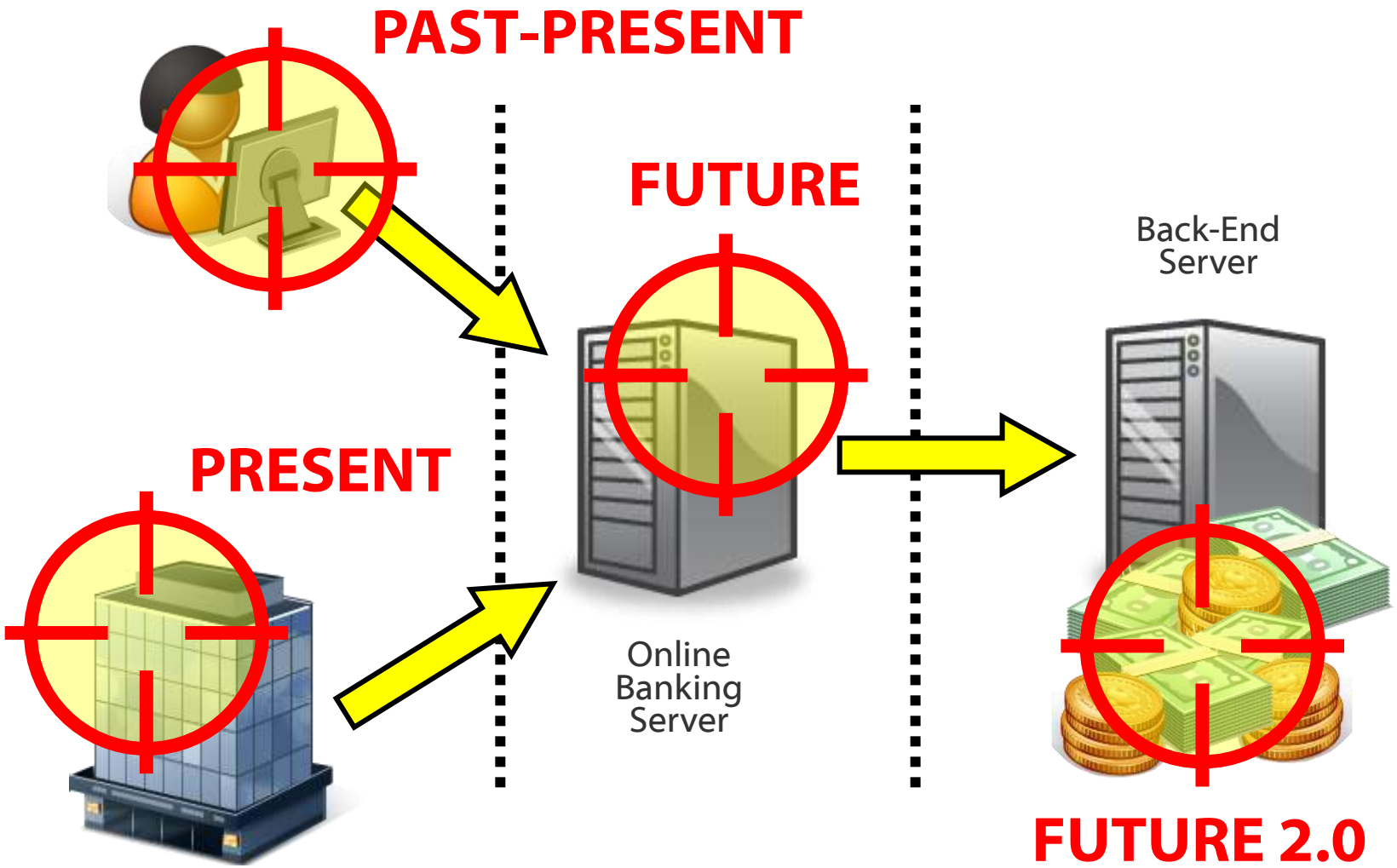
How to Rob an Online Bank
(and get away with it)

DeepSec 2011, Vienna

Mitja Kolsek
ACROS d.o.o.
mitja.kolsek@acrossecurity.com
www.acrossecurity.com

Evolution Of E-banking Attacks



acros

Attacks Against Individual Users

Goal: Identity Theft

Methods

Phishing, Fake security alerts

XSS, CSRF

Malware (man in the browser, extraction of certs and private keys)

Problems

User awareness

2-factor authentication

OOB transaction confirmations

Additional passwords/PINs

“Known good” target accounts

Attacks Against Corporate Users

Goal: Identity Theft

Methods & Problems

Same as with individual users

Advantages

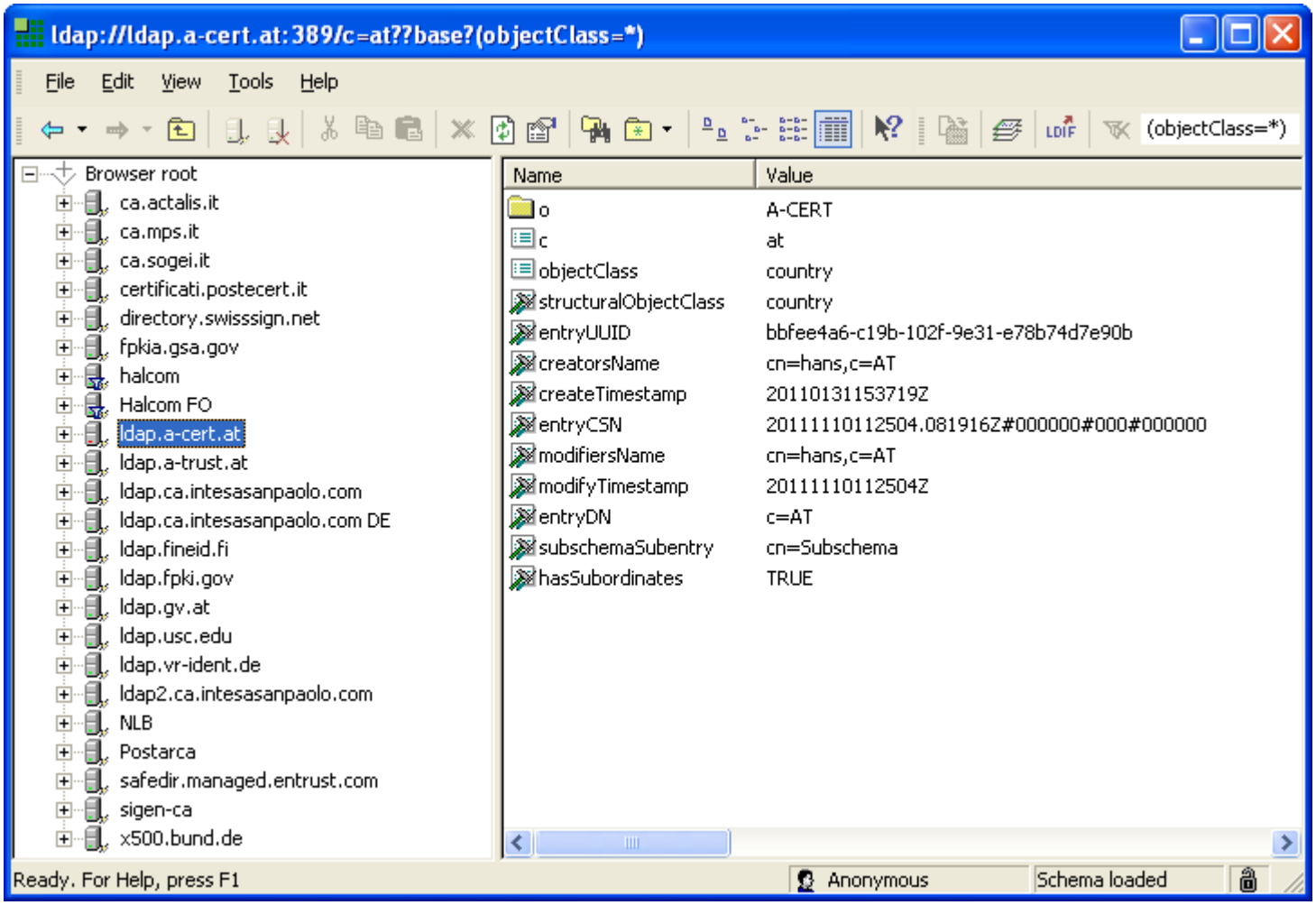
More money

Large transactions not unusual

Targets can be found in public certificate directories

acros

LDAP Explorer – Online Bank Robber's Google



acros

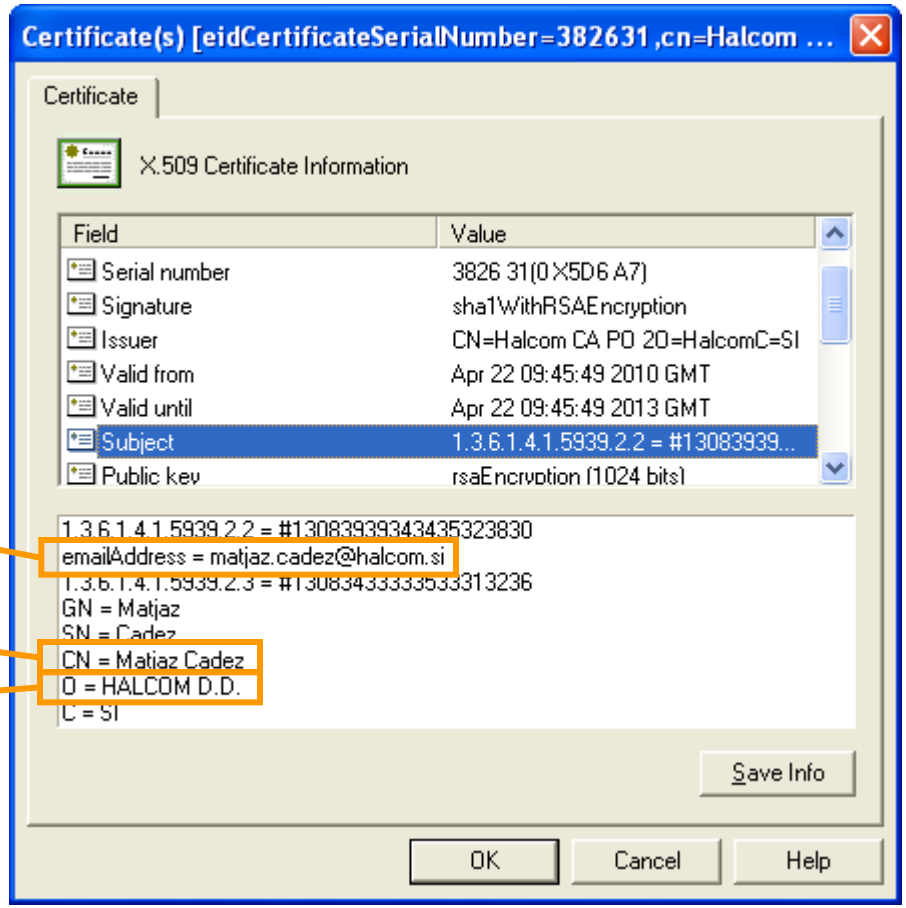
Example: Published Corporate Certificate

```
ldap://ldap.halcom.si:389/eidCertificateSerialNumber=382631
```

E-Mail Address

Personal Name

Company Name



Attacks Against Online Banking Servers

Goal: Exploiting Application Flaws

Methods

Hacking

Problems

Getting noticed while looking for flaws

Advantages

Unlimited amount of money
No user interaction (social engineering)
Possible creation of new money

Direct Resource Access

Direct Resource Access – URL Cleartext ID

https://bank/balance?uid=7728356
(my account balance data)

https://bank/balance?uid=772835**5**
(another user's account balance data)

Direct Resource Access – URL Base64 encoding

https://bank/balance?dWlkPTc3MjgzNTY=
(my account balance data)

Base64decode("dWlkPTc3MjgzNTY=")

"uid=7728356"

Base64encode("uid=772835**5**")

https://bank/balance?dWlkPTc3MjgzNTU=
(another user's account balance data)

Direct Resource Access – URL Encryption

/balance?Ko7hIGJJ2GqfhSZ9... (Base64)

/balance?AF86B301008AEF5... (Hex)

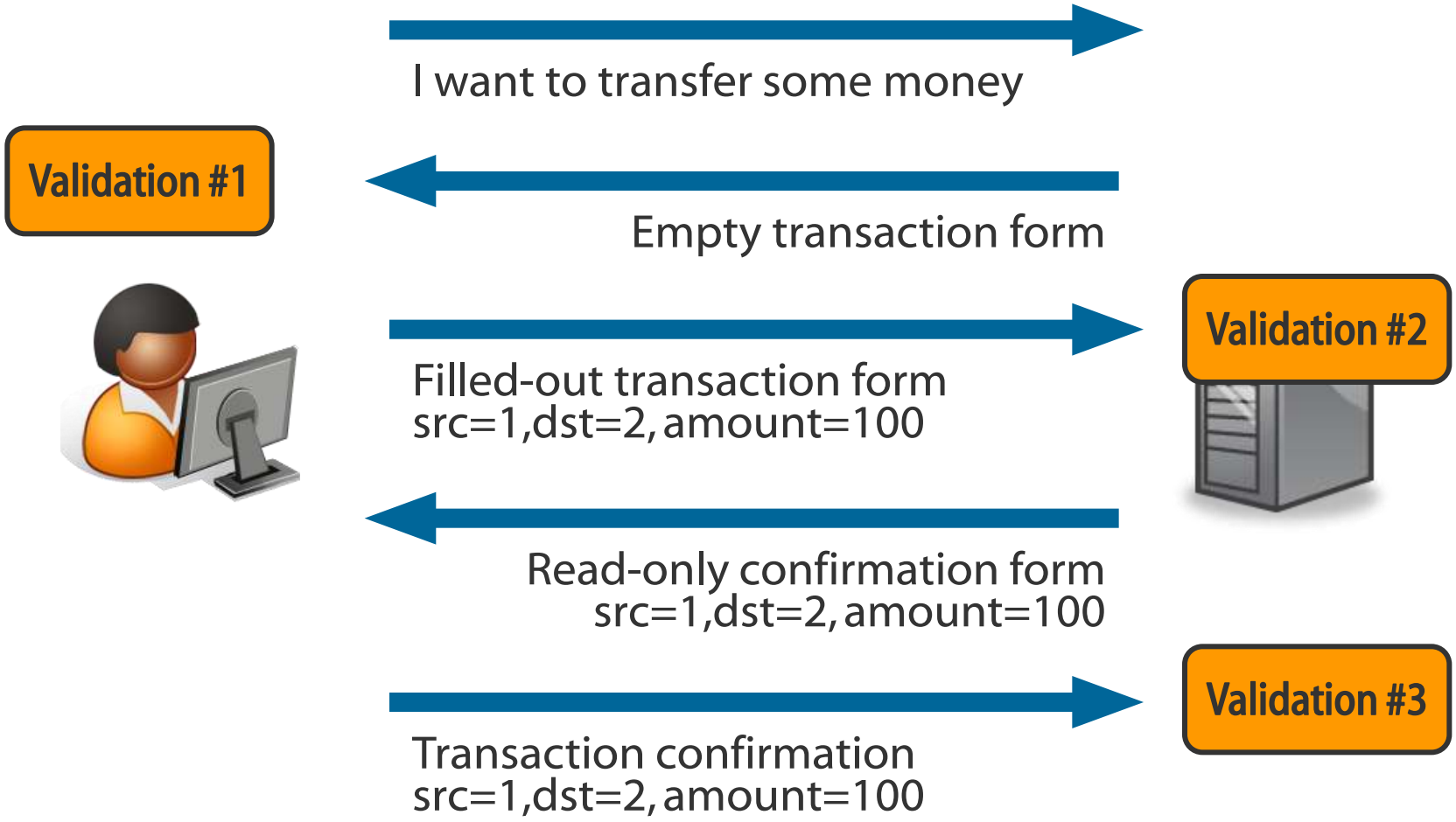
```
params = "uid=7728356"
params += "&salt=nobodywillguessthis"
params += "&rand=" + random()
enc_params = AES_encrypt(params, key)
path = "/balance?" + base64(enc_params)
```

Transferring Money From Other People's Accounts

/transfer? src=1 & dest=2 & amount=100
(from my account)

/transfer? src=**42** & dest=2 & amount=100
(from another user's account)

Transaction Creation Process



Negative Numbers

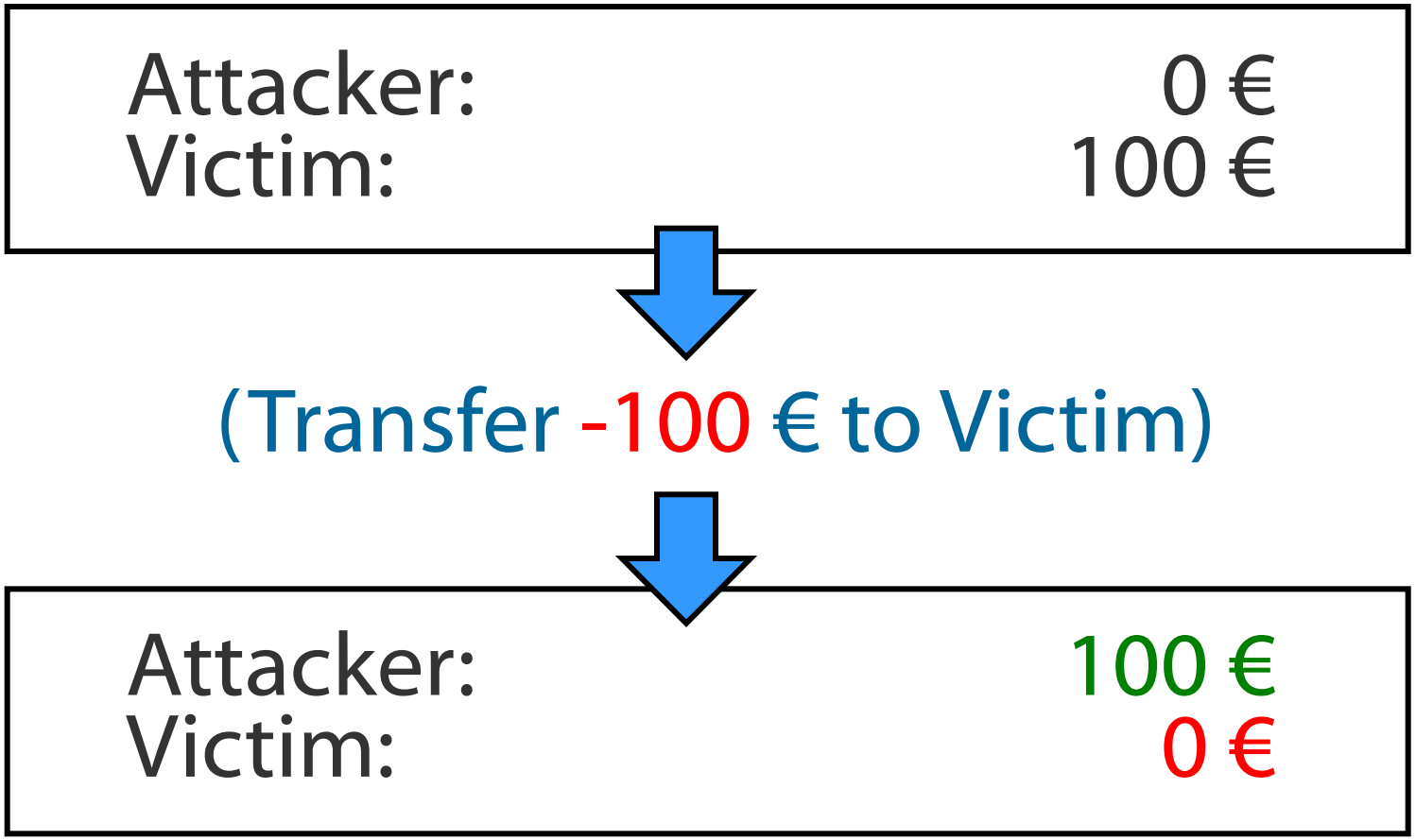
Negative Numbers – A Devastating Oversight

IF *RequestedAmount* > *DisposableAmount*
THEN ERROR();

IF 3.000 > 2.000
THEN ERROR(); // Error – Insufficient Funds

IF -100 > 2.000
THEN ERROR(); // No Error Here

"Here's minus hundred bucks for you"



Creating Money Out Of Thin Air

Normal Account:	0 €
Savings Account:	0 €

(Transfer -100 € to Savings Account)

Normal Account:	100 €
Savings Account:	0 €

Bypassing Limit Checks

Huge Overdraft

Account #1:	100 €
Account #2:	0 €

(Transfer 1.000.000 € from #1 to #2)

Account #1:	-999.900 €
Account #2:	1.000.000 €

HTTP Parameter Pollution

Luca Carettoni & Stefano di Paola
http://www.slideshare.net/Wisec/http-parameter-pollution-a-new-category-of-web-attacks

acros

User – Public Server – Back End Server

POST /transfer
source=1 & dest=2 & amount=100

JSP

```
source = request.getParameter("source") // 1
amount = request.getParameter("amount") // 100
IF NOT user_authorized_for(source) THEN ERROR()
IF disposable(source) < amount THEN ERROR()
Call BackEndTransaction(request)
```

POST /BackEndTransaction
source=1 & dest=2 & amount=100

PHP

```
source = $_POST["source"] // 1
dest = $_POST["dest"]     // 2
amount = $_POST["amount"] // 100
```

acros

HTTP Parameter Pollution – Source account

POST /transfer
source=1 & dest=2 & amount=100 & source=42

JSP

```
source = request.getParameter("source") // 1
amount = request.getParameter("amount") // 100
IF NOT user_authorized_for(source) THEN ERROR()
IF disposable(source) < amount THEN ERROR()
Call BackEndTransaction(request)
```

POST /BackEndTransaction
source=1 & dest=2 & amount=100 & source=42

PHP

```
source = $_POST["source"] // 42
dest = $_POST["dest"]     // 2
amount = $_POST["amount"] // 100
IF NOT user_authorized_for(source) THEN ERROR()
```

acros

HTTP Parameter Pollution – Transfer Amount

POST /transfer
source=1 & dest=2 & amount=100 & amount=100000

JSP

```
source = request.getParameter("source") // 1
amount = request.getParameter("amount") // 100
IF NOT user_authorized_for(source) THEN ERROR()
IF disposable(source) < amount THEN ERROR()
Call BackEndTransaction(request)
```

POST /BackEndTransaction
source=1 & dest=2 & amount=100 & amount=100000

PHP

```
source = $_POST["source"] // 1
dest = $_POST["dest"]     // 2
amount = $_POST["amount"] // 100000
IF NOT user_authorized_for(source) THEN ERROR()
```

acros

SQL Injection

SQL Injection – Data Theft

```
"SELECT rate FROM exch_rates
WHERE currency = '".$currency."'"
```

```
"SELECT rate FROM exch_rates
WHERE currency = '' UNION
SELECT balance FROM accounts
WHERE account_id = '887296'"
```

SQL Injection – Messing With Transactions

```
"BEGIN TRANSACTION"

"UPDATE accounts SET balance = 0
WHERE account_id = '".$acctid1."'"

"UPDATE accounts SET balance = 100
WHERE account_id = '".$acctid2."'"

"COMMIT TRANSACTION"
```

SQL Injection – Messing With Transactions

"BEGIN TRANSACTION"

"UPDATE accounts SET balance = 0
WHERE account_id = '123'"

"UPDATE accounts SET balance = 100
WHERE account_id = '456'"

"COMMIT TRANSACTION"

SQL Injection – Messing With Transactions

"BEGIN TRANSACTION"

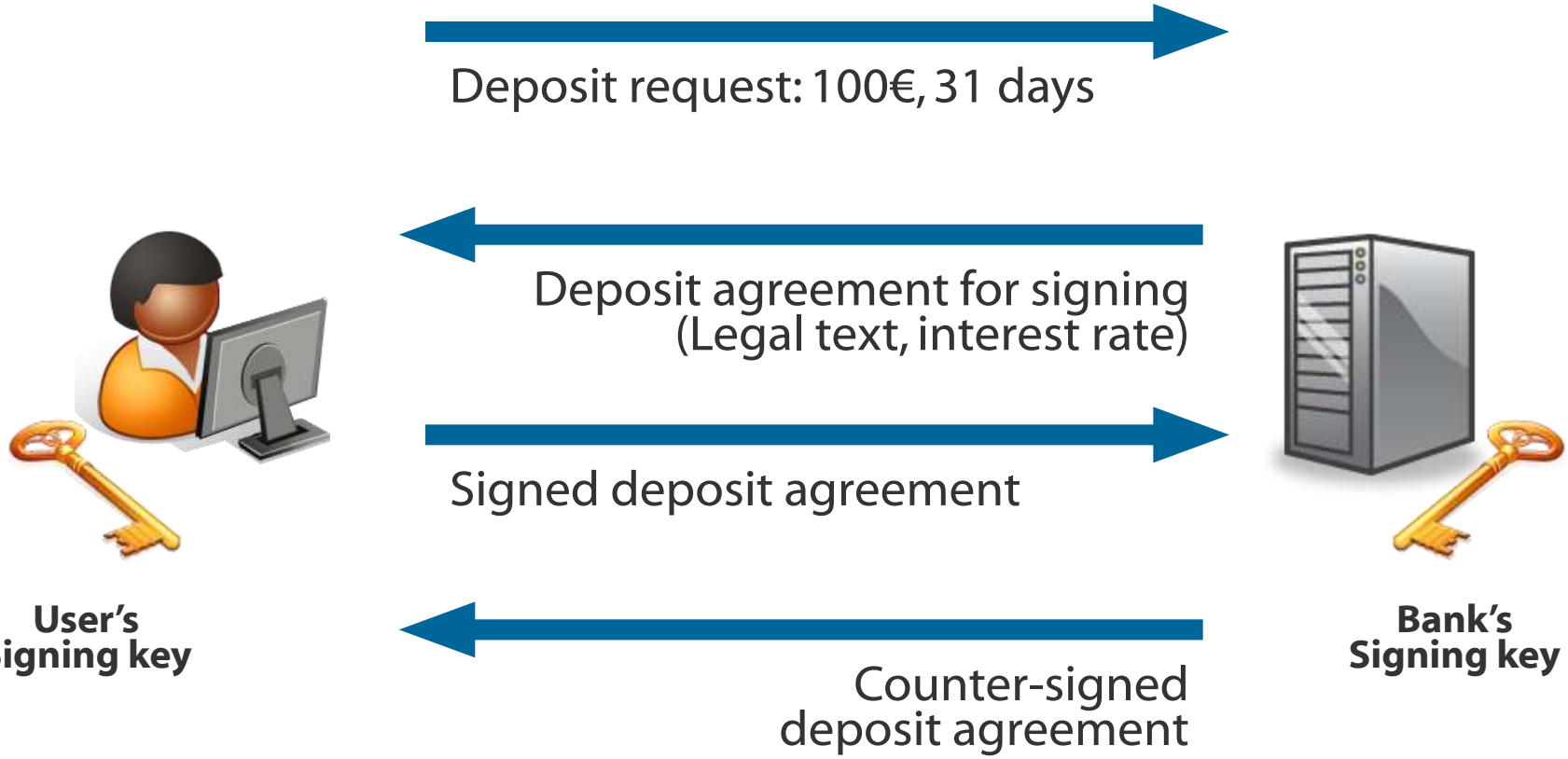
"UPDATE accounts SET balance = 0
WHERE account_id = '123'"

"UPDATE accounts SET balance = 100
WHERE account_id = '456' OR
account_id = '123'"

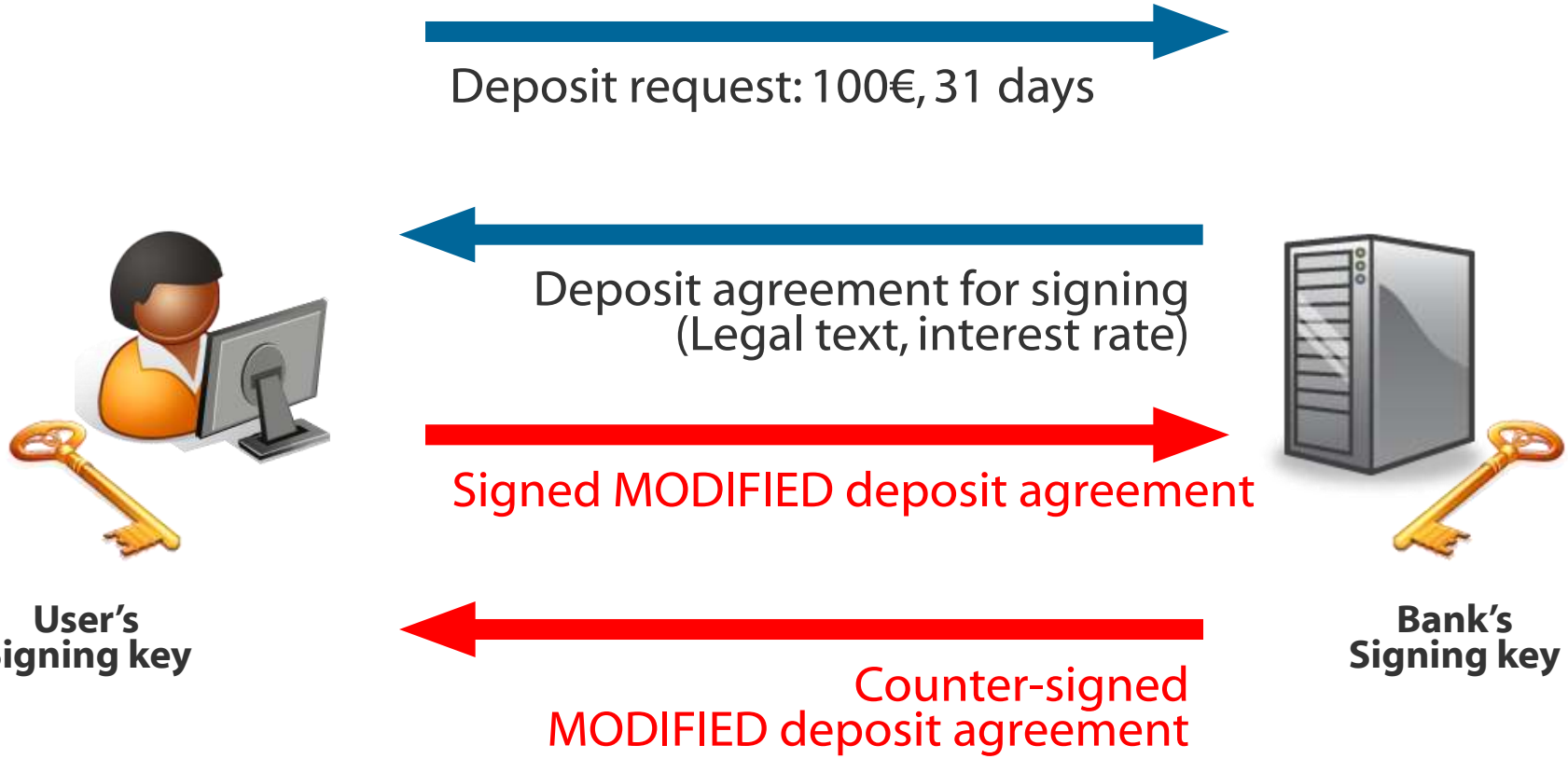
"COMMIT TRANSACTION"

Forging Bank's Digital Signatures

Automated Signing Of Deposit Agreement



Automated Signing Of Deposit Agreement



Server-Side Code Execution

Server-Side Code Execution

Examples

Java code injection (JBoss bug in 2010)

PHP code injection (eval, system, includes...)

Shell argument injection (command1&command2)

Buffer overflows

Impact

Change e-banking application code

Obtain database/WS credentials,

issue direct requests to DB or back-end WS

Getting Rich Without Breaking The Law

Rounding And Currency Exchange

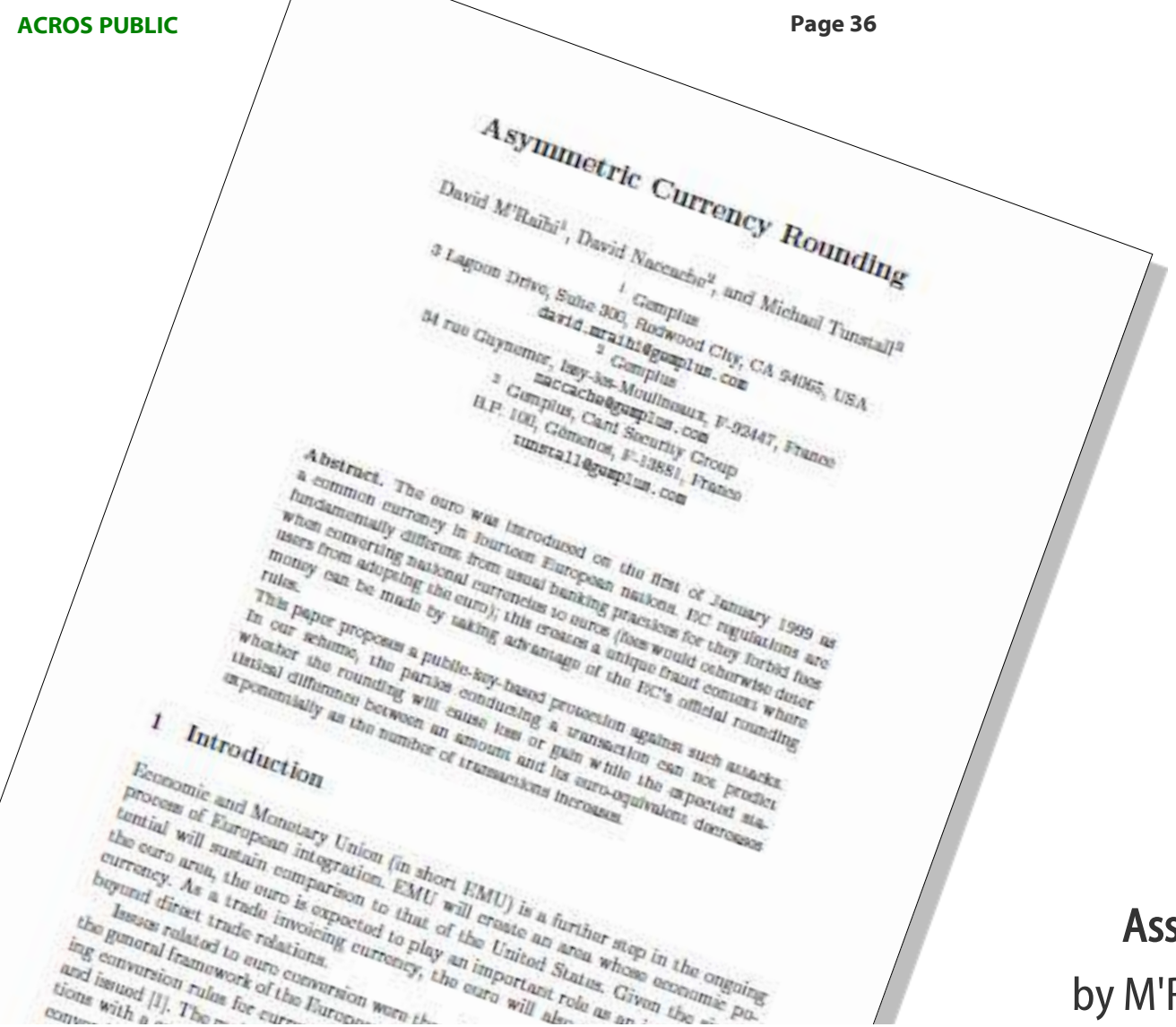
1 € ⇔ 1,364 $

0,01 € ⇒ 0,01 $

Loss : -0,00364 $ = -27%

0,01 € ⇐ 0,01 $

Profit : +0,00266 € = +36%



Asymmetric Currency Rounding

David M'Raïhi[1], David Naccache[2], and Michael Tunstall[3]

1 Gemplus
3 Lagoon Drive, Suite 300, Redwood City, CA 94065, USA
david.mraihi@gemplus.com
2 Gemplus
34 rue Guynemer, Issy-les-Moulineaux, F-92447, France
naccache@gemplus.com
3 Gemplus, Card Security Group
P.P. 100, Gémenos, F-13881, France
tunstall@gemplus.com

Abstract. The euro was introduced on the first of January 1999 as a common currency in fourteen European nations. EC regulations are fundamentally different from usual banking practices for they forbid fees when converting national currencies to euros (fees would otherwise deter users from adopting the euro); this creates a unique fraud context where money can be made by taking advantage of the EC's official rounding rules.
This paper proposes a public-key-based protection against such attacks. In our scheme, the parties conducting a transaction can not predict whether the rounding will cause loss or gain while the expected statistical difference between an amount and its euro-equivalent decreases exponentially as the number of transactions increases.

1 Introduction

Economic and Monetary Union (in short EMU) is a further step in the ongoing process of European integration. EMU will create an area whose economic potential will sustain comparison to that of the United States. Given the euro area, the euro is expected to play an important role as an [illegible] currency. As a trade invoicing currency, the euro will also [illegible] beyond direct trade relations.
Issues related to euro conversion were th[illegible] the general framework of the Europe[illegible] ing conversion rules for cur[illegible] and issued [1]. The [illegible] tions with a [illegible]

[illegible] was also prepared [illegible] to provide financial institu[illegible] ing all issues related to currency [illegible] great deal of attention was [illegible] different formulas, the constraint imposed by the [illegible] ducing conversion fees (a political requirement) opens [illegible] fraud strategies.

KNOWN
AT LEAST
SINCE

2001

Assymetric Currency Rounding

by M'Raïhi, Naccache and Tunstall

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.91.8055&rep=rep1&type=pdf

Currency Rounding Attack: Algorithm

```
1:  Convert 100€ to $
    // We have 136,40$

2:  for i = 1 to 13640
        Convert 0,01$ to 0,01€
    // Now we have 136,40€

3:  goto 1
```

Currency Rounding Attacks

The Speed Of Getting Rich

Assume: 10 exchanges / second

1 day = 86.400 seconds

Daily profit: 2.300 €

Monthly profit: ~ 70.000 €

Improvements

Optimal exchange rate (getting as close to 0,005 as possible)

Corporate banking: packets (1000s of exchanges in one packet)

Does it really work?

My personal e-banking: YES

My company's corporate e-banking: YES

Countermeasure

Limit minimum amount to 1 whole unit

Getting Away With It

Getting Away With It

Avoiding Detection

While searching for vulnerabilities

While performing the attack

Solution: *"User in the middle"* – hiding behind a user

Breaking The Money Trail

Money Mules, Western Union

BitCoin, WebMoney, Liberty Reserve, ...

Chaining multiple *"users in the middle"* in different countries

Perfect Crime: Print New Money, Don't Let Anyone Know

If nobody loses, nobody knows

DBA attack on back-end database, create fake transaction history

Attacker's Best Friends

"This would not work on our system."

"Our back-end validation would not allow this"

"Our employees would quickly notice this attack"

"This would not work with large sums of money"

"SQL injection is not possible because..."

New functionalities: automated deposits, loans, investment portfolio management, ...

Mitja Kolsek

ACROS d.o.o.
www.acrossecurity.com
mitja.kolsek@acrossecurity.com

Twitter: @acrossecurity

Thanks: Mikko H. Hypponen, René Pfeiffer, Claudio Criscione, Stefan Ortloff and Candi Carrera for help with gathering information on national digital certificate usage

acros