowy (com.konylabs.cbplpat)
- BGŻ BNP Paribas (com.comarch.mobile.banking.bnpparibas)
- BPH (pl.bph)
- Alior Bank (com.comarch.mobile)
- IdeaBank (pl.ideabank.mobilebanking)
- Eurobank (pl.eurobank)
- Credit Agricole (com.finanteq.finance.ca)
- T-Mobile Usługi Bankowe (alior.bankingapp.android)
- Orange Finanse (com.orangefinanse)
- Bank SMART (pl.fmbank.smart)

BRAK APLIKACJI MOBILNYCH

- Deutsche Bank Polska
- Santander Consumer Bank
- Volkswagen Bank
- Bank Pocztowy

BRAK MOBILNYCH APLIKACJI TRANSAKCYJNYCH

- SGB Bank
- BOŚ Bank

PGS SOFTWARE

O raporcie

Pomysł i realizacja: **Tomasz Zieliński,** tzielinski@pgs-soft.com
Redakcja: **Krzysztof Piskorski,** kpiskorski@pgs-soft.com
Skład: **Michał Cichoń,** micichon@pgs-soft.com