LET'S HACK ROCK

COURTNEY COOKSEY

Software Engineer

The Crossing EPC

Columbia, MO

Enid, OK

AGENDA

1. Rock Security 101
2. Person Merge Attack
3. Account Takeover Attack
4. Pitfalls with Impersonation Tokens
5. SQL Injection Attack
6. How You can Protect Your System

ROCK SECURITY 101

THERE ARE TWO MAIN ENTITIES INVOLVED IN AUTHENTICATION

Person Entity

Rules for what someone can access is tied to their person record

User Login Entity

User logins are used to verify that a specific person record belongs to you

ACCOUNT PROTECTION LEVELS

1

Low

No risk items

2

Medium

Has login account

3

High

Active Scheduled Financial Transaction

Saved Payment Account

In a Security Role Marked w/ High Elevated Security

4

Extreme

In a Security Role Marked w/ Extreme Elevated Security

SECURITY SETTINGS FOR ACCOUNT PROTECTION LEVELS

Disable Duplicate Checking for the Following Protection Profiles

- [ ] Low
- [x] Medium
- [x] High
- [x] Extreme

Allow Merges of Account Protection Profile - High

RSR - Data Integrity Worker

Disable Usage of Personal Tokens for the Following Protection Profiles

- [ ] Low
- [ ] Medium
- [ ] High
- [x] Extreme

PERSON MERGE ATTACK

ALL SYSTEMS VULNERABLE

TIMELINE OF ATTACK

SELECTING A TARGET

- Ideal target is a millennial or active on social media
- Someone with a public email or who's email we can obtain
- Someone we can easily obtain more data on via Google

Kristin Jeffries

Creative Arts Co-Director

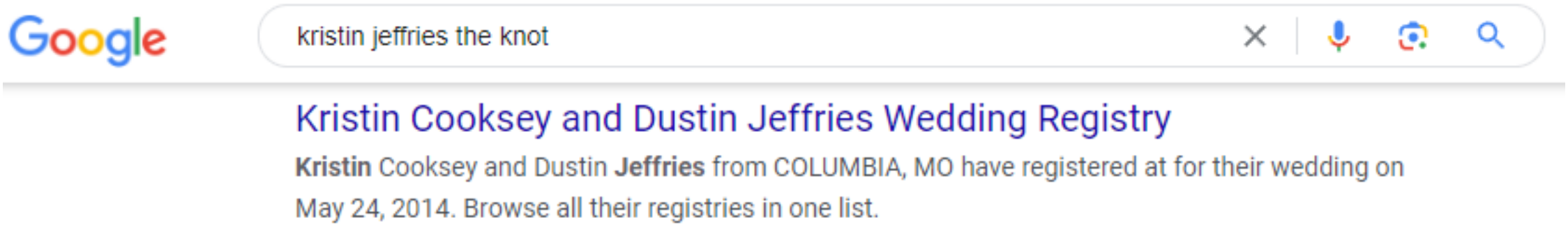
kristin@thecrossingchurch.com

Kristin was born and raised in Columbia. She began attending The Crossing as a high school student and became involved with the preschool and elementary large group teams while studying theatre at Stephens College. Through many years of serving in Crossing Kids, including three years as an intern, it became clear that Kristin had a heart for children's ministry. After marrying her husband, Dustin, in May of 2014, she left the world of theatre and dance to join the Crossing Kids team full-time in January of 2015.

As the Early Childhood Programming Director, Kristin oversees the preschool and little ones' teaching and worship teams, assists with family events, and creates early childhood curriculum. She loves leading worship and being involved in family events.

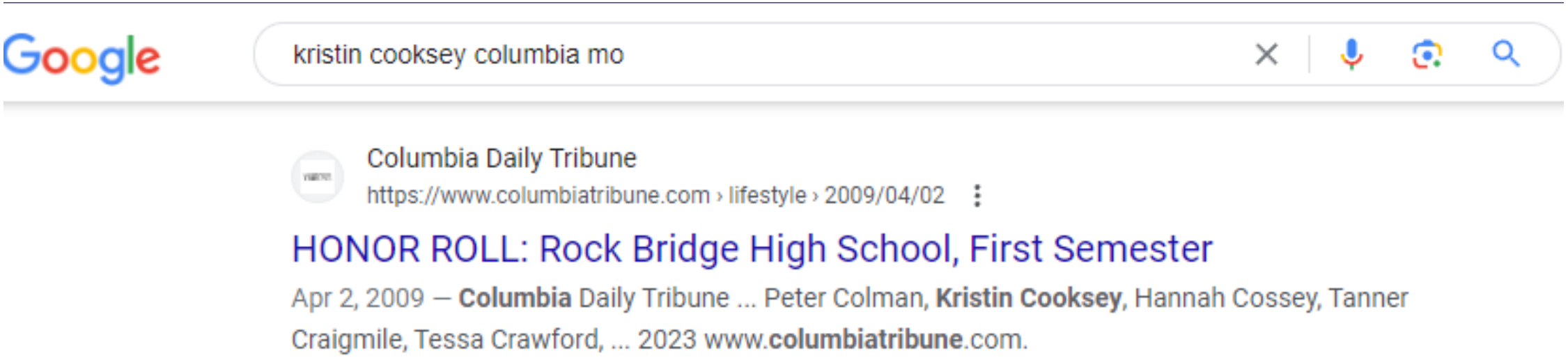
LOOK FOR A MAIDEN NAME

We have her current name, her husband's name, and the month and year they were married. With a few checks of the most popular wedding websites we get a hit

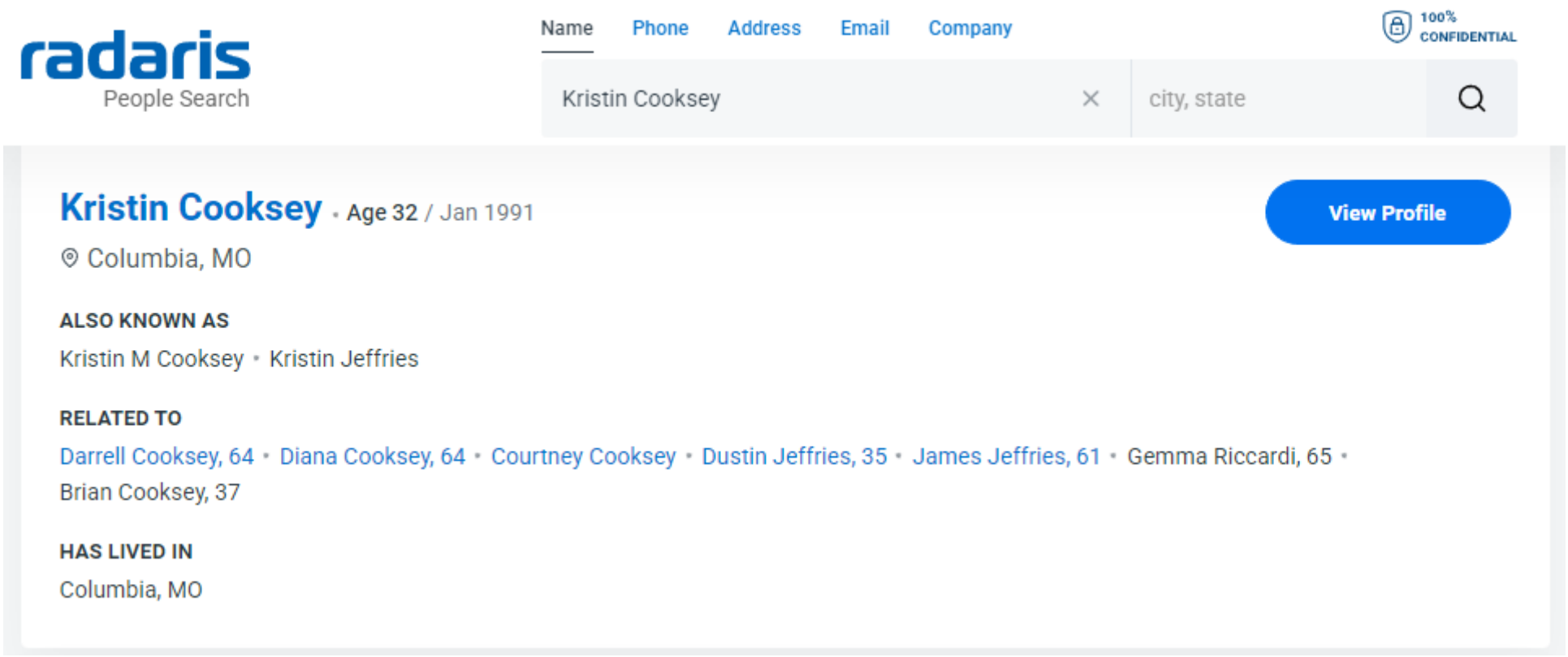


LOOK FOR DATE OF BIRTH INFORMATION

Armed with her maiden name and hometown we can look for other information about her. We get a hit for a high school honor roll for both 2008 and 2009 which narrows down her birth year to being within 1990-1992

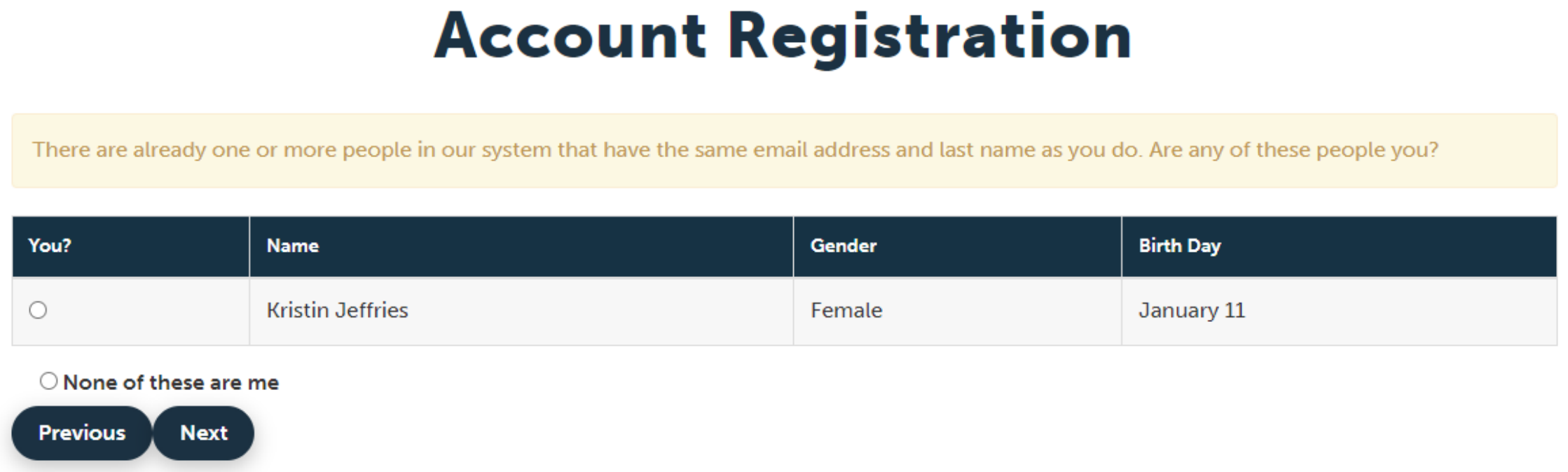


THE INTERNET IS A WEALTH OF PERSONAL INFORMATION



START TO CREATE A DUPLICATE ACCOUNT

We know her name, email, and two parts of her birth date. Let's see what happens when we create a new account in Rock with this information.



REGISTER OUR FAKE ACCOUNT

Account Registration

New Account

Username *
kmjeffries

The selected username is available.

Password *
••••••••••••••••••••

Confirm Password *
••••••••••••••••••••

Your Information

First Name *
Kristin

Last Name *
Jeffries

Email *
kristin.jeffries11@gmail.com

Gender
Female

Birthday *
Jan / 1 / 1991

UPDATE EMAIL OF FAKE ACCOUNT TO STAFF EMAIL

My Account

Photo

Personal Information

Title

First Name •
Kristin

Last Name •
Jeffries

Nick Name
Kristin

Suffix

Birthday
Jan / 1 / 1991

Gender •
Male Female

Contact Information

Email Address •
kristin@thecrossingchurch.com

Account Info

Change Password

Giving

You currently have no active profiles.

Manage View History

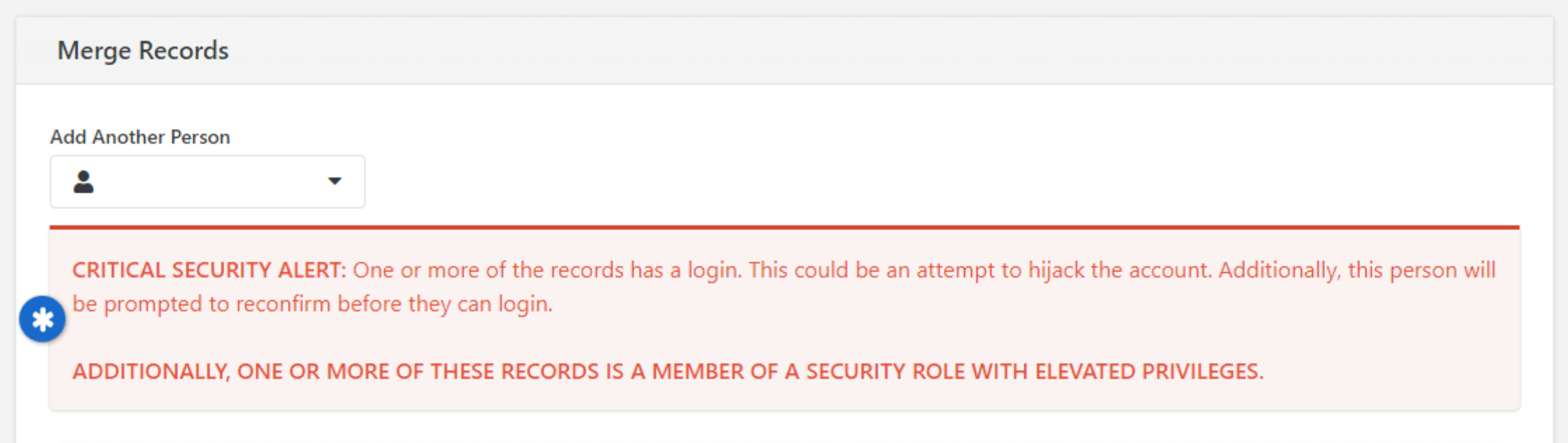
Give Now

Groups

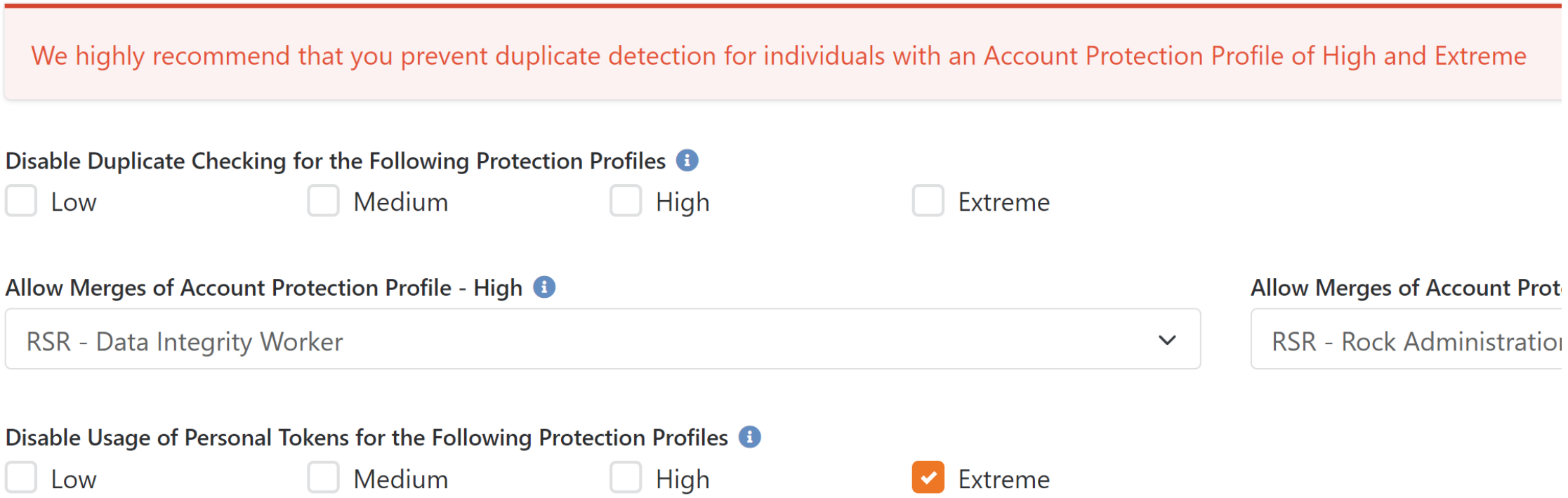
NOW WE WAIT...

THE MERGE ALERT IS NOT NECESSARILY HELPFUL

We are merging a staff member's record, it is expected that they have a login associated with their record



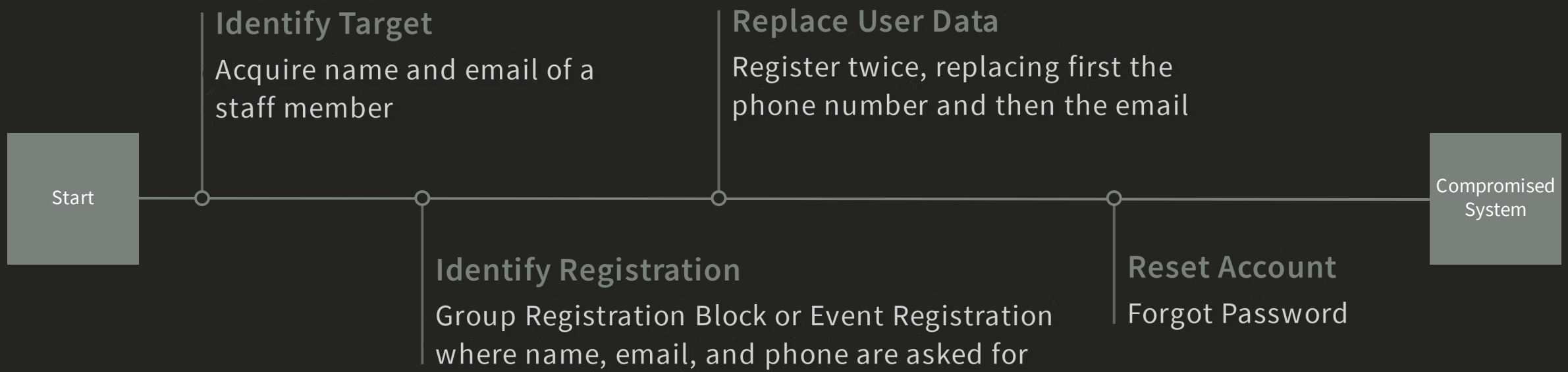
VULNERABILITIES WHEN DUPLICATE CHECKING IS NOT DISABLED



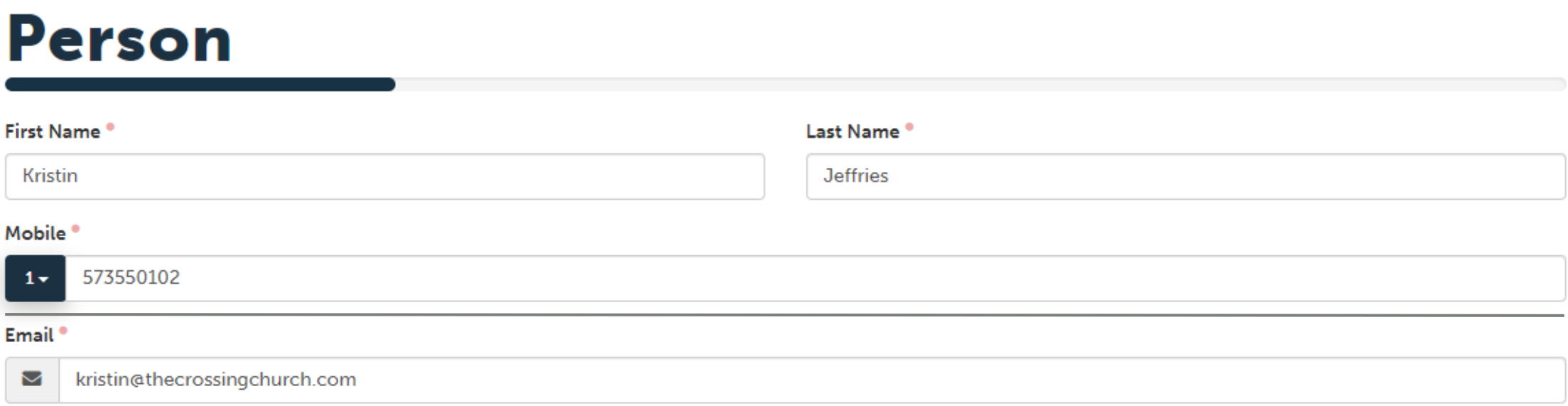
ACCOUNT TAKEOVER ATTACK

SYSTEMS WITHOUT DUPLICATE CHECKING DISABLED VULNERABLE

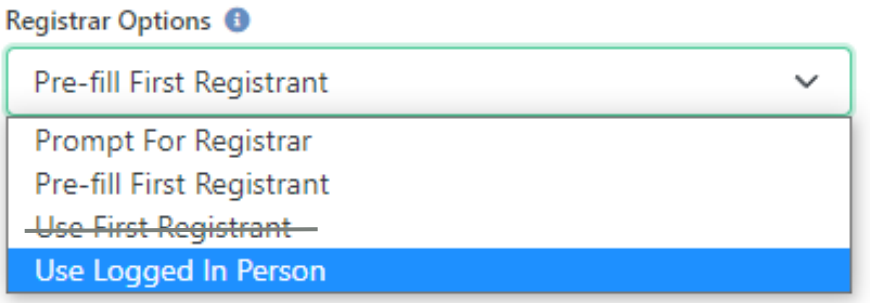
TIMELINE OF ATTACK

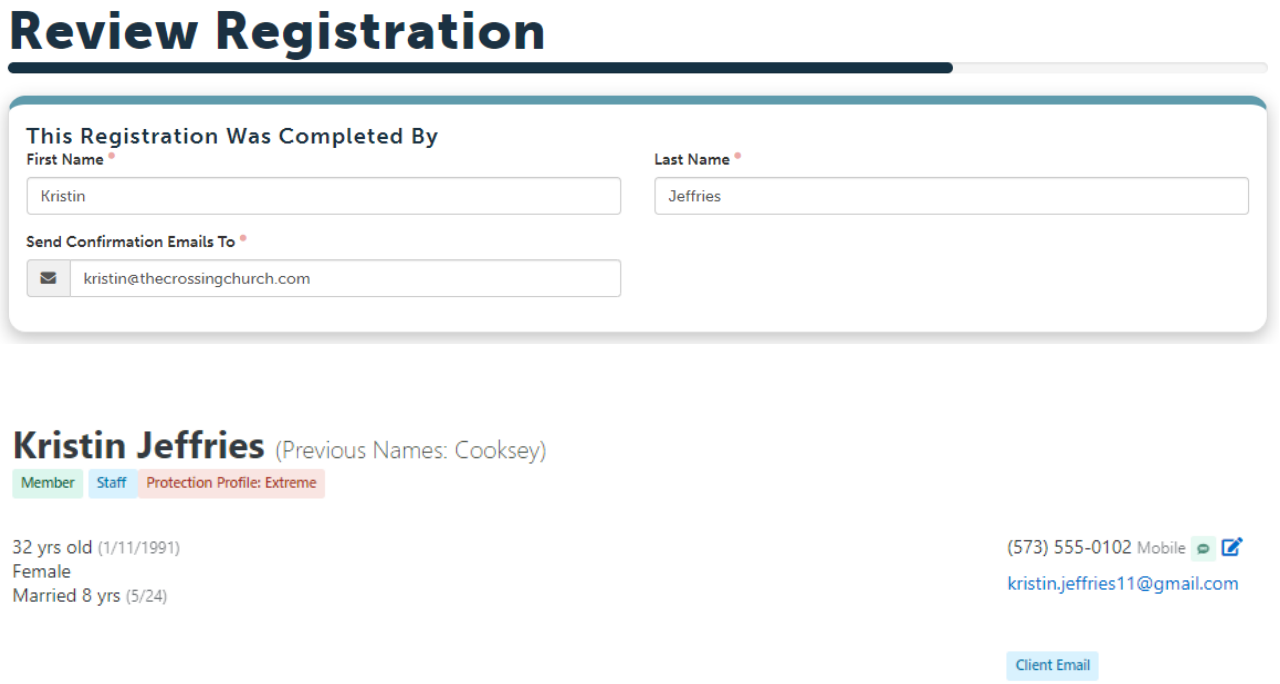


FIND AN EVENT OR GROUP REGISTRATION THAT ASKS FOR NAME, PHONE, AND EMAIL



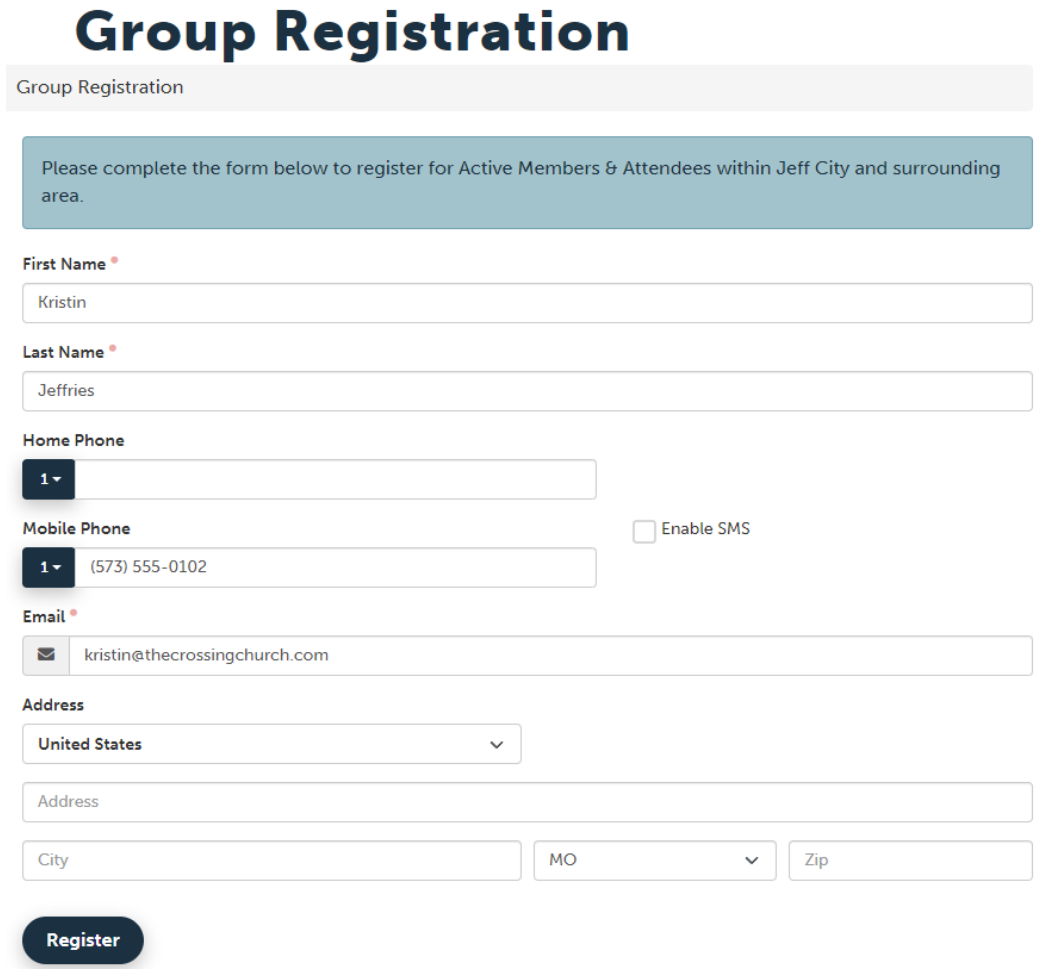
USING ANY REGISTRAR OPTION BESIDES USE FIRST REGISTRANT LETS US CHANGE ALL INFO





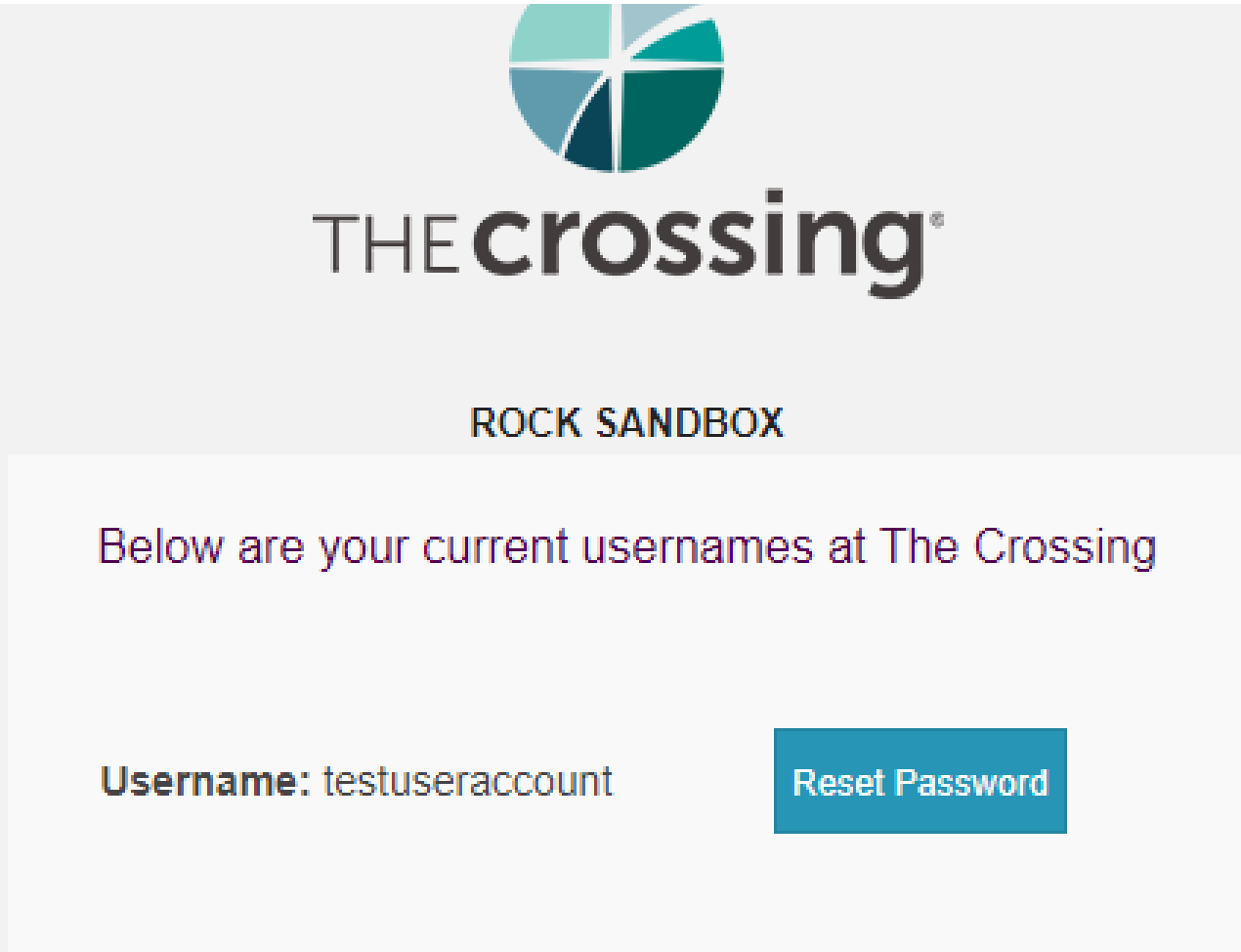
WE CAN DO THE SAME THING WITH GROUP REGISTRATION

- Register with correct name and email, new phone number
- Refresh Page
- Register with correct name and new phone number and email
- Person record in Rock is updated with new contact info



ROCK WILL SEND THE ATTACKER AN EMAIL TO RESET THE USER LOGIN

From here we have however much time until our target tries to log in before someone notices something is wrong.



PITFALLS WITH IMPERSONATION TOKENS

PARAMETERS FOR THE PERSON TOKEN CREATE LAVA FILTER

- Minutes
  How long should the token be valid?
- Max Usage
  How many times can you use it?
- Page Id
  What page should it work for?

THESE PARAMETERS ARE USED WHEN ESTABLISHING NEW SESSIONS

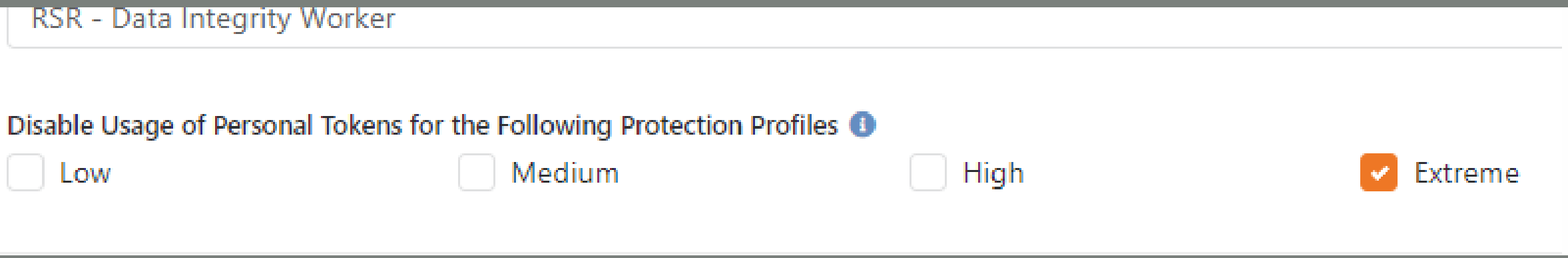
```
{{ Person | PersonTokenCreate:30,null,null }}
```

https://rock.mychurch.com?rckipid=ACDC123

30 minutes is the timeframe I can use that link to create a new session

A previously established session will not end after 30 minutes

LET'S ASSUME WE HAVE TOKENS DISABLED FOR EXTREME PROFILES



ISSUES WITH DISABLING IMPERSONATION

Group Attendance Reminders

```
Please remember to enter attendance for your group meeting.
{{ 'Global' | Attribute:'PublicApplicationRoot' }}page/368?
{{ Person.ImpersonationParameter }}&GroupId={{ Group.Id
}}&Occurrence={{ Occurrence | Date:'yyyy-MM-
ddTHH\%3amm\%3ass' }}
```

https://rock.mychurch.com/page/368?rckipid=TokenProhibited

FIX BAD USER EXPERIENCE BY CONDITIONALLY ADDING TOKENS

```
{% if Person.AccountProtectionProfile != 'Extreme' %}{%
endif %}
```

FIX BAD USER EXPERIENCE BY CONDITIONALLY ADDING TOKENS

```
{% if Person.AccountProtectionProfile != 'Extreme' %}{%
endif %}
```

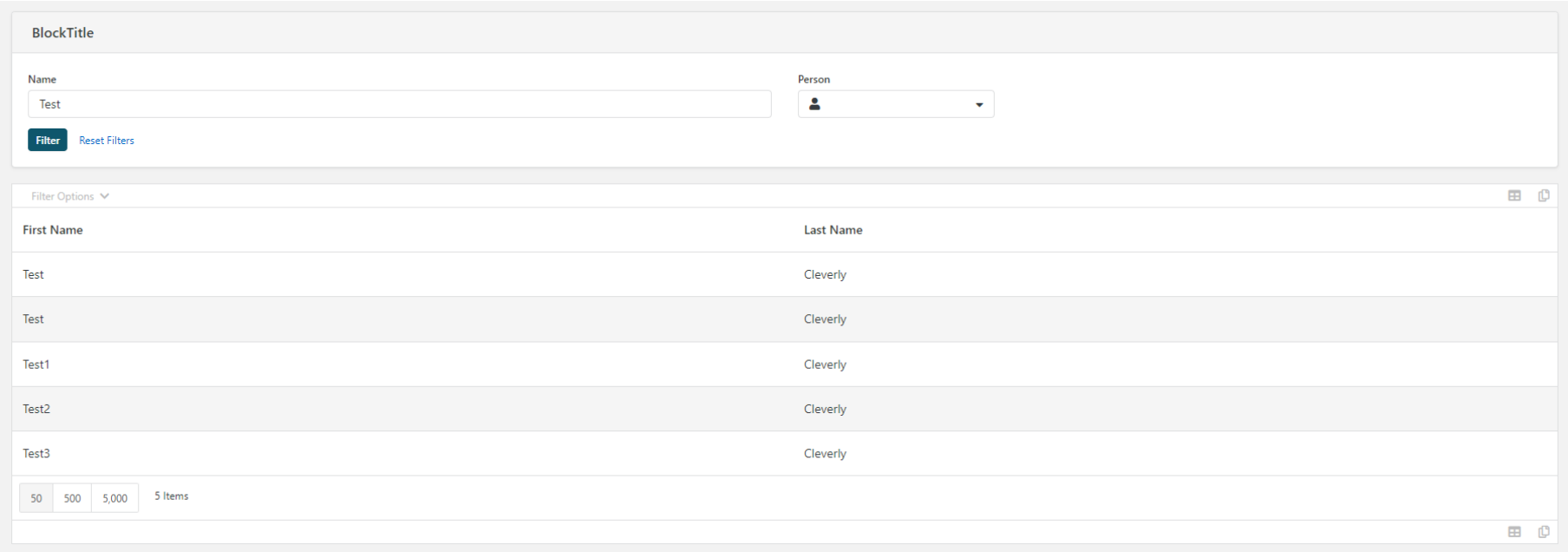
```
Please remember to enter attendance for your group meeting.
{{ 'Global' | Attribute:'PublicApplicationRoot' }}page/368?
{% if Person.AccountProtectionProfile != 'Extreme' %}{{
Person.ImpersonationParameter }}{% endif %}&GroupId={{
Group.Id }}&Occurrence={{ Occurrence | Date:'yyyy-MM-
ddTHH\%3amm\%3ass' }}
```

SQL INJECTION ATTACK

ALL SYSTEMS VULNERABLE

A BASIC DYNAMIC DATA BLOCK

```
{% assign searchOne = 'Global' | PageParameter:'Name' %}
SELECT TOP 10 FirstName, LastName FROM Person WHERE FirstName LIKE '%{{searchOne}}%'
```



A SIMPLE TABLE IN OUR DATABASE

SQL Command

SQL Text

```
1  SELECT * FROM TestTable
```

Selection Query?

No | Yes

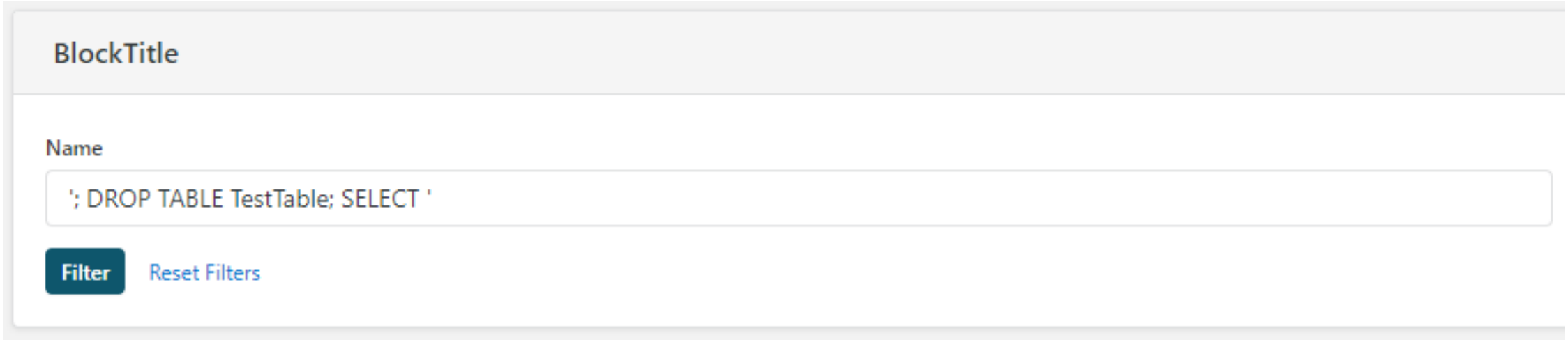
Run

column _1	column _2
1	1
2	2

50 | 500 | 5,000

2 Items

PERFORMING A SQL INJECTION ATTACK



Name='; DROP TABLE TestTable; SELECT '

OUTPUT OF EXECUTING THIS ATTACK

First Name	Last Name
Taylor	
DaeJera	Aaron
Joy	Aaron
Ann	Abare
Tom	Abare
Ashley	Abast
Jax	Abbadessa
Bella	Abbell
Mimi	Abbet
Amanda	Abbey

50 500 5,000 10 Items

Column 1
%

QUERY AFTER TABLE DROP IS PERFORMED

SQL Text

```
1  SELECT * FROM TestTable
```

Selection Query?

No | Yes

Run

Error SQL Error! Invalid object name 'TestTable'.

WHAT WAS ACTUALLY RUN

```
SELECT TOP 10 FirstName, LastName FROM Person WHERE
FirstName LIKE '%{{searchOne}}%'
```

```
SELECT TOP 10 FirstName, LastName FROM Person WHERE
FirstName LIKE '%'; DROP TABLE TestTable; SELECT '%'
```

We were able to turn the original query into three valid commands that performed unexpected operations on the database.

MODIFY QUERY TO USE A PERSON PICKER INSTEAD OF TEXT

```
{% assign searchTwo = 'Global' | PageParameter:'Person' | PersonByAliasGuid %}
SELECT TOP 10 FirstName, LastName FROM Person WHERE FirstName LIKE '%
{{searchTwo.FirstName}}%'
```

We're not accepting input directly from the URL so it should be safe, right?

NAMES ARE ALSO USER INPUT

'; DROP TABLE TempTable; SELECT ' DbTeamFamily

Member | Protection Profile: Medium

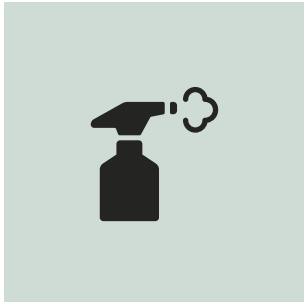
Unknown

A MORE SUBTLE INJECTION COMMAND

```
UPDATE UserLogin SET PersonId = 1 WHERE PersonId = 12345
```

Security is tied to person records not user logins. By changing the associated person record with a newly created login we can gain unauthorized access with our own credentials

TWO WAYS TO PREVENT SQL INJECTION



Sanitize User Input

Escaping single quotes in user input and performing operations to make sure number inputs are in fact valid numbers

Parameterize User Input

Send all user input as a SQL parameter instead of using Lava to generate the SQL statement

SANITIZING USER INPUT

Sanitizing input tries to ensure the user cannot add additional commands to the SQL we intend to run

- SanitizeSql Lava Filter

  Replaces single quotes with two single quotes so the character is escaped in the SQL query

  WHERE FirstName LIKE '{{search}}'

  If the user cannot add another single quote to end this statement and begin a new one, they cannot inject their own commands

- AsInteger Lava Filter

  Ensures input provided is a number and not text with another command

  WHERE Id = {{personId}}

  Ensuring the input is a number and not text also protects us from unauthorized commands

SANITIZING USER INPUT

Using our first example, we can see adding the the SanitizeSql filter keeps the user input inside the single quotes of our comparison instead of running new commands

```
{% assign searchOne = 'Global' | PageParameter:'Name' | SanitizeSql %}
SELECT TOP 10 FirstName, LastName FROM Person WHERE FirstName LIKE '%{{searchOne}}%'
```

```
SELECT TOP 10 FirstName, LastName FROM Person WHERE FirstName LIKE '%''; DROP TABLE
TestTable; SELECT ''%'
```

PARAMETERIZING USER INPUT

To add parameters to a Lava SQL command, just add them in the opening tag

```
{% assign searchOne = 'Global' | PageParameter:'Name' %}
{% sql search:'{{searchOne}}' %}
    SELECT TOP 10 * FROM Person WHERE FirstName LIKE @search
{% endsql %}
```

DYNAMIC DATA BLOCKS CAN MATCH SQL PARAMETERS TO URL QUERY PARAMETERS

Query Logic

Query

```
1  -- {% assign searchOne = 'Global' | PageParameter:'Name' %}
2  -- {% assign searchTwo = 'Global' | PageParameter:'Person' | PersonByAliasGuid %}
3  SELECT TOP 10 FirstName, LastName FROM Person WHERE FirstName LIKE @name
```

Query is a Stored Procedure

Timeout

30 (sec)

Parameters

name=

THE DIFFERENCE BETWEEN PARAMETERIZING AND SANITIZING

We interact with the database by sending text it interprets into commands. When the database is going to execute a set of commands it creates a plan for the operations it needs to perform.

- Sanitized input is included in the command text when the plan is made.
- The value of parameters are not included when the plan is made.

HOW YOU CAN PROTECT YOUR SYSTEM

PROTECT YOUR ROCK INSTANCE

Security Settings

- Disable duplicate checking for staff members
- Make sure staff members have elevated account protection

Training and Policies

- Staff in charge of merging records should be trained to spot potential attackers
- Don't over-grant access to Rock

Monitoring

- Create reports to help identify suspicious activity
- Verify staff members don't have multiple user logins
- Assign someone to check these reports regularly