Zabezpiecz formularz przed spamem

Prowadzi: Dawid Dęby

Agenda

1. Czym jest Spam
2. Poznamy swojego wroga
3. Captcha
4. Jak się zabezpieczyć
5. Moja droga do bezpiecznego formularza

SPAM

Wasze doświadczenia

1. Pracowałeś przy formularzach?
2. Zabezpieczałeś formularz?
3. Spotkałeś się ze spamem?
4. Sam bawiłeś się w hackera?

Metody łamania zabezpieczeń (spam)

Poznaj swojego *“wroga”*

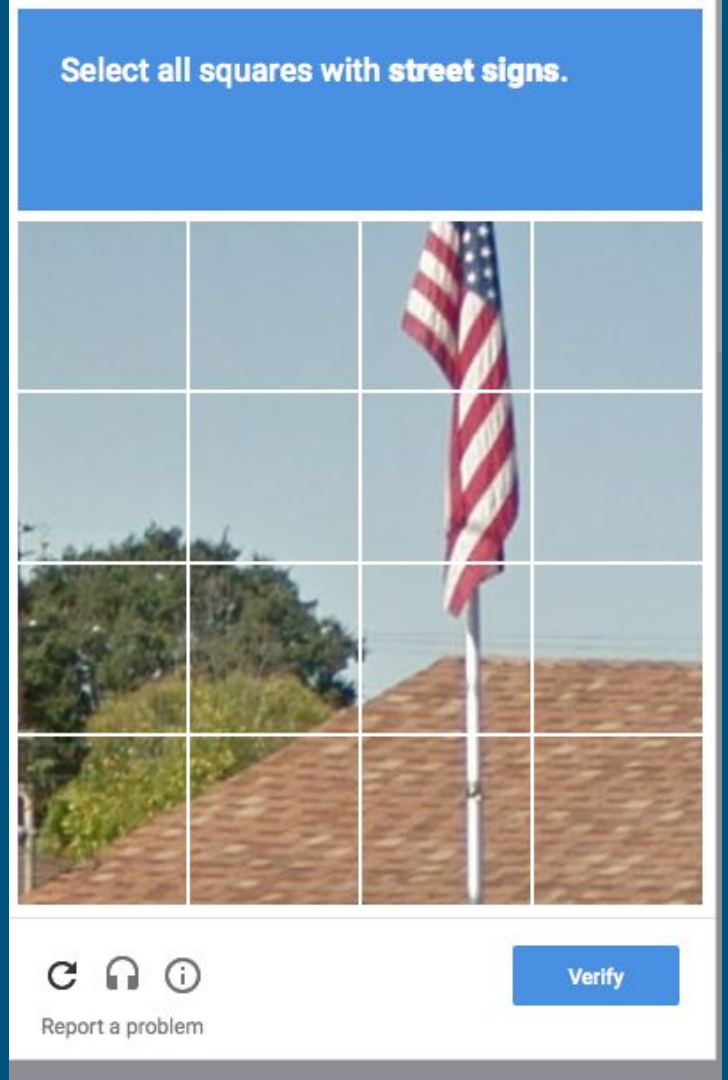
1. PayBack Record
2. Formfilling bot
3. Hackerzy (atak spersonalizowany)
4. Ludzie

Captcha?

Zmniejsza konwersję nawet do 3.2%

Frustrująca

Głosowanie/Like na FB



Jak zabezpieczyć się przed spamem?

https://ddeby.pl

Jak zabezpieczyć się przed spamem

1. HoneyPot
2. Stempel czasowy
3. Inkrementacja zmiennej w JS na front
4. Zmienne nazwy pól

Moja implementacja

1. HoneyPot
2. Walidacja z JS na front
3. Stempel czasowy
4. Stempel czasowy w postaci zaszyfrowanej
5. Zmienne nazwy pól

HoneyPot

```
class HoneyPotForm(forms.Form):
    text = forms.CharField()
    honeypot = forms.CharField(label=u'Zostaw to pole puste')

    def clean_honeypot(self):
        if self.cleaned_data.get('honeypot'):
            raise ValidationError(u'Zostaw to pole puste')
```

Inkrementacja w JS - formularz

```
class JSTimeForm(forms.Form):
    MIN_TIME = 5  # 5 sec
    MAX_TIME = 600  # 10 min

    text = forms.CharField()
    time = forms.IntegerField(widget=forms.HiddenInput(attrs={'class': 'time'}))

    def clean_time(self):
        time = self.cleaned_data.get('time', 0)
        if time < self.MIN_TIME:
            raise ValidationError(u'Zwolnij kowboju! ;)')

        if time > self.MAX_TIME:
            raise ValidationError(u'Formularz wygasł. Przeładuj stronę')

        return time
```

Inkrementacja w JS - JavaScript

```
function init(){
    let elements = document.getElementsByClassName('time');
    for (var i = 0; i < elements.length; i++) {
        elements[i].setAttribute('value', 0);
    }
}

function increment(){
    let elements = document.getElementsByClassName('time');
    for (var i = 0; i < elements.length; i++) {
        elements[i].setAttribute('value', parseInt(elements[i].getAttribute('value')) + 1);
    }
}
document.addEventListener("DOMContentLoaded", function() {
    init();
    setInterval(increment, 1000);
})
```

Stempel czasowy

```
class BackendTimeForm(forms.Form):
    MIN_TIME = 5  # 5 sec
    MAX_TIME = 600  # 10 min

    text = forms.CharField()
    backendtime = forms.DateTimeField(widget=forms.HiddenInput(), required=True)

    def __init__(self, *args, **kwargs):
        super(BackendTimeForm, self).__init__(*args, **kwargs)
        self.fields['backendtime'].initial = datetime.now()

    def clean_backendtime(self):
        backendtime = self.cleaned_data.get('backendtime')
        current = datetime.now()
        time = current - backendtime

        if time.seconds < self.MIN_TIME:
            raise ValidationError(u'Zwolnij kowboju! ;)')

        if time.seconds > self.MAX_TIME:
            raise ValidationError(u'Formularz wygasł. Przeładuj stronę')
```

Zaszyfrowany stempel czasowy

```python
class HashedBackendTimeForm(forms.Form):
    MIN_TIME = 5  # 5 sec
    MAX_TIME = 600  # 10 min

    text = forms.CharField()
    backendtime = forms.CharField(widget=forms.HiddenInput(), required=True)

    def __init__(self, *args, **kwargs):
        super(HashedBackendTimeForm, self).__init__(*args, **kwargs)
        self.fields['backendtime'].initial = cipher.encrypt(str(datetime.now()))

    def clean_backendtime(self):
        encrypted = cipher.decrypt(self.cleaned_data.get('backendtime'))
        current = datetime.now()
        time = current - parse_datetime(encrypted)

        if time.total_seconds() < self.MIN_TIME:
            raise ValidationError(u'Zwolnij kowboju! ;)')

        if time.total_seconds() > self.MAX_TIME:
            raise ValidationError(u'Formularz wygasł. Przeładuj stronę')
```

Zmienne nazwy pól - formularz

```
class SecureForm(forms.Form):
    THRESHOLD = 2  # 2 sec
    MIN_TIME = 5  # 5 sec
    MAX_TIME = 600  # 10 min

    text = forms.CharField()
    spin = forms.CharField(widget=forms.HiddenInput(), required=True)

    def __init__(self, *args, **kwargs):
        super(SecureForm, self).__init__(*args, **kwargs)
        # set spin initial value
        double_encrypted = self.data.get('spin') or cipher.encrypt(str(datetime.now()), 2)
        self.fields['spin'].initial = double_encrypted

        # set custom field
        self.field_name = cipher.decrypt(double_encrypted)
        self.fields[self.field_name] = forms.CharField(
            widget=forms.HiddenInput(), initial=self.data.get(self.field_name, 0)
```

Zmienne nazwy pól - Szablon

```
<form method="post" action=".">
    {% csrf_token %}
    {{ form }}
    <button type="submit">Wyślij</button>
</form>

<script>
    function increment(){
        let element = document.getElementById('id_' + '{{ form.field_name }}')
        element = setAttribute('value', parseInt(element.getAttribute('value')) + 1);
    }

    document.addEventListener("DOMContentLoaded", function() {
        setInterval(increment, 1000);
    });
</script>
```

Zmienne nazwy pól - walidacja

```
def clean_spin(self):
        double_encrypted = self.cleaned_data['spin']
        encrypted, time_delta = self.__check_time(double_encrypted)
        decrypted, backendtime_delta = self.__check_backendtime(encrypted)
        self.__deltas_compare(backendtime_delta, time_delta)
        return decrypted
```

Zmienne nazwy pól - walidacja czasu

```
def __check_time(self, double_encrypted):
        encrypted = cipher.decrypt(double_encrypted)

        try:
            time = int(self.data.get(encrypted))
        except (ValueError, TypeError):
            raise ValidationError(u'Włącz obsługę JavaScript')

        if time < self.MIN_TIME:
            raise ValidationError(u'Zwolnij kowboju! ;)')

        if time > self.MAX_TIME:
            raise ValidationError(u'Formularz wygasł. Przeładuj stronę')

        return encrypted, time
```

Zmienne nazwy pól - walidacja stempla

```
def __check_backendtime(self, value):
        date_str = cipher.decrypt(value)

        current_date = datetime.now()
        form_date = parse_datetime(date_str)
        delta = current_date - form_date

        if delta.total_seconds() < self.MIN_TIME:
            raise ValidationError(u'Zwolnij kowboju! ;)')

        if delta.total_seconds() > self.MAX_TIME:
            raise ValidationError(u'Formularz wygasł. Przeładuj stronę')

        return date_str, delta.total_
```

Zmienne nazwy pól - walidacja przez porównanie

```
def __deltas_compare(self, backendtime_delta, time_delta):
        if abs(int(backendtime_delta - time_delta)) < self.THRESHOLD:
            raise ValidationError(u'Nie kombinuj')
```

Rozwiązanie idealne?

+ Proste
+ Szybkie
+ Elastyczne
+ Niezależne od bazy

- Odwołanie do zmiennej
- Skrypt w szablonie
- Nie daje 100% skuteczności

Podsumowanie

https://ddeby.pl/blog/zabezpiecz-formularz-przed-spamem


A
BICYCLE®
GUARDIANS
THE U.S. PLAYING CARD COMPANY
UNITED STATES OF AMERICA
1812-R6591
A