АВТОМАТИКА, ТЕЛЕМЕХАНИКА, СВЯЗЬ, ЭЛЕКТРОЭНЕРГЕТИКА, ИНФОРМАЦИОННЫЕ СИСТЕМЫ

The Bulletin of Kazakh Academy of Transport and Communications named after M. Tynyshpayev ISSN 1609-1817. Vol. 116, No.1 (2021), pp.275-281

UDC 004.9 **10.52167/1609-1817-2020-116-1-275-281**

DJANGO AS SECURE WEB-FRAMEWORK IN PRACTICE

Kulanda Duisebekova, PhD in Physics and Mathematics, Associate Professor, Al-Farabi Kazakh National University, IITU, Almaty, Kazakhstan, dkulan1@mail.ru

Roman Khabirov, master student, Al-Farabi Kazakh National University, Almaty, Kazakhstan, khabirovroman@gmail.com

Azamat Zholzhan, master student, Al-Farabi Kazakh National University, Almaty, Kazakhstan, azamatzholzhan@gmail.com

Abstract. Today information security has become one of the most important parts of our social media life. Social and media resources are based on web-services in the cloud. It means security of web-services is the equality of people’s social, media, data and information security. In this paper the most important focus was on special secure techniques and tools inside the most popular web-framework on Python programming language - Django. Django has several really strong design patterns and techniques with special tools to store and send user’s data in very secure methods. Developer can easily install in Django-application some new extra instruments, tools and special libraries to make web-application more securable. Django has such extremely useful instruments like Django-ORM, CSRF-tokens, XSS-protection and so many else. For example, Django-ORM (Object-Relational Mapping) is a really powerful instrument to be used for protection of such attacks like SQL-injections. One more instance, CSRF-token (Cross-site request forgery - token) is really amazing internal Django's tool against cross-site request forgery attacks that Django uses in html-templates. The best practice and good examples of these tools are shown inside this paper. Moreover, in the paper were demonstrated comparison of different attack cases and their deep analysis with protection methods from these attacks by Django's tools and techniques. One more thing, we also briefly reviewed other types of vulnerabilities and methods of protection against them and hope this article has given an understanding of the Django security techniques. Finally, Django could become more securable after each next version.

Keywords: webserver, django, web-framework, ORM, SQL-injection, XSS, CSRF.

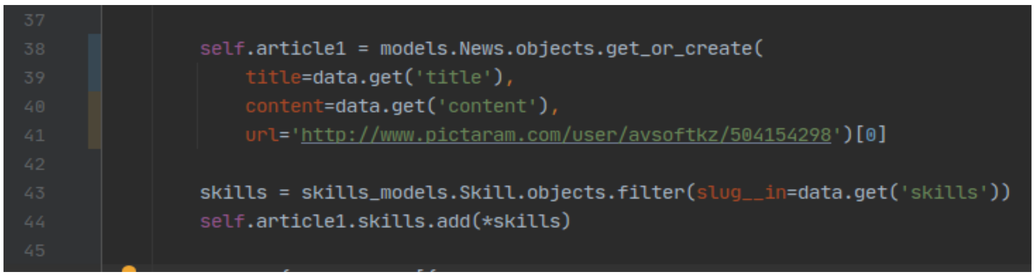
INTRODUCTION

If we talk about web-frameworks, we can easily find non-securable frameworks. The research is directed to one of the most popular web-framework. Django is one of the easiest to use and manage web-frameworks. Django has an amazing, very comfortable technique to use, which is called ORM. Django ORM creates specific queries to a database that are more secure and can save website from the SQL-injection attacks. There also can be CSRF-attacks that can be harmful for the clients if an attacker catches the response that was sent to a server. More likely known attack is XSS because it has different vectors of attacks to a webserver. For defending from XSS Django has its own templates that foresees all attack vectors. Services written by Django automatically become more secure and it is really not easy to hack Django services. Furthermore, knowledge of customization of Django's settings and correct usage of all Django’s included methods and techniques stands backend developers to the next level of professionalism. SQL-queries to databases from web-frameworks are in the deep past.

These methods and other techniques will be shown in the next chapters.

DJANGO OBJECT RELATIONAL MAPPING (ORM)

Django Object Relational Mapping (ORM) - is a really complex set of special techniques the main purpose of which to make the life of a developer easier. Django ORM connects Django projects with databases to make some different queries. It is not important which type of database is used in a project, because Django ORM translates queries to understandable ways for dissimilar databases. First of all, Django ORM is designed in the specified way to be very similar with object-oriented programming (OOP) paradigm. OOP is very popular within programming culture as a result Django ORM became very useful to understand and easy to use technique [1].

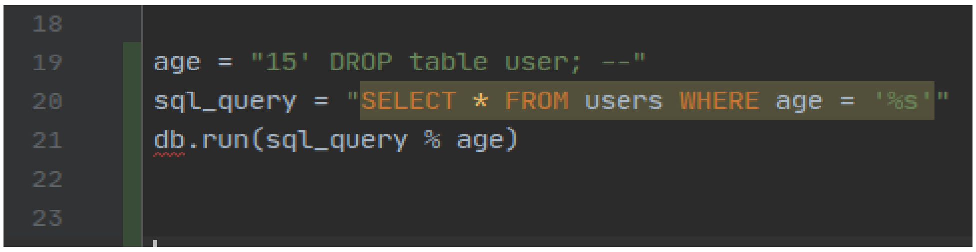


```
37
38        self.article1 = models.News.objects.get_or_create(
39            title=data.get('title'),
40            content=data.get('content'),
41            url='http://www.pictaram.com/user/avsoftkz/504154298')[0]
42
43        skills = skills_models.Skill.objects.filter(slug__in=data.get('skills'))
44        self.article1.skills.add(*skills)
45
```

Figure 1 – Example of Django ORM usage
Рисунок 1 – Пример использования Django ORM

Figure 1 shows Django ORM in action: method get_or_create combines inside 2 scenarios: 1) If object with parameters inside the method does not exist then creates an instance with such parameters; 2) If object exists with such parameters then returns found obj back. On the 43rd line of code in Figure 1 is a written query set of "skills" where "slug field" exists in "data" where data is a dictionary with key "skills". In the final step, on the 43rd line of code "skills" is added to article's "skills". In the result, from given example we can make conclusion that Django web-framework is not working with databases directly, Django web-framework uses ORM for all database related operations.

SQL-injection - is one of the most popular type of attacks to webservers. The main idea of this attack is adding SQL-code to the parameters while SQL-query is running.



```
age = "15' DROP table user; --"
sql_query = "SELECT * FROM users WHERE age = '%s'"
db.run(sql_query % age)
```

Figure 2 – Example of SQL-injection
Рисунок 2 – Пример SQL-инъекции

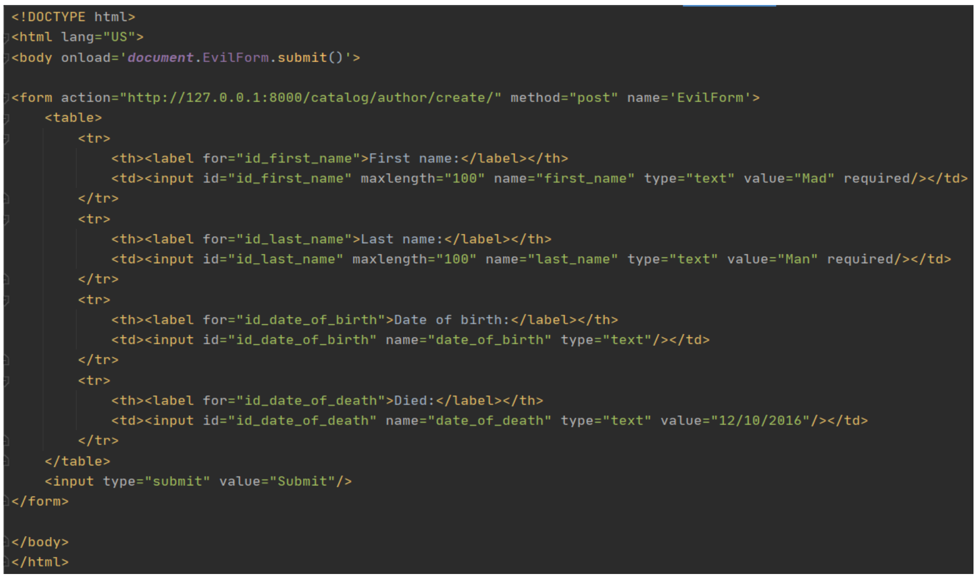
In Figure 2 shows the process of SQL-injection while SQL query is trying to get "users" with special age, hackers are dropping table of users. In Django, probability of successful results for such attacks are minimal, because design patterns of ORM are strong in typization of fields for queries. Moreover, in Django ORM could be written special validators for fields to make typization more strong and secure, it is equality to say that queries to databases become more secure [2].

EXPLANATION ABOUT CSRF ATTACK WITH EXAMPLES

One of the most used attacks for clients of websites is Cross Site Request

Forgery (CSRF) method, which mostly uses limitations of protocol Hyper Text Transfer Protocol (HTTP). If a victim visits the web-site which is created by an intruder, a request of victim is secretly sent on behalf of him known to another server, and intruder can perform actions on behalf of another user without their consent and can manipulate with configures of victims data. To carry out this CSRF attack, the victim must be authenticated on the server where the server request is sent, and this request must not require any confirmation form from the victim, which cannot be ignored or tampered with by the attacking script [3].

For example, assume there is an intruder who wants to add the author to our Local Company application. Obviously our intruder is not doing this attack for the money, more ambitious intruders will think forward and can use the described approach to perform more dangerous tasks (for example, transferring users money to their personal accounts, etc.). In order to do this, an intruder can create an HTML file similar to the one shown below in Figure 3, which will contain an author creation form, which will be sent as soon as this file is loaded into the browser. The hacker will send this file to all Librarians and will wait for any of them to open the file (it contains only harmless information). If the file will be opened by any logged in user with Librarian rights, then the form will be submitted on his behalf and create a new user.

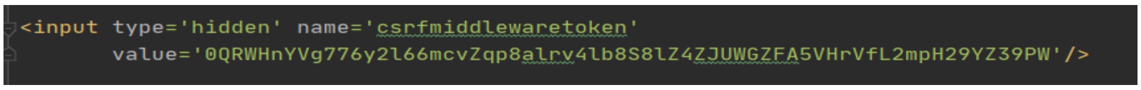


```html
<!DOCTYPE html>
<html lang="US">
<body onload='document.EvilForm.submit()'>

<form action="http://127.0.0.1:8000/catalog/author/create/" method="post" name='EvilForm'>
    <table>
        <tr>
            <th><label for="id_first_name">First name:</label></th>
            <td><input id="id_first_name" maxlength="100" name="first_name" type="text" value="Mad" required/></td>
        </tr>
        <tr>
            <th><label for="id_last_name">Last name:</label></th>
            <td><input id="id_last_name" maxlength="100" name="last_name" type="text" value="Man" required/></td>
        </tr>
        <tr>
            <th><label for="id_date_of_birth">Date of birth:</label></th>
            <td><input id="id_date_of_birth" name="date_of_birth" type="text"/></td>
        </tr>
        <tr>
            <th><label for="id_date_of_death">Died:</label></th>
            <td><input id="id_date_of_death" name="date_of_death" type="text" value="12/10/2016"/></td>
        </tr>
    </table>
    <input type="submit" value="Submit"/>
</form>

</body>
</html>
```

Figure 3 – HTML form to realize CSRF attack to the LocalCompany
Рисунок 3 – HTML-форма для реализации CSRF-атаки на локальную компанию

Start the development web server and log into the super user account. Copy the code to a new html file and then open it in a browser. You should get a CSRF error because Django has protection against this kind of attack.

The security mechanism is that you add the {% csrf_token%} template tag to your form. This token will be rendered in your HTML as shown below in Figure 4, with a value that is unique to each user requesting the form.



```html
<input type='hidden' name='csrfmiddlewaretoken'
       value='0QRWHnYVg776y2l66mcvZqp8alrv4lb8S8lZ4ZJUWGZFA5VHrVfL2mpH29YZ39PW'/>
```

Figure 4 – Adding a token for the HTML form
Рисунок 4 – Добавление токена для HTML-формы

Django generates a user / browser unique token and rejects all forms that do not contain it or contain an invalid value. To continue using this type of attack, the intruder must have to find and add the correct CSRF token for each targeted user. This means that an intruder can no longer use mass mailings of one malicious file to all Librarians, since the CSRF token will be unique for each of them.

Django's CSRF protection is enabled by default. You should always use the {% csrf_token%} tag in your forms and use POST for requests that might change or add data to your database.

DJANGO SETTINGS AS PART OF PROTECTION

In this part of the paper, we will discuss configurations in projects, developer's and production settings and the importance of dividing environments by some specified cases. The best practice for good Django developers is dividing's settings to production and dev versions, because in dev versions some services for project could be not so important like in production. If we talk about security, the gap between dev and prod versions increases several times. Production version of Django project must be secure, uses ssl-certificates, running on a strong, scalable and stable webserver like NGINX or Apache.

Settings SSL/HTTPS can be used on a web server to encrypt all traffic between the server and the user, including login data that would otherwise be sent as plain text (readable by any person who intercepts the request). Using HTTPS is highly recommended [3]. If HTTPS is used, Django allows the following security methods:

- SECURE_PROXY_SSL_HEADER can be used to verify that a secure connection is always used, even if the data comes from a proxy using a protocol other than HTTP.
- SECURE_SSL_REDIRECT is used to redirect all requests from HTTP to HTTPS.
- Use HTTP Strict Transport Security (HSTS). This HTTP header informs the browser that all subsequent requests should always use HTTPS. Combined with redirecting HTTP requests to HTTPS, this option allows you to ensure that HTTPS is used in every request. HSTS can also be configured with the SECURE_HSTS_SECONDS.
- SECURE_HSTS_INCLUDE_SUBDOMAINS options or on the web server.
- Use 'secure' cookies by setting SESSION_COOKIE_SECURE and CSRF_COOKIE_SECURE to True. This will ensure that these cookies are only sent over HTTPS.
- Use ALLOWED_HOSTS to only accept requests from trusted hosts.

There are also many other protection techniques and guidelines for their use. However, show techniques are the most popular in Django security design and patterns.

XSS ATTACKS

Cross-Site Scripting (XSS) is a term used to describe a class of attacks that allows an attacker to inject scripts through a website that will be executed on the device of the user who has entered the page. This often happens by storing malicious code in a database, from where this code will be returned and executed for the user who requested some data responded by the server (a typical example is storing the <script> tag with malicious code in a comment that can be seen by another user). Another attack vector is to generate a specific link, when clicked by the user, it will trigger the execution of some disguised JavaScript code in their browser.

Django's templating system protects against most XSS attacks by escaping certain characters that are considered "dangerous" in HTML. We can demonstrate this by trying to inject arbitrary JavaScript code into our Local Company application via the add author form. For example, first of all run the web-site, by using developer server (python manage.py runserver), in the browser enter the web-site by the rules of administrator, then open the page adding author(it must be accessible in URL: http://127.0.0.1:8000/catalog/author/create/), type in the forms name and surname and then add in surname field script(<script>alert('Test alert');</script>), press the button Submit for saving typed information, after saving the system will not run the alert(), instead of it will be displayed in plain text. If you look at the HTML source code, you will find that the "dangerous" characters - for example, such as the tag brackets - have been replaced by their equivalent safe html entities (for example > to & gt;).

Using Django templates protects you from most XSS attacks. However, it is possible to disable this protection so that escaping will not automatically be applied to all fields that will not have to be filled in by the user (for example, the help_text field is usually not filled by the user, so Django will not escape its value). Also, XSS attacks can be carried out through other unreliable data sources, such as cookies, third-party services or downloaded files (and other sources, the data of which was not specially processed before being displayed on the page). If you display data from these sources, you must add your own handler to sanitize the data [3].

CONCLUSION

Django has methods to provide protection against common types of attacks, including XSS and CSRF attacks. In this article, we have demonstrated how various types of attacks are handled by Django using our Local Company application as an example. Django has shown itself from very good sides. Django has protection from common hacker's attacks. Furthermore, Django has a huge community and it is still developing and growing up. Good communities in the near future can easily develop new features to create some extra ways of server and data protection. In the result Django could become more secure after each next version is released. In this paper we have not talked about all possible Django security techniques, but we tried to analyze the most common and more popular of them.

We also briefly reviewed other types of vulnerabilities and methods of protection against them. We hope this article has given you an understanding of the security Django techniques.

REFERENCES

[1] Kouraklis, John. In the Land of ORM. DOI: 10.1007/978-1-4842-5013-6_1. (10.11.2020).
[2] Yang, Jean & Hance, Travis & Austin, Thomas & Solar-Lezama, Armando & Flanagan, Cormac & Chong, Stephen. End-To-End Policy-Agnostic Security for Database-Backed Applications. https://www.researchgate.net/publication/280061993 (10.11.2020).
[3] Shyam, Adamya & Mukesh, Nitin. A Django Based Educational Resource Sharing Website: Shreic. // Journal of Scientific Research. – 2020. – Vol. 64. – P. 138 – 152.
[4] Robertson, William & Vigna, Giovanni. Static enforcement of web application integrity through strong typing // SSYM'09: Proceedings of the 18th conference on USENIX security symposium. – 2009. – P. 283-298.
[5] Hobson, Tanner & Doucet, Mathieu & Leal, Ricardo. Django Remote Submission // The Journal of Open Source Software. – 2017. – P. 1-2.
[6] Shyamasundar, R. & Nelabhotla, Narendra Kumar & Taware, Abhijit & Vyas, Parjanya. An Experimental Flow Secure File System // 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering. – 2018. – P. 790-799.
[7] Petukhov Andrey & Kozlov Dmitry. Detecting Security Vulnerabilities in Web Applications Using Dynamic Analysis with Penetration Testing. https://owasp.org/www-pdf-archive/OWASP-AppSecEU08-Petukhov.pdf (17.11.2020).
[8] Finifter Matthew & Wagner David. Exploring the Relationship Between Web Application Development Tools and Security // Proceedings of the 2nd USENIX conference on Web application development. – 2011. – P. 9.
[9] Susheel S., Narendra Kumar N.V., Shyamasundar R.K. Enforcing Secure Data Sharing in Web Application Development Frameworks Like Django Through Information Flow Control // 11th International Conference, ICISS 2015. – P. 3-10.

REFERENCES

[1] Kuraklis, Dzhon. *V strane ORM* [in USA: In the Land of ORM]. DOI: 10.1007 / 978-1-4842-5013-6_1. (10.11.2020).
[2] Yan, Dzhin i Khans, Trevis i Ostin, Tomas i Solar-Lezama, Armando i Flanagan, Kormak i Chong, Stiven. *Skvoznaya nezavisimaya ot politik bezopasnost' dlya prilozheniy na baze baz dannykh* [in USA: End-To-End Policy-Agnostic Security for Database-Backed Applications]. https://www.researchgate.net/publication/280061993 (10.11.2020).

[3] SH'yam, Adam'ya i Mukesh, Nitin. *Veb-sayt dlya obmena obrazovatel'nymi resursami na osnove Django: Shreic* [in India: A Django Based Educational Resource Sharing Website: Shreic]. Journal of Scientific Research. – 2020. – Vol. 64. – P. 138 – 152.
[4] Robertson, Uil'yam i Vin'ya, Dzhovanni. *Staticheskaya realizatsiya tselostnosti veb-prilozheniy za schet strogoy tipizatsii* [in USA: Static enforcement of web application integrity through strong typing]. SSYM'09: Proceedings of the 18th conference on USENIX security symposium. – 2009. – P. 283-298.
[5] Khobson, Tanner i Duse, Mat'ye i Lil, Rikardo. *Udalennaya otpravka Django* [in USA: Django Remote Submission]. The Journal of Open Source Software. – 2017. – P. 1-2.
[6] SH'yamasundar, R. i Nelabkhotla, Narendra Kumar i Tavare, Abkhidzhit i V'yas, Pardzhan'ya. *Eksperimental'naya bezopasnaya faylovaya sistema Flow Shreic* [in India: An Experimental Flow Secure File System]. 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering. – 2018. – P. 790-799.
[7] Petukhov Andrey i Kozlov Dmitriy. *Obnaruzheniye uyazvimostey bezopasnosti v veb-prilozheniyakh s pomoshch'yu dinamicheskogo analiza s testirovaniyem na proniknoveniye* [in Russian: Detecting Security Vulnerabilities in Web Applications Using Dynamic Analysis with Penetration Testing]. https://owasp.org/www-pdf-archive/OWASP-AppSecEU08-Petukhov.pdf (17.11.2020).
[8] Finifter Met'yu i Vagner David. *Izucheniye vzaimosvyazi mezhdu instrumentami razrabotki veb-prilozheniy i bezopasnost'yu* [in USA: Exploring the Relationship Between Web Application Development Tools and Security]. Proceedings of the 2nd USENIX conference on Web application development. – 2011. – P. 9.
[9] Sushil S., Narendra Kumar N.V., SH'yamasundar R.K. *Obespecheniye bezopasnogo obmena dannymi v platformakh razrabotki veb-prilozheniy, takikh kak Django, posredstvom upravleniya informatsionnymi potokami* [in India: Enforcing Secure Data Sharing in Web Application Development Frameworks Like Django Through Information Flow Control]. 11th International Conference, ICISS 2015. – P. 3-10.

DJANGO КАК БЕЗОПАСНЫЙ ВЕБ-ФРЕЙМВОРК НА ПРАКТИКЕ

Дуйсебекова Кулянда Сейтбековна, кандидат физико-математических наук, доцент**,** Казахский национальный университет им. аль-Фараби, МУИТ, г.Алматы, Казахстан, dkulan1@mail.ru.
Хабиров Роман, магистрант, Казахский национальный университет им. аль-Фараби, г.Алматы, Казахстан, khabirovroman@gmail.com
Жолжан Азамат, магистрант, Казахский национальный университет им. аль-Фараби, г.Алматы, Казахстан, azamatzholzhan@gmail.com

Аннотация. Сегодня информационная безопасность стала одной из важнейших частей нашей жизни в социальных сетях. Социальные и медийные ресурсы основаны на веб-сервисах в облаке. Это означает, что безопасность веб-сервисов — это равноценно безопасности людей в социальных сетях, СМИ, данных и информации. В этой статье будут показаны и объяснены специальные методы безопасности внутри самого популярного веб-фреймворка на языке программирования Python - Django. В Django есть несколько действительно эффективных шаблонов проектирования и методов для хранения и отправки пользовательских данных очень безопасными методами. В Django есть такие чрезвычайно полезные инструменты, как Django-ORM, CSRF-токены, XSS-защита и многое другое. Передовая практика и хорошие примеры этих инструментов будут показаны внутри этой статьи. Кроме того, в статье будет сравнение и анализ случаев атак, и защита от этих атак с помощью методов и приемов Django.

Ключевые слова: webserver, django, web-framework, ORM, SQL-injection, XSS, CSRF.

DJANGO - ҚАУІПСІЗ ВЕБ-ФРЕЙМВОРК ҚОЛДАНЫСЫ ПРАКТИКАДА

Дуйсебекова Кулянда Сейтбековна, физика-математика ғылымдарының кандидаты, доцент, Әл-Фараби атындағы ҚазҰУ, ХАТУ, Алматы, Қазақстан, dkulan1@mail.ru.
Хабиров Роман, Әл-Фараби атындағы Қазақ ұлттық университетінің магистранты, Алматы, Қазақстан, khabirovroman@gmail.com
Жолжан Азамат, Әл-Фараби атындағы Қазақ ұлттық университетінің магистранты, Алматы, Қазақстан, azamatzholzhan@gmail.com

Аңдатпа. Бүгінгі таңда ақпараттық қауіпсіздік біздің әлеуметтік медиа өміріміздің маңызды бөліктерінің біріне айналды. Әлеуметтік және медиа ресурстар бұлттағы веб-қызметтерге негізделген. Бұл веб-қызметтердің қауіпсіздігі дегеніміз - бұл адамдардың әлеуметтік, бұқаралық ақпарат құралдарының, деректердің және ақпараттық қауіпсіздіктің теңдігі. Бұл жұмыста Python бағдарламалау тіліндегі ең танымал Django веб-фреймворкінде арнайы қауіпсіз техникалар көрсетілген және түсіндірілген. Django-да пайдаланушының деректерін өте қауіпсіз әдістермен сақтауға және жіберуге арналған бірнеше мықты дизайн үлгілері мен әдістері бар. Django-да Django-ORM, CSRF-токендер, XSS-қорғаныс және тағы басқалар сияқты өте пайдалы құралдар қолданылады. Олардың ең жақсы тәжірибесі мен үлгілері осы жұмыста көрсетілген. Сонымен қатар мақалада шабуыл жағдайларына салыстыру және талдау жасалып және аталған шабуылдардан Джанго әдістері мен тәсілдері арқылы қорғану жолдары қарастырылған.

Түйінді сөздер: webserver, django, web-framework, ORM, SQL-injection, XSS, CSRF.

The Bulletin of Kazakh Academy of Transport and Communications named after M. Tynyshpayev ISSN 1609-1817. Vol. 116, No.1 (2021), pp.281-286

METHODS FOR SOLVING DIFFERENTIAL EQUATIONS IN MATCHCAT

Nurbapa Mekebayev, PhD, Al-Farabi Kazakh National University, Almaty, Kazakhstan, ,nurbapa@mail.ru

Aruzhan Alpysbajқyzy, master degree, Kazakh National Women's Pedagogical University.

Abstract. In the article, students are invited to solve differential equations using computer programs. One of the main areas of application of information technology is mathematical and scientific and technical calculations. These calculations are very important today. One of the most powerful and efficient mathematical systems for this is Mathcad. Mathcad-Matlab, a system that has a special place among many similar systems. Mathcad remains the only system that describes the solution of common math problems. Mathcad is a system with a very efficient mathematical-oriented interface and an excellent tool for scientific graphics, and also allows you to perform numerical and analytical calculations. The article discusses ways to solve differential equations in MatchCat. The solution of simple differential equations is widely used in the practice of scientific and technical calculations. Although linear simple differential equations have solutions in the form of special functions, many physical systems are characterized by nonlinear and linear simple differential equations without analytic solutions. In this case, you should use numerical methods to solve simple differential equations.

Continuous improvement of computer-oriented technologies provides their use in the learning process, making them a powerful tool for students to receive a variety of information, as well as to increase interest in learning, increase motivation, visual and scientific character, etc. makes it an effective tool. The introduction of such technologies in the educational process in some way changes the traditional didactic system, helps to take into account the specifics of teaching disciplines, promotes the rational management of educational and cognitive activities of students. Provides educational interactivity of future professionals in the study of disciplines, both in the classroom and in the process of independent work.

Keywords: differential equations, MatchCat, higher order equations, information technology, multimedia technology.

ҒТАХР 27.29.19 **10.52167/1609-1817-2020-116-1-281-286**

Н.О. Мекебаев[1], А.А. Алпыспай[1]

[1]Қазақ ұлттық қыздар педагогикалық университеті, Алматы, Қазақстан

ДИФФЕРЕНЦИАЛДЫҚ ТЕҢДЕУЛЕРДІ МАТСНСАТ ОРТАСЫНДА ШЫҒАРУ ЖОЛДАРЫ

Аңдатпа. Мақалада студенттерге компьютерлік бағдарламаны қолдана отырып дифференциалдық теңдеулерді шешу қарастырылған. Ақпараттық технологияларды қолданудың бірден-бір негізгі облысы математикалық және ғылыми-техникалық есептеу болып табылады. Бүгінгі күні мұндай есептеу жұмыстарын жүргізу өте өзекті мәселе. Бұл үшін барынша сәйкес келетін ең бір мықты және тиімді математикалық жүйе-Mathcad болып табылады. Mathcad-Matlab,Maple,Mathematica және тағы да басқа көптеген осындай