POLITECHNIKA GDAŃSKA
WYDZIAŁ ELEKTRONIKI, TELEKOMUNIKACJI I INFORMATYKI

Imię i nazwisko autora rozprawy: Paweł Lubomski
Dyscyplina naukowa: informatyka

ROZPRAWA DOKTORSKA

Tytuł rozprawy w języku polskim:
Kontekstowo zorientowany model bezpieczeństwa systemów internetowych

Tytuł rozprawy w języku angielskim:
Context-oriented security model of Internet systems

Promotor
podpis
prof. dr hab. inż. Henryk Krawczyk, prof. zw.

Gdańsk, rok 2015

„Żeby widzieć jasno, wystarczy zmienić perspektywę."
Antoine de Saint-Exupéry

Podziękowania

Praca nad niniejszą rozprawą była dla mnie przygodą porównywalną z odkrywaniem nowych lądów przez żeglarzy. Jak każdy sternik, by móc w pełni stawić czoła nieznanym wodom potrzebuję dwóch rzeczy: macierzystego, spokojnego portu, do którego można zawsze bezpiecznie wrócić oraz niezawodnego kompasu.

Portem tym jest moja kochana Rodzina tworzona i pielęgnowana wspólnie ze wspaniałą Żoną Dorotą, natomiast kompasem na bezkresnych oceanach Profesor Henryk Krawczyk.

Tylko dzięki inspiracji, wsparciu, cierpliwości i wyrozumiałości tych dwóch osób wyprawa ta mogła zakończyć się sukcesem, za co gorąco im dziękuję.

Spis treści

Wykaz pojęć, symboli i oznaczeń

Symbol	Znaczenie
U	zbiór użytkowników, klientów systemu/usługi, podmiotów (*ang. user, subject*)
R	zbiór ról (*ang. role*)
P	zbiór uprawnień (*ang. permission*)
P(u)	zbiór uprawnień użytkownika u
P(s)	zbiór uprawnień usługi s (wymaganych przez tą usługę)
S	zbiór usług, operacji (*ang. service*)
$\mathbf{S_f}$	zbiór usług/operacji funkcjonalnych udostępnianych przez system
$\mathbf{S_s}$	zbiór usług/operacji bezpieczeństwa udostępnianych przez system
$\mathbf{S_c}$	zbiór usług wyznaczania kontekstu udostępnianych przez system
O	zbiór obiektów (*ang. object*)
D	zbiór danych, zasobów informacyjne (*ang. data*)
T	zbiór zagrożeń (*ang. threat*) T = {T1, T2, T3, T4, T5, T6}
T1 … T6	zbiory kategorii zagrożeń T1 = $\{t_1, t_2, \ldots, t_n\}$
V	zbiór podatności (*ang. vulnerability*)
ρ	poziom ryzyka, jaki związany jest z wykorzystaniem wykrytej podatności (luki)
φ	liczba podatności w analizowanym systemie
I	zbiór intruzów, atakujących (*ang. intruder, attacker*)
C	zbiór wartości kontekstu (*ang. context*)
CP	zbiór wartości parametru kontekstu (*ang. context parameter*)
|A|	moc zbioru A
approx	wartość przybliżona c (wykorzystywana w macierzach M_{PCN})
ISS	internetowy system usługowy (*ang. Internet service system*)
TL	poziom zaufania do użytkownika (*ang. user trust level*)
λ	poziom zaufania użytkownika do systemu (*ang. system trust level*); w niniejszej rozprawie przyjęto, że jest on tożsamy z poziomem bezpieczeństwa systemu β
β	poziom bezpieczeństwa systemu
ε	efektywność, użyteczność systemu rozumiana jako stosunek czasu poświęconego przez użytkownika na pracę merytoryczną w stosunku do całkowitego czasu poświęconego na pracę z systemem (merytoryczną i poświęconą na kontrolę bezpieczeństwa)
τ_B	czas użytkownika poświęcony na realizację działań użytkowych w systemie
τ_P	czas użytkownika poświęcony na kontrolę bezpieczeństwa związaną z przypisanymi mu uprawnieniami
τ_C	czas użytkownika poświęcony na kontrolę bezpieczeństwa związaną z przypisanymi mu uprawnieniami z uwzględnieniem bieżącego kontekstu

τ_T	czas użytkownika poświęcony na realizację mechanizmów bezpieczeństwa wynikających z bieżącego zaufania systemu do użytkownika (TL)
RBAC	model bezpieczeństwa Role Based Access Control
CoRBAC	model bezpieczeństwa Context-oriented Role Based Access Control (wykorzystujący bieżący kontekst)
TCoRBAC	model bezpieczeństwa Trust- and Context-oriented Role Based Access Control (wykorzystujący bieżący kontekst i zaufanie do użytkownika)
Vert	zbiór wierzchołków grafu
E	zbiór krawędzi grafu
G(Vert, E)	graf skierowany o wierzchołkach ze zbioru Vert i krawędziach ze zbioru E
G($Vert_1$, $Vert_2$, E)	graf dwudzielny o zbiorach wierzchołków $Vert_1$ i $Vert_2$ oraz zbiorze krawędzi E
G_Z	podgraf grafu G
M_Z	Macierz reprezentacji grafu G_Z
uis	scenariusz działań użytkownika (*ang. user interaction scenario*)
ss	scenariusz bezpieczeństwa (*ang. security scenario*)

Pojęcie	Znaczenie
podatność	wykryta w systemie luka bezpieczeństwa (*ang. vulnerability*)
żądanie użytkownika	pojedyncze połączenie użytkownika z systemem w celu wywołania konkretnej usługi (np. HTTP request)
sesja użytkownika	ciąg interakcji użytkownika z systemem zgodnie z pewnym scenariuszem działań stanowiącym sekwencję żądań użytkownika i odpowiedzi systemu (np. sesja HTTP)
bieżący kontekst	kontekst definiowany poprzez konkretne wartości każdego z parametrów kontekstu (określających stan tego parametru) w zadanym momencie czasu
bieżąca sytuacja	wartość parametrów kontekstu oraz innych parametrów/atrybutów związanych z obsługą żądania użytkownika w zadanym momencie czasu

Rozdział 1. Wprowadzenie

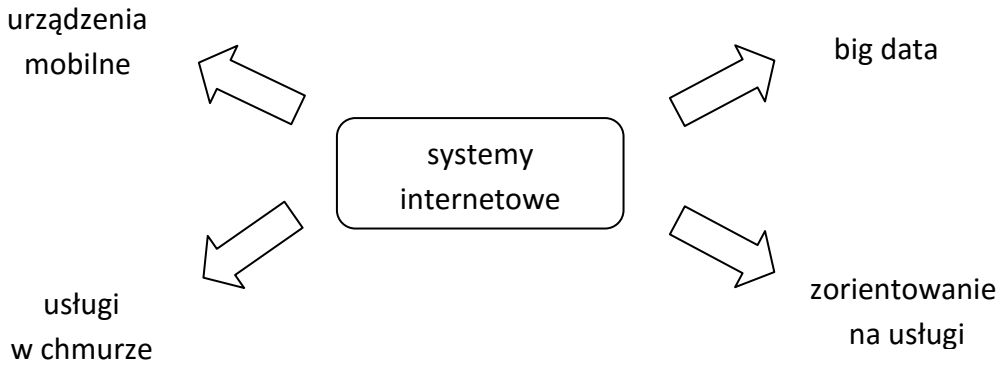
Zdefiniowano pojęcie systemu internetowego o charakterze usługowym. Przedstawiono problemy bezpieczeństwa tego typu systemów. Wskazano na istotną relację pomiędzy bezpieczeństwem, a zaufaniem użytkowników. Zaprezentowano motywację i cel rozprawy oraz podstawowe tezy będące przedmiotem rozważań. Umożliwiło to przyjęcie właściwej struktury rozprawy oraz odpowiedniej sekwencji rozważań.

1.1 Internetowe systemy usługowe

Zmieniająca się w bardzo dużym tempie technologia stawia coraz to nowsze wyzwania w obszarze bezpieczeństwa systemów informatycznych. Powstają duże, zcentralizowane centra danych (*ang. datacenters*) wykorzystujące wirtualizację i oferujące tzw. usługi w chmurze (*ang. cloud services*) [1][2]. Jednocześnie stosuje się rozwiązania zwiększające wydajność i niezawodność oferowanych usług. Stosowane jest w szerokim zakresie skalowanie poziome (klastrowanie) oraz georedundancja [3]. Trendy te są pochodną popularyzacji i powszechnego wykorzystania **systemów internetowych**. Są to systemy informatyczne dostępne przez Internet, najczęściej z wykorzystaniem przeglądarki WWW. Powstały one jako odpowiedź na problemy, z jakimi borykały się natywne aplikacje: m.in. zapewnienie kompatybilności z systemem operacyjnym klienta, efektywnej dystrybucji kolejnych wersji oprogramowania, a także umożliwienie wykorzystania dużego rozproszenia, wieloplatformowości i różnorodności typów urządzeń klienckich. Umożliwił to postęp, jaki dokonał się w zakresie przeglądarek WWW oraz możliwości i wydajności wykorzystywanego przez nie języka JavaScript.

Osadzanie systemów w modelu chmurowym wraz z rozpraszaniem i profesjonalizacją ich funkcjonalności implikuje **charakter usługowy** powstających rozwiązań. Dotyczy to zarówno usług skierowanych do użytkownika końcowego, jak i wykorzystywanych pomiędzy systemami. Związane jest to z potrzebą zapewnienia interoperacyjności będącej podstawą przepływu dużej ilości danych przetwarzanych obecnie w postaci cyfrowej w różnego typu systemach.

Rysunek 1 przedstawia cztery główne kierunki rozwoju współczesnych systemów internetowych, tzn. zorientowanie na obliczenia chmurowe w postaci usług, z uwzględnieniem dużych zbiorów danych oraz wykorzystaniem końcowych urządzeń mobilnych.

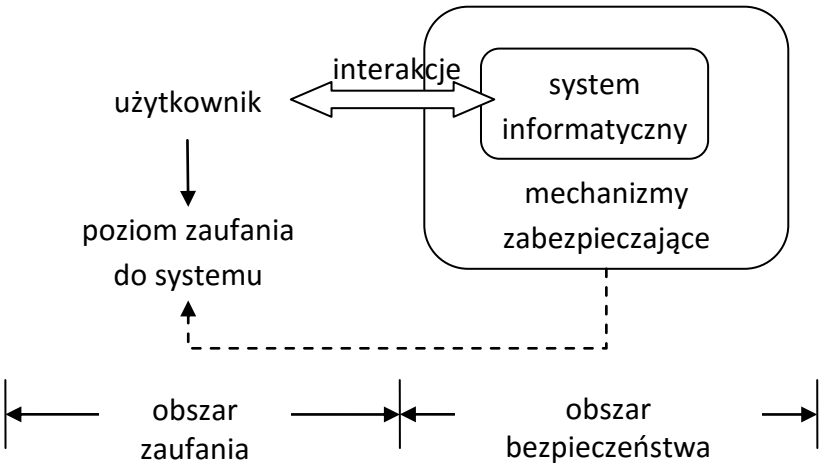


Rysunek 1 Cztery główne kierunki rozwoju systemów internetowych

W takich systemach wyzwaniem jest bezpieczne odseparowanie danych i właściwa kontrola dostępu do nich [4]. Podobnie konstrukcja architektury sieciowej wymaga dostosowania się do nowych tendencji rozwojowych [5]. W zakresie bezpieczeństwa kluczową rolę odgrywa szeroko rozumiana kontrola dostępu bazująca na cyfrowych tożsamościach (*ang. Digital identity*) [6][7][8] i w zdecydowanym stopniu dotyczy również samych aplikacji oraz systemów.

1.2 Zaufanie użytkownika, a bezpieczeństwo systemów informatycznych

Relację użytkownika z systemem informatycznym można rozpatrywać w dwóch obszarach dotyczących bezpieczeństwa samego systemu i zaufania jego użytkowników. Pierwszy z nich jest stricte technologiczny i proceduralny. Drugi natomiast dotyczy sfery miękkiej i sprowadza się do oceny, w jakim stopniu użytkownicy postrzegają system bezpiecznym i funkcjonalnym. Mimo tych różnic obszary te są ze sobą mocno powiązane i wpływają na siebie wzajemnie. Rysunek 2 ilustruje relację pomiędzy użytkownikiem, jego zaufaniem, a systemem i jego bezpieczeństwem. **Bezpieczeństwo systemu** rozumiane jest jako zbiór mechanizmów zabezpieczających system, tak, aby działał on zgodnie z oczekiwaniami użytkowników. Uwzględniane są tu wszystkie zabezpieczenia zarówno z obszaru kontroli dostępu (*ang. security*), jak i zapewnienia ciągłości działania (*ang. safety*). Zaufanie do systemu natomiast określa się przez postrzegany przez użytkownika poziom wiarygodności działania systemu. Dotyczy on również tych samych obszarów – wiarygodności danych powiązanych z kontrolą dostępu i niezawodności systemu powiązanej z zapewnieniem ciągłości działania. Im system jest bezpieczniejszy, tym większym zaufaniem użytkowników będzie się cieszył. Co oznacza, że wyższy poziom zaufania użytkowników do systemu uzyskuje się poprzez zapewnienie wyższego poziomu bezpieczeństwa systemu.



Rysunek 2 Relacja pomiędzy zaufaniem użytkownika do systemu, a bezpieczeństwem systemu informatycznego

Znane powszechnie jest powiedzenie, że najsłabszym ogniwem każdego systemu bezpieczeństwa jest człowiek [9]. Pahnila, Siponen i Mahmood wykazują powiązanie bezpieczeństwa systemu informatycznego z zachowaniami użytkowników [10]. Dowodzą oni, że sankcje wobec niestosujących się do polityki bezpieczeństwa użytkowników mają znikomy wpływ na poziom bezpieczeństwa. Właściwszym kierunkiem jest budowanie świadomości u użytkowników oraz wykorzystanie silnych mechanizmów zabezpieczających, lecz jak najmniej uciążliwych dla nich [11]. DeLone i MacLean wskazują sześć czynników z tym związanych: jakość i użyteczność systemu, zadowolenie użytkownika, wpływ polityki bezpieczeństwa na użytkownika i na organizację oraz jakość informacji przechowywanej w systemie [12]. Czynniki te decydują o poziomie zaufania użytkowników do systemu.

Warto zaznaczyć, że zdecydowane podnoszenie poziomu bezpieczeństwa systemu poprzez stosowanie wielu mechanizmów bezpieczeństwa, mimo że utwierdza użytkowników w przekonaniu o bezpieczeństwie, wpływa negatywnie na użyteczność samego systemu. Wpływ ten przejawia się w dwóch aspektach: wydajności systemu (każda dodatkowy mechanizm bezpieczeństwa wydłuża czas obsługi żądania użytkownika) oraz wygody użytkownika (np. jest on zmuszany do wielokrotnego uwierzytelniania różnymi metodami).

Kluczowym zagadnieniem jest więc opracowanie takich mechanizmów bezpieczeństwa, które w znacznym stopniu podnoszą poziom bezpieczeństwa, czyniąc to z minimalną stratą dla opisywanej wcześniej użyteczności systemu, zatem również zwiększają poziom zaufania użytkownika do systemu.

W ramach niniejszej rozprawy zaproponowano właśnie takie oryginalne mechanizmy bezpieczeństwa. Funkcjonują one w oparciu o szerokorozumiany kontekst działań oraz uwzględniają zaufanie systemu do użytkownika. Zagadnienia te są dokładnie opisywane w kolejnych rozdziałach rozprawy. Zastosowanie takich rozwiązań pozwala w znaczącym

stopniu podnieć poziom bezpieczeństwa, jednocześnie maksymalizując zaufanie użytkownika do systemu. Zostało to empirycznie wykazane w końcowej części rozprawy.

1.3 Motywacja i cel pracy

Kontrola dostępu użytkowników jest istotną funkcją systemów informatycznych. Oprócz klasycznych rozwiązań typu RBAC [13][14][15] stosuje się nowe podejścia polegające na uwzględnianiu kontekstu związanego z realizowanymi funkcjami systemu [16]. Obszar ten podlega badaniom od ponad 20 lat i ciągle jeszcze nie jest dobrze zdefiniowany [17]. Xu Feng i in. zaproponowali formalny model context-aware service-oriented role based access control (CSRBAC) [18]. Wiąże on użytkowników, role, uprawnienia, sesje użytkowników, konteksty oraz usługi operując w sferze usług sieciowych zgodnych ze standardem SOAP. Definiuje również procedurę aktywacji roli, która określa, czy dana rola/uprawnienie może zostać przyznane użytkownikowi. Dalszą modyfikacją tej pracy jest model CGRBAC (RBAC model for global services and context) [19]. Przewiduje on mapowanie ról specyficznych dla poszczególnych dostawców usług na tzw. globalne role – uniwersalne pomiędzy dostawcami usług. R. Bhatti, E. Bertino i A. Ghafoor zaproponowali model XML-based Generalized Temporal Role Based Access Control (X-GTRBAC) [20]. Definiuje on format zapisu warunków z wykorzystaniem notacji XML oraz umożliwia opracowanie algorytmu decydującego na podstawie kontekstu o przyznaniu dostępu. J. W. Woo i in. w [21] proponują rozszerzenie modelu RBAC o dynamiczne przydzielanie ról na podstawie zaufania pomiędzy usługami (w komunikacji usługa – usługa). Zaufanie to jest wyznaczane na podstawie reputacji usługi oraz satysfakcji z wcześniejszych jej wywołań. W ostatnich badaniach A. Gupta i in. skupia się na kontekście różnego typu wzajemnej bliskości/sąsiedztwie (*ang. proximity*) użytkowników. Analizowanych jest pięć typów bliskości: geograficzna, oparta o atrybuty użytkownika, uwzględniająca relacje w sieciach społecznościowych, wzajemnej interakcji online użytkowników oraz czasowej zależności pomiędzy realizowanymi działaniami [22]. Xuan Hung Le i in. rozważają rozszerzenie modelu RBAC o kontekst w dostępie do informacji podczas pracy grupowej (*ang. team collaboration*) i przepływie pracy (*ang. workflow*) [23]. Problematykę kontekstu w połączeniu z modelem RBAC w systemach publicznych typu e-government bazujących również na przepływach pracy (*ang. workflow*) rozważają Stevan Gostojić i in. [24].

Opisane rozwiązania nie wskazują jednak mechanizmów pozyskania kontekstu (metod analizy bieżącego kontekstu). Jedynie praca [25] sygnalizuje, że jest to trudne i skomplikowane. Określają one kontekst w sposób opisowy. Procedury dostępu natomiast są „punktowe" – nie wiążą ze sobą poszczególnych wywołań usługi i nie analizują relacji pomiędzy nimi (w powiązaniu z kontekstem). Dodatkowo rozwiązania te pozostają w sferze koncepcji, bez implementacji proponowanych rozwiązań w systemach rzeczywistych. Problemy te wymagają oddzielnego potraktowania.

W pracy wzięto pod uwagę systemy internetowe realizujące swoje funkcje poprzez dostępne w ich środowisku usługi informacyjne. Dla tej klasy systemów podano ogólny model bezpieczeństwa, jak i architekturę implementującą taki model. Proponowane rozwiązanie jest, podobnie jak w przytaczanych wyżej, rozwinięciem modelu RBAC. Wykorzystuje ono zarówno analizę kontekstu, jak też ocenę zaufania systemu do użytkownika. Stanowi oryginalne połączenie kilku przytaczanych wyżej pomysłów i całościowo tworzy dwa nowe modele: CoRBAC i TCoRBAC. Wykazano wzrost bezpieczeństwa systemów posiadających rozwiązania oparte o takie modele, co umożliwiło określenie pierwszej tezy rozprawy. Inną istotną sprawą w przedstawionym rozwiązaniu jest również fakt implementacji i wdrożenia tych modeli w praktyce na przykładzie systemu Moja PG [26] oraz przebadanie jego użyteczności.

Uwzględnienie kontekstu przy zapewnianiu bezpieczeństwa w internetowym systemie usługowym prowadzi do realizacji dodatkowych obliczeń i działań użytkownika. Wiąże się ze zmniejszeniem efektywności działania systemu, inaczej użyteczności z punktu widzenia użytkownika. Powstaje konieczność kompensacji tej niekorzystnej cechy poprzez wykorzystanie dodatkowych zasobów obliczeniowych, bądź uruchamianie dodatkowych mechanizmów bezpieczeństwa tylko w przypadkach, gdy zaufanie systemu do użytkowników jest niskie. Można wykazać, że użyteczność działania systemu z metodą TCoRBAC uwzględniającą zaufanie systemu do użytkowników jest bardzo zbliżona do użyteczności systemów wykorzystujących modele RBAC i CoRBAC. Na tej podstawie formułujemy drugą tezę rozprawy.

Opracowany system został zaimplementowany i wdrożony na Politechnice Gdańskiej na bazie autorskiej platformy IP^2. Jest to rozproszony system internetowy o architekturze usługowej. W ramach systemu uruchomiono wiele aplikacji wspierających działalność zarówno dydaktyczną, jak i naukową uczelni. Główny dostęp użytkownika odbywa się poprzez portal Moja PG, ale możliwe jest również korzystanie z usług za pomocą jednego z wielu zintegrowanych systemów (np. systemy wydziałowe łączące się i wymieniające dane z centralnym systemem). Całość wykonana jest w technologiach otwartego oprogramowania (*ang. open source*), w tym głównie Java Enterprise Edition [27]. Jak każdy system internetowy, rozpatrywany system charakteryzuje się cienkim klientem zlokalizowanym w przeglądarce WWW użytkownika (z dosyć intensywnie wykorzystywaną technologią AJAX i JavaScript). Aktualnie system posiada prawie 30 000 aktywnych użytkowników, a przetwarza dane dotyczące przeszło 80 000 osób. W związku z tym zapewnienie odpowiedniego bezpieczeństwa jest sprawą niezwykle ważną, ale również bardzo trudną. Dlatego zaproponowaną nową koncepcję opartą o modele CoRBAC i TCoRBAC praktycznie zrealizowano i wykorzystano podczas budowy e-uczelni na Politechnice Gdańskiej.

1.4 Tezy rozprawy

Rozpatrzmy system obsługi studentów zawierający newralgiczne dane istotne dla zapewnienia poprawnego przebiegu i oceny procesu kształcenia. Stąd motywacja do zwiększenia poziomu bezpieczeństwa systemu. Z drugiej strony liczba użytkowników tego systemu jest duża, a wykorzystanie złożonych i czasochłonnych mechanizmów zabezpieczeń prowadzi do zmniejszenia efektywności działania tego systemu. Konieczne jest więc kompromisowe podejście, akceptowalne w ciągłym działaniu internetowego systemu usługowego.

Przyjmijmy, że internetowy system usługowy ISS (*ang. Internet service system*) z mechanizmami RBAC, CoRBAC i TCoRBAC oznaczać będziemy poprzez odpowiednio ISS_{RBAC}, ISS_{CoRBAC} i $ISS_{TCoRBAC}$. Poziom bezpieczeństwa dla tych systemów oznaczamy odpowiednio $\beta(ISS_{RBAC})$, $\beta(ISS_{CoRBAC})$ oraz $\beta(ISS_{TCoRBAC})$. Podobnie przyjmijmy, że użyteczność tych systemów (rozumiana jako stosunek czasu poświęconego przez użytkownika na pracę użytkową τ_B w stosunku do całkowitego czasu poświęconego na pracę z systemem (użytkową i związaną z kontrolą bezpieczeństwa)) wynosi odpowiednio $\varepsilon(ISS_{RBAC})$, $\varepsilon(ISS_{CoRBAC})$ oraz $\varepsilon(ISS_{TCoRBAC})$. Czas użytkownika poświęcony na kontrolę bezpieczeństwa można podzielić na czas związany kontrolą bezpieczeństwa na podstawie przypisanych uprawnień τ_P, na analogiczną kontrolę z uwzględnieniem bieżącego kontekstu τ_C oraz czas poświęcony na realizację mechanizmów bezpieczeństwa wynikających z bieżącego zaufania systemu do użytkownika τ_T.

$$\varepsilon(\mathrm{ISS}) = \frac{\tau_B}{\tau_B+\tau_P+\tau_C+\tau_T} \quad (1.1)$$

Przy takich oznaczeniach tezy rozprawy można sformułować w sposób następujący:

Teza 1.

Poziom bezpieczeństwa internetowego systemu usługowego wykorzystującego kontrolę bezpieczeństwa zgodną z modelem CoRBAC jest nie mniejszy niż tego samego systemu w przypadku wykorzystania modelu RBAC. Analogicznie poziom bezpieczeństwa internetowego systemu usługowego wykorzystującego kontrolę bezpieczeństwa zgodną z modelem TCoRBAC jest nie mniejszy niż tego samego systemu w przypadku wykorzystania modelu CoRBAC, tzn.:

$$\beta(ISS_{RBAC}) \le \beta(ISS_{CoRBAC}) \le \beta(ISS_{TCoRBAC}) \quad (1.2)$$

Teza 2.

Użyteczność internetowych systemów usługowych wykorzystujących modele RBAC, CoRBAC i TCoRBAC jest zbliżona, a wahania nie przekraczają 10%, tzn.:

$$\varepsilon(ISS_{RBAC}) \approx \varepsilon(ISS_{CoRBAC}) \approx \varepsilon(ISS_{TCoRBAC}) \quad (1.3)$$

Dla udowodnienia tych tez w rozprawie wzięto pod uwagę modele tego typu systemów wraz z odpowiednimi mechanizmami zabezpieczeń i oceny zaufania do użytkowników. Na tej podstawie określono procedury wyznaczania poziomu bezpieczeństwa oraz wspomnianej wyżej użyteczności internetowych systemów usługowych. Opracowane procedury wykorzystano w rzeczywistym systemie Moja PG, który jest reprezentatywny dla internetowych systemów usługowych, co więcej posiada dużą liczbę użytkowników i jego działanie było dokumentowane w ostatnich 4 latach.

1.5 Układ pracy

Rozprawa doktorska w rozdziale 2 prezentuje trzy aspekty bezpieczeństwa systemów internetowych. Omawia zagadnienia polityki bezpieczeństwa, architektury systemu pod kątem bezpieczeństwa oraz zagrożenia, na jakie taki system jest narażony. Na tej podstawie określa również poziom zaufania do systemu (λ). Rozdział 3 definiuje kontekst oraz zastosowanie tego kontekstu w autorskich modelach bezpieczeństwa CoRBAC i TCoRBAC. Modele te są rozszerzeniem tradycyjnego modelu RBAC poprzez uwzględnienie kontekstu i zaufania do użytkownika. W rozdziale 4 przedstawiono praktyczną implementację zaproponowanych modeli kontekstowych w systemie działającym na Politechnice Gdańskiej. Zaprezentowano metody pozyskania i analizy bieżącego kontekstu. Przedstawiono również przykładowe kontekstowe mechanizmy bezpieczeństwa wdrożone w systemie. Rozdział 5 zawiera ocenę modeli CoRBAC i TCoRBAC pod kątem bezpieczeństwa oraz wpływu na wygodę użytkowników. Badania przeprowadzono na podstawie analizy danych zebranych z logów systemowych oraz raportów dostarczonych z audytów bezpieczeństwa, którym podlegał system. Na podstawie tej oceny wykazano zaproponowane tezy rozprawy. Rozdział 6 zawiera podsumowanie i wnioski co do kierunku dalszych badań.

Rozdział 2. Problemy bezpieczeństwa systemów internetowych

Opisano pojęcie bezpieczeństwa systemów internetowych poprzez główne atrybuty bezpieczeństwa takie jak identyfikacja i uwierzytelnianie, kontrola dostępu do zasobów, zabezpieczenie danych i komunikacji poprzez zapewnienie im poufności i integralności, niezaprzeczalność działań w systemie oraz jego dostępność. Następnie powiązano je z implementacją architektury systemu, polityką bezpieczeństwa oraz zestawem podstawowych zagrożeń dotyczących tego typu systemów. Przybliżono metodę oceny poziomu bezpieczeństwa systemów wykorzystującą analizę ryzyka wykrytych podatności. Stanowi ona podstawę szacowania poziomu zaufania użytkownika do systemu.

2.1 Atrybuty bezpieczeństwa

Bazując na triadzie C-I-A (*ang. confidentiality – integrity – availability*) [28] definiuje się tzw. atrybuty bezpieczeństwa. Klasyfikacja ta jest powszechnie stosowana i służy jako punkt wyjścia do większości rozważań dotyczących bezpieczeństwa systemów [29][30][31]. Zgodnie z tą klasyfikacją bezpieczeństwo systemu obejmuje pięć głównych elementów: identyfikację i uwierzytelnianie, kontrolę dostępu do zasobów, zabezpieczenie danych i komunikacji (zapewnienie poufności i integralności), niezaprzeczalność oraz dostępność systemu.

Dla potwierdzenia odpowiedniego poziomu bezpieczeństwa informacji w organizacji stosuje się certyfikację zgodności z normą ISO/IEC 27001 [32][33], która odbywa się na podstawie audytu przeprowadzanego przez akredytowane ośrodki. Procedura audytu jest ściśle określona i opiera się o listy kontrolne (*ang. checklists*). Warto zauważyć, że takie podejście umożliwia systematyczne spojrzenie na zagadnienie bezpieczeństwa w szerszym zakresie niż sam system komputerowy – dotyczy bezpieczeństwa informacji w całej organizacji, w której używany jest rozpatrywany system. Rozpoczynając prace nad wdrożeniem polityki bezpieczeństwa należy rozpocząć od analizy indywidualnych potrzeb organizacji w zakresie bezpieczeństwa i określeniem zakładanego jego poziomu [34]. Następnie opracowane założenia polityki bezpieczeństwa na poziomie organizacyjnym należy transformować m.in. na politykę kontroli dostępu w systemie.

Identyfikacja i uwierzytelnianie (*ang. authentication*) to proces wiarygodnego powiązania fizycznej osoby, hosta, modułu lub procesu z odpowiadającym mu obiektowi w systemie (*ang. identity*). W przypadku osób dotyczy to powiązania fizycznej osoby z odpowiadającą jej cyfrową tożsamością [35][36][37]. Identyfikacja odbywa się poprzez zadeklarowanie przez wykonującego akcję kim jest w systemie. Powiązanie takie musi być jednoznaczne, oparte o unikalną cechę bądź sztuczny identyfikator (np. w przypadku osób będzie to cyfrowa

tożsamość identyfikowana przez konto o konkretnym loginie). Samo uwierzytelnianie polega na przedstawieniu dowodu zadeklarowanego powiązania. W najprostszym przypadku jest to potwierdzenie loginu hasłem. Istnieją bardziej zaawansowane techniki uwierzytelniania, np. z wykorzystaniem kryptografii (podpisy cyfrowe, klucze SSH), bądź biometryki (odcisk linii papilarnych palca, obraz tęczówki oka, sekwencja DNA). Operacja ta powinna być wykonana co najmniej przy każdym dostępie do systemu. Niektóre polityki bezpieczeństwa wymagają dodatkowego ponowienia tej operacji w trakcie interakcji z systemem, przeważnie przed wykonaniem szczególnie chronionej operacji.

Kontrola dostępu do zasobów systemu określa zbiory możliwych operacji do wykonania przez użytkownika na określonych zbiorach zasobów. Prawidłowo skonstruowana polityka kontroli dostępu zabezpiecza przed nieautoryzowanym bezpośrednim, bądź pośrednim dostępem do zasobów. Tak rozumianą kontrolę określa się mianem autoryzacji (*ang. authorization*). Bardzo częstym sposobem implementacji tego elementu polityki bezpieczeństwa jest utworzenie powiązania w systemie tożsamości z możliwymi do wykonania przez nią operacjami, zakresem działania oraz wątkiem (sesja użytkownika, proces lub wątek systemowy) [38]. Następnie każda aktywność w ramach danego wątku jest weryfikowana pod kątem zgodności ze stworzonymi wcześniej powiązaniami. Tak więc kontrola dostępu jest związana z wątkiem (sesją użytkownika, procesem lub wątkiem systemowym) i może być zmienna w czasie.

Zagadnienie zabezpieczenia danych i komunikacji rozpatruje się z następujących punktów widzenia:

- integralności danych poprzez uniknięcie niedopuszczalnej ich modyfikacji,
- autentyczności pochodzenia danych,
- poufności przechowywanych danych,
- poufności przetwarzanych danych w pamięci oraz podczas ich transmisji, np. przez sieć komputerową.

Aby móc uznać dane za wiarygodne muszą być spełnione dwa warunki dotyczące zapewnienia integralności i autentyczności danych. Integralność danych najczęściej uzyskuje się poprzez jednokierunkowe funkcje skrótu takie jak CRC, MD5, SHA-1, SHA-256, itp. Oprócz samych danych równolegle udostępnia się ich sumy kontrolne (*ang. checksum*) lub tzw. skróty (*ang. hash*). W ten sposób użytkownik systemu może zweryfikować po swojej stronie otrzymane dane wykonując identyczną operację wyliczenia skrótu i porównując otrzymany wynik z udostępnionym przez system. Zabezpieczenie to pozwala na wykrycie niedopuszczalnej modyfikacji lub uszkodzenia danych zarówno w czasie składowania w systemie, jak i w czasie transmisji.

W pewnych przypadkach wiarygodność danych wymaga zapewnienia, że pochodzą one z tego źródła, z którego są deklarowane. Mówi się wtedy o autentyczności źródła ich pochodzenia. Uzyskuje się to również poprzez udostępnianie skrótów z danych

jednoznacznie je identyfikujących. Może to być osiągnięte poprzez publikację ich w niezależnym, dobrze określonym kanale komunikacyjnym. Jednak najczęściej wykorzystuje się w takich przypadkach podpis cyfrowy - podpisywany jest skrót danych i w tej formie udostępniany razem z tymi danymi. W pierwszej kolejności odbiorca weryfikuje poprawność podpisu na skrócie, a następnie porównuje wyliczoną wartość skrótu z wartością podpisaną. Dzięki zastosowaniu kryptografii asymetrycznej potwierdzana jest autentyczność wszelkiego rodzaju dokumentów cyfrowych w postaci znakowej (np. xml) lub binarnej (np. skany) oraz wiadomości cyfrowych (np. e-maile). Nawet skompilowane oprogramowanie w postaci binarnej może być także podpisywane. Mechanizm ten wykorzystano m.in. przy nadawaniu uprawnień apletom Javy w przeglądarkach WWW, które są uruchamiane w tzw. piaskownicach (*ang. sandbox*) – weryfikowany jest wówczas cyfrowy podpis biblioteki.

Poufność danych zapewnia się najczęściej poprzez zastosowania różnego rodzaju algorytmów kryptograficznych. Dzięki takiemu rozwiązaniu tylko posiadacz klucza jest w stanie rozszyfrować daną porcję danych i ją właściwie wykorzystać. Dla wszystkich innych użytkowników dane te są bezużyteczne. To podejście można uznać również za jeden ze sposobów kontroli dostępu do nich, który jest jednak bardzo złożony obliczeniowo, co w efekcie oznacza zmniejszenie wydajności pracy systemu. W niektórych zastosowaniach poufność składowanych danych uzyskuje się poprzez wykorzystanie jednokierunkowych funkcji skrótu (*ang. hash*), na przykład gdy nie jest wymagana możliwość ponownego przekształcenia informacji do postaci jawnej, gdy wystarczające jest porównanie wyliczonej wartości skrótu. Ten sposób zapewniania poufności składowanych danych wykorzystuje się w systemach uwierzytelniania. Hasła składowane są w postaci niejawnej, dzięki czemu nikt nie jest w stanie podejrzeć oryginalnego hasła. Wówczas moduł uwierzytelniający dokonuje porównania wartości skrótu, aby upewnić się, że otrzymane dane uwierzytelniające (*ang. credentials*), tj. login i hasło, są właściwe. Ponieważ funkcje skrótu podatne są na tzw. kolizje (można podać inny ciąg znaków, z którego otrzyma się taką samą wartość skrótu) oraz w celu utrudnienia ataków siłowych (*ang. brute-force*) z wykorzystaniem tablic tęczowych (*ang. rainbow tables*) zaleca się dodawanie do wejściowego ciągu znaków losowego ciągu znaków tzw. soli (*ang. salt*).

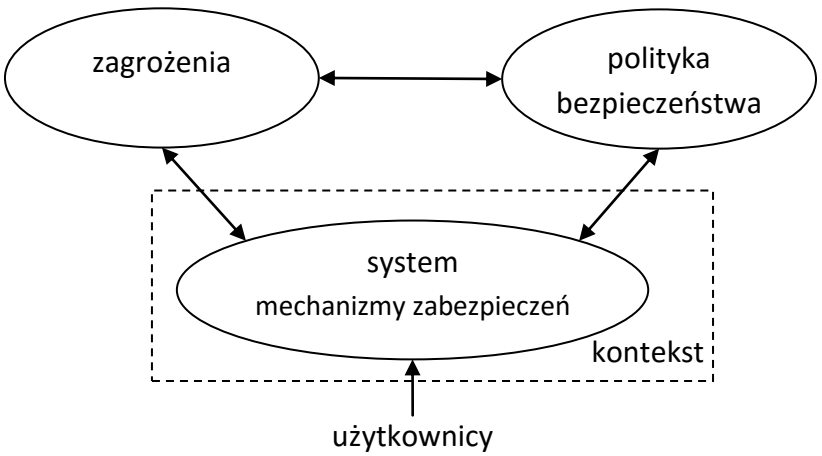
Zapewnienie poufności danych przetwarzanych w pamięci sprowadza się to do zapewnienia odpowiedniej izolacji wątków systemu w dostępie do zmiennych. Istnieje szereg problemów związanych z tym obszarem. Najczęściej spotykane to przepełnienie bufora (*ang. buffer overflow*) i dostęp do pamięci zarezerwowanej dla innego procesu systemowego oraz niewystarczająca izolacja poszczególnych systemów uruchamianych w ramach środowiska wirtualizowanego (*ang. guest system*). Poza tym przesyłanie ogromnej ilości danych, czy komunikatów przez sieć stwarza szansę podsłuchania transmisji i ujawnienia przesyłanych informacji. W tym przypadku stosuje się różnego rodzaju tunele kryptograficzne, np. SSH, VPN, SSL. Niezależnie od tego można przesyłać dane w postaci niejawnej, co zostało wspomniane wyżej.

Niezaprzeczalność akcji to proces zapewnienia niemożliwości wyparcia się przez aktora wykonanej przez niego czynności w systemie. Jest to bardzo skomplikowane zagadnienie. Może być realizowane od najprostszej formy bezpiecznego składowania logów wykonanych operacji poprzez kryptografię klucza publicznego do wykorzystania podpisów cyfrowych. Oczywiście pozostaje nadal problem, czy fizyczny aktor jest rzeczywiście tym, w imieniu kogo się uwierzytelnił. Dotyczy to zarówno bardzo wysokiego poziomu (kto się uwierzytelnił do systemu, np. ktoś poznał login i hasło innej osoby lub wykorzystuje jego podpis cyfrowy, ponieważ ma dostęp do klucza prywatnego), jak i niższego poziomu dotyczącego zagwarantowania separacji i/lub uwierzytelniania pomiędzy wątkami funkcjonującymi w systemie.

Niezawodność dostępu i responsywność systemu na założonym z góry poziomie jest podstawą współdziałania systemu z użytkownikami. Warto podkreślić, że ataki typu DOS (*ang. denial of service*) i DDOS (*ang. distributed denial of service*) mają na celu przeciążenie systemu do takiego stopnia, że nie będzie on osiągalny dla autoryzowanych użytkowników, co równoważne jest z naruszeniem polityki bezpieczeństwa. Dlatego użytkownicy uprawnieni do dostępu do danych i usług muszą ją zachować w każdych okolicznościach.

2.2 Zapewnianie bezpieczeństwa w systemie

Zapewnianie bezpieczeństwa w systemie wymaga uwzględnienia potencjalnych zagrożeń, przyjęcia właściwej polityki bezpieczeństwa i wdrażanie odpowiadających jej mechanizmów w tym systemie. Te trzy najistotniejsze aspekty bezpieczeństwa systemowego są przedstawione na rys. 3. Są one mocno ze sobą powiązane.



Rysunek 3 Aspekty bezpieczeństwa w rozważanym systemie internetowym

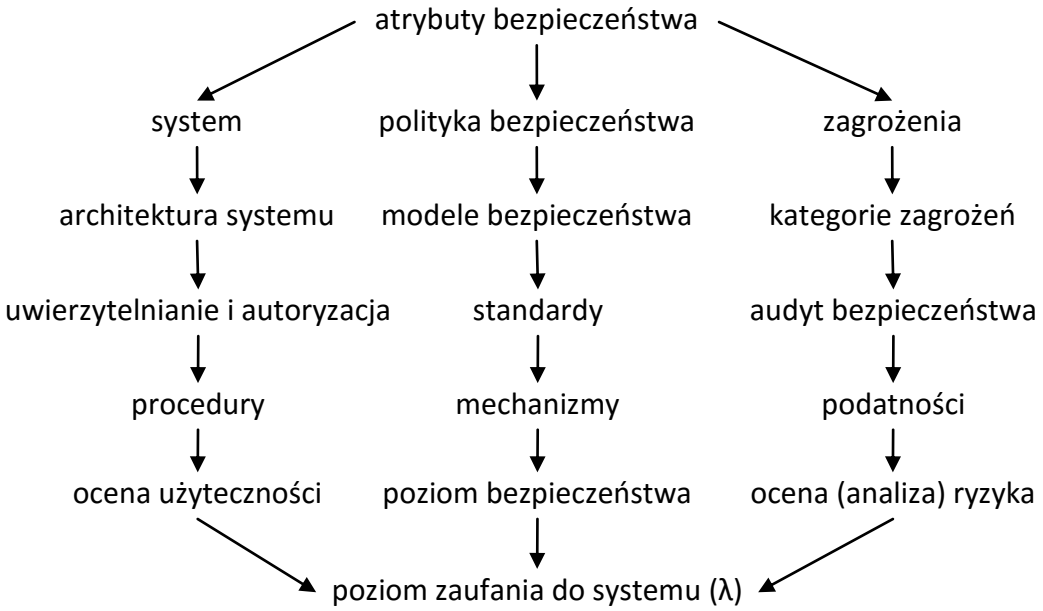
Oczywiste jest, że każdy system powinien udostępnić użytkownikom zakładaną funkcjonalność. Zgodnie z założeniem rozważane są systemy o architekturze zorientowanej na usługi (SOA) [39], a w szczególności systemy internetowe dostępne w globalnej sieci.

Są to coraz częściej systemy rozproszone o kliencie zlokalizowanym w przeglądarce internetowej. W ostatnich latach obserwuje się tendencję odchodzenia od modelu chudego klienta (*ang. thin client*) na rzecz coraz bardziej rozbudowanego, jednak ciągle zlokalizowanego w przeglądarce WWW, co daje dużą przenaszalność rozwiązań [40]. Wiąże się to z intensywnym wykorzystaniem JavaScriptu oraz technologii AJAX [40]. W chwili obecnej proste pakiety biurowe są w pełni dostępne jako aplikacje uruchamiane w przeglądarce WWW (patrz: GoogleDocs [41], MS Office 365 [42]). Tendencja ta wkracza również na pole specjalistycznego oprogramowania, jak np. Adobe Photoshop [43], itp.

Każdy system internetowy działa w określonym środowisku zewnętrznym charakteryzującym się odpowiednimi właściwościami technologicznymi czy ludzkimi. Co więcej użytkownicy i inne systemy/usługi mogą wchodzić w interakcję z rozpatrywanym systemem. Istotny jest czas zajścia zdarzenia, lokalizacja użytkowników, bieżące obciążenie systemu, liczba realizowanych połączeń, wykorzystywane urządzenia jak i ich stan bieżący [44]. Wszystkie te uwarunkowania tworzą kontekst działania systemu. Analiza tego kontekstu i jego wpływu na bezpieczeństwo jest tematem niniejszej rozprawy. Kolejne rozdziały szczegółowo analizują to zagadnienie.

Każdy działający system komputerowy narażony jest na zagrożenia (*ang. threats*). Podlegają one kategoryzacji i analizie pod kątem ryzyka skutku niewłaściwego funkcjonowania systemu. Dotyczy to w szczególności systemów dostępnych w internecie, więc posiadających potencjalnie dużą liczbę normalnych użytkowników, jak i świadomie atakujących intruzów. Nie istnieją mechanizmy zabezpieczeń o 100% skuteczności, najistotniejsze jest szacowanie i minimalizowanie ryzyka, jakie niesie przełamanie zabezpieczeń (wykorzystanie podatności). Zatem należy tak projektować zabezpieczenia systemu, aby maksymalnie ograniczyć konsekwencje niezgodnego z oczekiwanym działania systemu. Kluczowa w tym procesie jest analiza ryzyka [45]. Warto zwrócić uwagę, że zagrożenia występują w obu obszarach bezpieczeństwa – zarówno w sferze kontroli dostępu (*ang. security*), jak i zapewnienia ciągłości działania (*ang. safety*) [46]. W pracy ograniczono się tylko do rozważań w zakresie kontroli dostępu.

Ostatnim aspektem jest przyjęta polityka bezpieczeństwa, która określa poziom akceptowalnego ryzyka oraz wskazuje sposoby ochrony przed zagrożeniami. Zgodnie z tzw. cyklem Deminga raz zbudowana polityka powinna być na bieżąco monitorowana i modyfikowana (modernizowana) [47]. Norma ISO/IEC 27001 opisuje model „Planuj – Wykonuj – Sprawdzaj – Działaj" [32]. Określa on odpowiednio: inicjalne utworzenie polityki bezpieczeństwa, wdrożenie i eksploatację mechanizmów bezpieczeństwa, ich monitorowanie i przegląd oraz ich utrzymanie i doskonalenie (modernizowanie).



Rysunek 4 Dekompozycja trójki: system, polityka bezpieczeństwa oraz zagrożenia na poszczególne poziomy szczegółowej analizy

Opisaną powyżej trójkę ściśle ze sobą powiązanych aspektów bezpieczeństwa można rozpatrywać na pięciu poziomach szczegółowości analizy. Rysunek 4 przedstawia dekompozycję każdego z aspektów bezpieczeństwa na niższe, korespondujące ze sobą poziomy. Poziom drugi składa się z ogólnej architektury systemu, modeli bezpieczeństwa i kategorii zagrożeń. Kolejny poziom jest uściśleniem powyższego poprzez wskazanie wybranej architektury uwierzytelniania i autoryzacji, zastosowanych standardów bezpieczeństwa oraz ustandaryzowanego audytu bezpieczeństwa. Czwarty poziom stanowią elementy związane z implementacją, tzn. takie jak: procedury systemowe, zastosowane mechanizmy bezpieczeństwa oraz wykryte w systemie podatności. Na ostatnim poziomie następuje ocena użyteczności, inaczej wygody użytkownika, poziomu bezpieczeństwa oraz analiza ryzyka wykrytych podatności. Ocena ta wpływa bezpośrednio na wynikowy poziom zaufania użytkownika do systemu (λ). Poszczególne wymienione elementy są szczegółowo opisane w kolejnych sekcjach niniejszego rozdziału rozprawy.

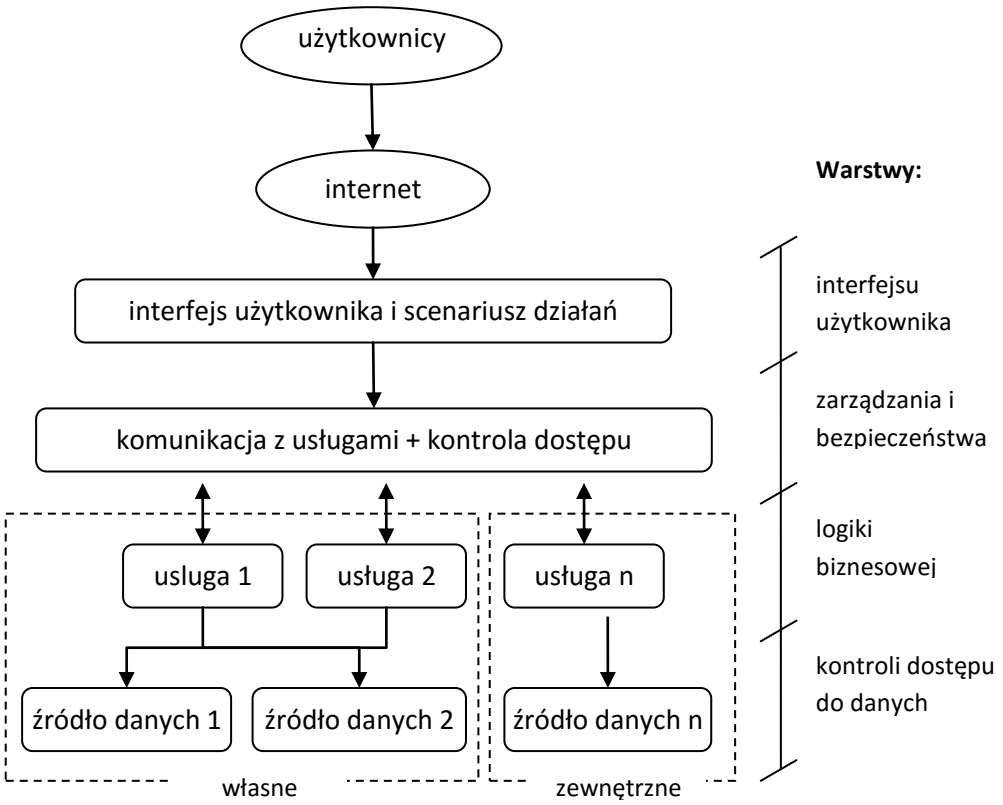
W dalszej części rozprawy przyjęto tego typu podejście (przedstawione na rysunku 4) i na przykładzie rzeczywiście funkcjonującego systemu Moja PG podano propozycję nowych mechanizmów bezpieczeństwa i oceniono poziom ryzyka i zaufania do tak zmodyfikowanego systemu.

2.3 Internetowy system usługowy

Każdy system internetowy charakteryzuje jego architektura oraz wykorzystane standardy i technologie jego wykonania. Zagadnienie całościowo jest bardzo szerokie, w związku z czym dalsze rozważania zostaną zawężone do aspektów istotnych pod kątem bezpieczeństwa tego systemu. Skupiono się również głównie na rozproszonych systemach internetowych o charakterze usługowym.

2.3.1 Architektura systemu

Przyjęta architektura systemu internetowego stanowi połączenie architektury zorientowanej na usługi (SOA) z lekko zmodyfikowaną, typową dla systemów webowych architekturą trójwarstwową [48][49]. W niej można wyróżnić mechanizmy zarządzania i bezpieczeństwa oraz kontroli dostępu do danych. Ilustruje to Rysunek 5.



Rysunek 5 Architektura rozważanego systemu internetowego

Jest to dosyć często spotykana architektura w dużych rozproszonych systemach o charakterze usługowym. Funkcje uwierzytelniania i autoryzacji w jej funkcjonalnych warstwach są wplecione w różnego typu mechanizmy bezpieczeństwa. Możliwych jest kilka wariantów rozwiązań, które przedstawiono w kolejnych sekcjach tego rozdziału.

2.3.2 Wymagania w zakresie uwierzytelniania i autoryzacji

Uwierzytelnianie i autoryzacja to dwa kluczowe zagadnienia w obszarze bezpieczeństwa dotyczące zabezpieczenia przed nieautoryzowanym dostępem do systemu (*ang. security*). Wybór modelu uwierzytelniania i autoryzacji odpowiedniego dla przyjętej architektury systemu determinuje kształt całej warstwy bezpieczeństwa systemu i jest bardzo trudny do zmiany w późniejszym etapie, co implikuje istotność tej decyzji przy projektowaniu systemu.

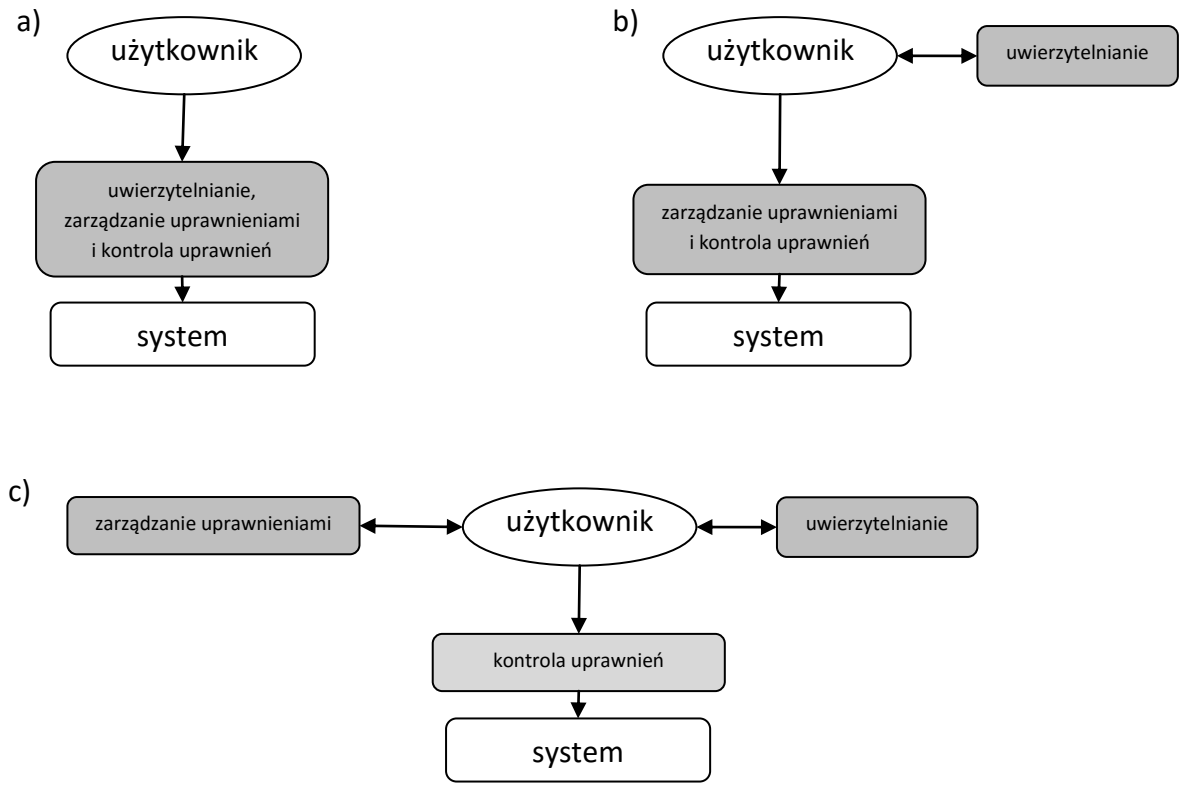
Obserwując aktualne kierunki rozwoju systemów rozproszonych zdecydowanie zauważa się tendencję do wydzielania funkcji uwierzytelniających poza projektowany system (patrz Rysunek 6b i Rysunek 6c). Ma to kilka zalet: m.in. użytkownik musi pamiętać mniej poświadczeń (haseł) oraz możliwe jest wykorzystanie mechanizmu jednokrotnego logowania (ang. *Single Sign On*) [50][35][51]. Mechanizm taki ma istotny wpływ na zwiększenie prywatności użytkownika [6] bowiem to on decyduje jakiemu systemowi, jakie informacje o sobie chce udostępnić. Dodatkowo zwiększa się bezpieczeństwo, ponieważ dane uwierzytelniające podawane są tylko w zaufanych systemach uwierzytelniających, co zmniejsza ryzyko wycieku ich w różnych systemach biznesowych w wyniku błędów implementacji. Poza tym dane przechowywane są tylko w jednym miejscu, o bezpieczeństwo którego należy szczególnie zadbać.

Istnieją dwa sposoby wykorzystania tak wyniesionego mechanizmu autoryzacji. Pierwszy z nich ukierunkowany jest na wygodę użytkownika – często dla systemów biznesowych nie jest istotne powiązanie użytkownika z konkretną osobą fizyczną. Istotne jest jedynie każdorazowe potwierdzenie, że to ten sam użytkownik. Dlatego korzystają one z zewnętrznych usług uwierzytelniania. Sam użytkownik zaś nie musi pamiętać kolejnego loginu i hasła. W trakcie logowania wskazuje on adres usługi uwierzytelniającej, za pomocą której chciałby potwierdzić swoją tożsamość. Najbardziej znanym rozwiązaniem w tym zakresie jest protokół OpenID [52]. Mechanizmy z nim związane stosowane są przede wszystkim w portalach internetowych, gdzie i tak zakłada się pewną dopuszczalną anonimowość użytkowników.

W przypadku systemów narzędziowych (wspierających działalność organizacji, przechowujących jej dane), gdzie istotne jest kto się uwierzytelnił, nie można korzystać z dowolnego systemu uwierzytelniającego – użytkownik ma do wyboru jedną lub kilka wskazanych, zaufanych usług uwierzytelniających. Dominujące w tym modelu protokoły komunikacji to OAuth [53], CAS [54] oraz ponownie, wspomniany już wcześniej, OpenID [52]. Warto zwrócić uwagę, że analogiczne centralne usługi uwierzytelniające budowane są przez instytucje rządowe. Doskonałym przykładem jest tu polska elektroniczna platforma usług administracji publicznej ePUAP [55].

W momencie, gdy usługi uwierzytelniające zostały już wyniesione i zcentralizowane, możliwe są 3 warianty zarządzania uprawnieniami użytkownika w systemach [51]. Pierwszym z nich jest model tradycyjny – każdy system (usługa biznesowa) posiada swój pełny, autonomiczny

mechanizm zarządzania i kontroli uprawnień. W ten sposób zmiana uprawnień jednego użytkownika musi być wykonana we wszystkich systemach, do których miał on dostęp. W przypadku dużej ilości takich systemów jest to mało wygodne i może prowadzić do powstawania niespójności w konfiguracji realizujących politykę bezpieczeństwa. Drugi model to wyniesienie uprawnień poza system/usługę biznesową. W takim przypadku uprawnienia nadawane są przez zaufane centrum lub centra np. w postaci podpisanych certyfikatów poświadczających określone uprawnienie [56]. Zmiana uprawnień użytkownika odbywa się w jednym miejscu, więc pozostają one spójne. Rozwiązanie takie implikuje jednak potrzebę implementacji mechanizmów weryfikujących aktualność przedstawianych poświadczeń do uprawnień. Przykładem mógłby być mechanizm weryfikacji ważności wystawionych certyfikatów w oparciu o protokół OCSP (*ang. Online Certificate Status Protocol*) [57]. Ilustruje to Rysunek 6c.



Rysunek 6 Modele wyniesionego i zcentralizowanego uwierzytelniania i zarządzania autoryzacją: (a) tradycyjny, (b) z wyniesionym uwierzytelnianiem, (c) z wyniesionym uwierzytelnianiem i zarządzaniem uprawnieniami

Istnieje jeszcze rozwiązanie mieszane (dwustopniowe), gdzie zarządzanie „grubymi" uprawnieniami odbywa się w sposób wyniesiony i zcentralizowany (uprawnienia odpowiadające cechom użytkownika), natomiast poszczególne uprawnienia szczegółowe, specyficzne dla danego systemu, zarządzane są tylko w tym systemie. Taka architektura wydaje się być rozsądnym kompromisem, szczególnie w przypadku rozwiązań zróżnicowanych funkcjonalnie i rozległych obszarowo.

2.4 Polityka bezpieczeństwa

Politykę bezpieczeństwa systemu definiuje się na kilku poziomach szczegółowości postępując stopniowo z góry do dołu. Na najwyższym poziomie wyróżniamy politykę kontroli dostępu. Następnie określany jest model bezpieczeństwa, który tą politykę realizuje. Kolejnym krokiem jest wybór standardów i ostatecznie implementacja mechanizmów bezpieczeństwa, które będą zapewniać odpowiedni poziom bezpieczeństwa.

2.4.1 Zasady kontroli dostępu

Polityka bezpieczeństwa z perspektywy kontroli dostępu jest zbiorem reguł wyznaczanych przez organizację, które musi spełniać użytkownik otrzymujący dostęp do zasobu [38]. Wyróżnia się 4 podstawowe typy polityk kontroli dostępu. Poniżej pokrótce scharakteryzowano każde z nich.

Attribute-based access control (ABAC) polega na nadaniu użytkownikowi zestawu atrybutów. Obiekt określa, jakie atrybuty musi posiadać użytkownik, aby uzyskać do niego dostęp. Przykładowo mogą być to strony dostępne dla osób powyżej 18 lat. Warto zwrócić tu uwagę, że w tym przypadku nie jest konieczne uwierzytelnienie użytkownika – może być on anonimowy.

Discretionary access control (DAC) – ACL opiera się na polityce dostępu określanej przez właściciela obiektu. To on określa kto i na jakich zasadach ma dostęp do obiektu (*ang. Access Control Lists*). Zakłada się więc, że każdy obiekt ma swojego właściciela. Przykładem są tu systemy plików wykorzystywane w systemach operacyjnych, np. Windows, Linux lub innych.

Mandatory access control (MAC) opiera się na założeniu zgoła odwrotnym niż DAC. Tu system przypisuje zarówno użytkowników, jak i obiekty (zasoby systemu) do poszczególnych poziomów wrażliwości / tajności, które stanowią bariery zabezpieczeń. Wskazany użytkownik ma dostęp tylko do obiektów sklasyfikowanych na tym samym poziomie, bądź niższych. Przykładem takiego rozwiązania może być mechanizm utajniania dokumentów wywiadowczych.

Role-based access control (RBAC) jest obecnie najczęściej stosowany w nowoczesnych systemach internetowych. Zakłada przypisanie użytkownikom odpowiednich ról, które przekładają się na poszczególne uprawnienia do wykonywania operacji w systemie. Należy tu zaznaczyć, dla odróżnienia od DAC, że to nie właściciel obiektu, a system określa jakie uprawnienia, więc i role, są wymagane do wykonania operacji. Model jest bardzo elastyczny, pozwala na wygodne budowanie szerokiej gamy ról, umożliwia hierarchiczne ich zagnieżdżanie, co przekłada się również na wygodę administrowania. O zmianie uprawnień użytkownika decyduje administrator systemu.

2.4.2 Modele bezpieczeństwa

Sposób zapewniania odpowiedniego poziomu bezpieczeństwa zawiera się w polityce bezpieczeństwa. Określa ona wytyczne jak powinien zachować się system i jakie procedury powinny być wykorzystywane w poszczególnych przypadkach. Informuje również jasno i spójnie co i na jakim poziomie system powinien zabezpieczać.

Odwzorowaniem polityki bezpieczeństwa na konkretny system informatyczny jest model bezpieczeństwa, który odwzorowuje jej abstrakcyjne założenia w zbiór jednoznacznych reguł. Definiuje wykorzystane modele danych i techniki niezbędne dla zapewnienia zakładanego poziomu bezpieczeństwa.

Model bezpieczeństwa jest też narzędziem, za pomocą którego można opisać w sposób usystematyzowany, precyzyjny i niekiedy formalny jedną lub więcej polityk bezpieczeństwa. Takie podejście umożliwia przeprowadzanie analiz założonych polityk bezpieczeństwa ukazując jej słabe i silne strony [38].

Na przestrzeni lat powstało wiele różnych modeli bezpieczeństwa. Poniżej przytoczono i krótko scharakteryzowano tylko kilka najbardziej znanych i popularnych.

Access Control Matrix model został zaproponowany przez B. W. Lampsona [58]. Analogiczny model został przedstawiony przez Harrisona, Ruzzo i Ullmana [59]. Zakłada on stworzenie macierzy, w której kolumnami są wyróżnione zasoby systemu (np. dane, aplikacje), natomiast w wiersze odpowiadają podmiotom (np. użytkownikom) operującym na tych zasobach. W przecięciu kolumny i wiersza umieszczone są wszystkie przypisane danej parze uprawnienia (np. odczyt, modyfikacja, usunięcie, uruchomienie, itp.). Z modelu tego wynikają dwa widoki: każdy wiersz tworzy profil dostępu (*ang. Access profile*) podmiotu, natomiast każda kolumna tworzy listę kontrolną dostępu do danego obiektu (*ang. Access-control list*). To właśnie Access-control List (ACL) jest najczęściej wykorzystywaną strukturą do implementacji matryc kontroli dostępu.

Lattice-Based Access Control (LBAC) to model stworzony przez D. E. Denninga [60]. Zbudowany jest on na bazie kraty będącej częściowo uporządkowanym zbiorem (*ang. partially ordered set - poset*). Krata ta służy do zdefiniowania poziomów bezpieczeństwa, do których mogą być przypisywane obiekty (np. dane, komputery, aplikacje) oraz do których mają dostęp poszczególne podmioty (np. osoby, grupy, organizacje). Podmiot może otrzymać dostęp do obiektu tylko wtedy, gdy poziom bezpieczeństwa podmiotu jest większy bądź równy poziomowi bezpieczeństwa obiektu [61].

Bell—LaPadula (BLP) model został stworzony dla zapewnienia poufności w systemach rządowych i militarnych (o tzw. statycznej polityce bezpieczeństwa) [62]. Jest on zgodny z LBAC. Zakłada on przypisanie obiektom i podmiotom etykiet (*ang. labels*) / klas bezpieczeństwa. Poszczególne klasy obejmują zakres od najbardziej wrażliwych (np. „Top

secret”) do najmniej wrażliwych (np. „niezdefiniowane” lub „publiczne”). Dodatkowo zdefiniowane są dwie reguły:

- podmiot może mieć dostęp do obiektów o poziomie bezpieczeństwa równym lub niższym niż poziom podmiotu (read-down property),
- podmiot może tworzyć tylko takie obiekty, które charakteryzują się poziomem bezpieczeństwa równym lub wyższym niż poziom podmiotu (write-up policy).

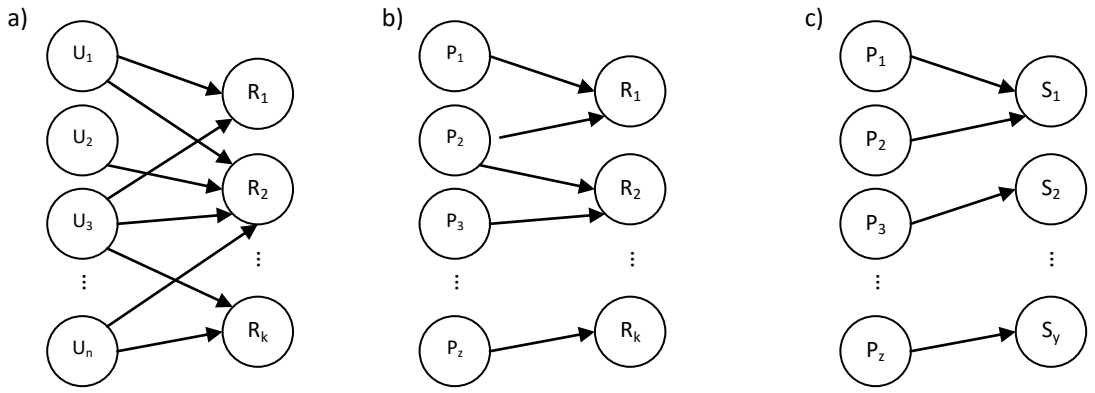
Biba model (Biba Integrity Model) został stworzony przez K. J. Biba [63] dla zapewnienia integralności danych (obiektów). Podobnie jak w modelu BLP obiektom i podmiotom przypisuje się klasy integralności. Dodatkowo obowiązują dwie reguły przeciwstawne regułom modelu BLP: read-up property i write-down policy. Przykładem może być taktyka wojenna spisana przez dowództwo niedostępna dla osób o niższej szarży, jedynie dostępna wyższemu dowództwu (read-up property). Natomiast rozkazy wynikające z taktyki wydawane mogą być tylko „z góry do dołu”. W ten sposób zapewniona jest integralność misji.

Brewer—Nash (Chinese Wall) model [64] został skonstruowany celem zapewnienia kontroli dostępu, która niweluje sytuację, w której pojedynczy podmiot ma w tym samym czasie dostęp do kilku obiektów będących w konflikcie interesów (np. w organizacjach komercyjnych). Przykładowo informacje wrażliwe o banku B nie powinny być dostępne dla podmiotu, który posiada informacje wrażliwe o banku A i mu doradza. Warto zaznaczyć, że w przeciwieństwie do modeli BLP i Biba model Chinese Wall stosowany jest w środowiskach o dynamicznie zmiennych politykach dostępu.

Podstawowy model RBAC opiera się o przyporządkowania dotyczące użytkowników oraz ich ról, wiąże uprawnienia z rolami, a także odnosi się do powiązania uprawnień i operacji [38][65][16]. Określa się je następująco:

- UR (UŻYTKOWNICY x ROLE)
- PR (UPRAWNIENIA x ROLE)
- PS (UPRAWNIENIA x OPERACJE/USŁUGI)

Rysunek 7 przedstawia tego typu relacje pomiędzy przytoczonymi wyżej użytkownikami (U), rolami (R), uprawnieniami (P) oraz operacjami/usługami (S). Jest on bardzo wygodny do zarządzania uprawnieniami w skomplikowanych systemach wykorzystywanych przez wiele użytkowników.



Rysunek 7 Relacje w modelu RBAC pomiędzy: a) użytkownikami (U) i rolami (R), b) uprawnieniami (P) i rolami (R), c) pomiędzy uprawnieniami (P), a operacjami/usługami (S)

Hierarchiczny RBAC jest roszrzeniem podstawowego modelu RBAC. Zakłada on możliwość hierachicznego budowania drzewka ról. Dzięki temu w bardziej intuicyjny sposób można zamodelować i zarządzać rolami odpowiadającymi poszczególnym stanowiskom w danej organizacji [66].

Stosując taksonomię realizacji poszczególnych polityk bezpieczeństwa otrzymujemy zgodnie z [38] następujące przyporządkowanie modeli:

DAC:

- Access Control Matrix model [58]
- Harrison—Ruzzo—Ullman model [59]

MAC:

- Lattice-Based Access Control (LBAC) information flow model [60][61]
- Bell—LaPadula confidentiality model [62]
- Biba integrity model [63]
- Brewer—Nash (Chinese Wall) model [64]

RBAC:

- Basic Role-Based Access Control (RBAC) model [65][16]
- Hierarchical RBAC model [66]

Model RBAC jest neutralny pod względem polityk bezpieczeństwa - umożliwia zamodelowanie zarówno polityki MAC jak i DAC. Z tego powodu, oraz ze względu na wygodę jego użytkowania w praktyce, jest często wykorzystywany. W związku z tym został on wybrany jako podstawa rozwiązań przedstawionych w rozprawie.

2.4.3 Standardy w obszarze bezpieczeństwa

Wykorzystanie standardów w obszarze bezpieczeństwa ukierunkowane jest na dwa kluczowe cele. Pierwszym z nich jest umożliwienie komunikacji pomiędzy różnymi systemami, często implementowanymi w różnych technologiach, pomijając standardy dotyczące tylko jednej technologii. Drugim natomiast jest uniknięcie błędów w implementacji poszczególnych polityk i modeli bezpieczeństwa. W praktyce polega to najczęściej na wyniesieniu warstwy bezpieczeństwa poza budowane oprogramowanie i wykorzystanie zabezpieczeń oferowanych przez kontener aplikacji. W większości przypadków jest to wystarczające rozwiązanie, które skraca czas potrzebny na wytworzenie systemu oraz minimalizuje ryzyko wprowadzenia błędów zabezpieczeń.

Standardów dotyczących bezpieczeństwa systemów opracowano bardzo wiele. Poniżej wymieniono kilka propozycji reprezentujących poszczególne typy standardów. Do pierwszego typu należą standardy pomocnicze wykorzystywane niezależnie od technologii i służące jako pewne elementy innych standardów. Do drugiego typu zalicza się standardy umożliwiające bezpieczne porozumiewanie się systemów budowanych w różnych technologiach. Do tego typu zalicza się Web Services Security (WSS) Standards Framework [67], który jest rozszerzeniem standardu komunikacji WebService SOAP, oraz np. HOTP [68] dla protokołów REST. Trzecim typem jest zestaw standardów specyficznych dla konkretnych technologii, przeważnie umożliwiających wyniesienie implementacji warstwy bezpieczeństwa poprzez delegację jej na kontener aplikacji, w której system jest uruchamiany [69].

Poniżej przedstawiono przykłady wyżej wymienionych standardów. Standardy pomocnicze, szczegółowo opisane w [67], są następujące:

- SSL/TLS [70][71] – zapewnia poufność i integralność komunikacji oraz uwierzytelnianie serwera i klienta. Wykorzystuje szyfrowanie asymetryczne i certyfikaty X.509.
- XML Encryption [72] – definiuje sposób szyfrowania zawartości elementu XML
- XML Signature [73] – definiuje XML-ową składnię dla podpisów cyfrowych. Wykorzystywany jest m.in. przez SOAP, SAML.
- SAML [74] – definiuje format danych wymienianych podczas uwierzytelniania i autoryzacji pomiędzy zaangażowanymi stronami: przeważnie dostawcą tożsamości (*ang. identity provider*) oraz systemem oferującym usługi (*ang. service provider*).

Web Services Security (WSS) Standards Framework ([75]) jest rozszerzeniem standardu SOAP dedykowanym usługom sieciowym (*ang. web services*). Obejmuje praktycznie wszystkie aspekty bezpieczeństwa w kontekście usług sieciowych takie jak: komunikację oraz zarządzenie uwierzytelnianiem i autoryzacją. Poszczególne jego składowe przedstawia Tabela 1.

Tabela 1 Standardy składowe „Web Serivce Security Standards Framework”

WS-SecureConversation	WS-Federation	WS-Authorization
WS-Policy	WS-Trust	WS-Privacy
WS-Security		
SOAP foundation		

Warto przy tym zauważyć, że standard SOAP ze względu na wykorzystanie standardu XML, posiada bardzo duży czasowy narzut komunikacyjny. Dlatego jest on sukcesywnie wypierany przez standard JSON, którego narzut komunikacyjny jest 3-krotnie mniejszy [76][77]. Przykładem takiego standardu jest JSON-RPC 2.0 [78][79]. Popularność zdobywają również webserwisy RESTful [80][81]. W związku z tym wymagane jest opracowanie odpowiednich standardów bezpieczeństwa dla tych rozwiązań.

Przykładem standardu specyficznego dla konkretnej technologii jest Java Authentication and Authorization Service (JAAS) w technologii Java [82]. Jest to przykład standardu wspominanego wcześniej rozwiązania z delegacją na kontener aplikacji zarządzania uwierzytelnianiem i autoryzacją.

2.4.4 Mechanizmy bezpieczeństwa

Mechanizmy kontroli dostępu uwzględniają metody, narzędzia lub wytyczne wynikające z przyjętej polityki bezpieczeństwa [38]. Mechanizmy takie mogą być zautomatyzowane, w pełni zaszyte w system (najlepszy wariant ze względu na eliminację pomyłek i braku bezwzględnej obiektywności człowieka). Mogą być również częściowo lub całkowicie manualne, poza systemem (w postaci określonych wytycznych postępowania) – w takim przypadku realizowane przez fizyczne osoby, np. administratora systemu.

Wybór zastosowanych mechanizmów zależy w znacznym stopniu również od specyfiki systemu/aplikacji. Inne będą stosowane dla systemów internetowych, a inne np. dla urządzeń mobilnych [83][84][85]. W pierwszym przypadku stosuje się warstwową kontrolę zgodną z modelem RBAC, w drugim zaś mechanizmy tzw. piaskownicy (*ang. sandbox*) [86].

Rozpatrzmy politykę zmiany hasła do konta. Przyjmijmy, że jego zmiana musi być realizowana np. nie częściej niż 3 dni oraz co najmniej raz na 1 miesiąc. Pierwszy warunek zabezpiecza przed próbą powrotu użytkownika do poprzedniego hasła poprzez dwukrotną zmianę hasła (na nowe i kolejno na pierwotne) tego samego dnia. Drugi warunek narzucony jest przez politykę realizującą wymagania ustawy o ochronie danych osobowych (konieczność zmiany hasła minimum raz w miesiącu) [87].

Dla systemów uwzględniających kontekst można rozwijać mechanizmy kontekstowej kontroli dostępu. Dla przytoczonego przykładu można uwzględnić kontekst związany z porą dnia czy lokalizacją użytkownika, co będzie analizowane w dalszej części pracy.

2.5 Możliwe zagrożenia

Zagrożenia powodujące zmniejszenie bezpieczeństwa systemów komputerowych (*ang. threats*) zmieniają się wraz z rozwojem technologii. Oprogramowanie wykorzystywane w systemach jest wytwarzane coraz szybciej kosztem obniżenia jego jakości. Dodatkowo rośnie poziom skomplikowania funkcji systemowych. Przekłada się to na charakter i liczbę nieświadomie wprowadzanych tzw. podatności (*ang. vulnerability*) tzn. przyczyn osłabiających bezpieczeństwo systemu [88]. Ponieważ dane zawarte w systemach stanowią coraz większą wartość, pojawia się wiele szkodliwego oprogramowania wykorzystującego te podatności celem przełamania zabezpieczeń. Zaobserwować można pewnego rodzaju wyścig pomiędzy osobami szukającymi nowych podatności w systemach, a zespołami je eliminującymi. Okoliczności te powodują, że sytuacja jest bardzo zmienna. Jednym z przykładów jest liczba codziennie identyfikowanych nowych wirusów komputerowych. Chcąc zapanować nad tak dynamicznie zmienną sytuacją potrzebne jest prowadzenie odpowiednio skategoryzowanej ewidencji wykrytych zagrożeń i ich typów. Różne instytucje zajmujące się omawianym zagadnieniem tworzą swoje własne klasyfikacje i katalogi. Najczęściej używane katalogi wykrytych zagrożeń to:

- US-CERT Vulnerability Notes [89]
- MITRE Common Vulnerabilities and Exposures (CVE) [90]
- National Vulnerability Database (NVD) [91]
- SecurityFocus Vulnerability Database and BugTraq mail list [92]
- Open Source Vulnerability Database (OVSDB) [93]
- biuletyny bezpieczeństwa dostawców oprogramowania [67]

Możliwości ich zostały szerzej opisane w [67]. Większość z nich jest dostępna za darmo, część na zasadach komercyjnych. Dotyczą różnych systemów komputerowych. CVE podlega kategoryzacji wg. klasyfikacji opracowanej przez tą samą organizację - MITRE Common Weakness Enumeration (CWE) [94]. Pozostałe odwołują się również do tej samej klasyfikacji lub OWASP Top Ten [95]. Często też odwołują się do obu wspomnianych i opcjonalnie również własnej.

Ze względu na powszechną dostępność systemów internetowych zorientowanych na usługi, dotyczą je wszystkie typy zagrożeń, przy czym zorientowane są one na wyróżnione komponenty, czy istotne właściwości systemu.

Identyfikacja poszczególnych podatności w systemie odbywa się najczęściej w trakcie przeprowadzanego ustandaryzowanego audytu bezpieczeństwa (testów penetracyjnych). Audyt taki wykonywany jest przez zewnętrznego audytora w oparciu o przytoczone klasyfikacje i katalogi zagrożeń. Dodatkowym wsparciem może być OWASP Testing Guide [96]. Kluczowe obszary każdego audytu bezpieczeństwa systemów internetowych to:

1. Konfiguracja systemu i jego wdrożenia (*ang. deployment*) na środowisko produkcyjne,
2. Ustawienia zarządzania użytkownikami,
3. Proces uwierzytelniania,
4. Proces autoryzacji,
5. Zarządzanie sesją użytkownika,
6. Walidacja danych wejściowych,
7. Obsługa błędów,
8. Stosowane standardy kryptograficzne,
9. Logika procedur biznesowych,
10. Środowisko uruchomieniowe (przeglądarka WWW użytkownika).

Podczas audytu każda funkcjonalność systemu jest testowana pod kątem wszystkich typów podatności. Dla przeciętnego systemu oferującego co najmniej 10 funkcjonalności sprawdzane jest ponad 1000 potencjalnych podatności.

Bardzo często wykryte podatności można sklasyfikować zgodnie z kilkoma różnymi klasyfikacjami. Dlatego też poszczególne kategoryzacje posiadają mapowania i odwołania do innych kategorii. Poniżej przedstawiono dwie kategoryzacje zagrożeń: STRIDE [31] i OWASP Top Ten [95]. Pierwsza z nich jest bardziej teoretyczna, ale bardzo dobrze prezentuje relację pomiędzy poszczególnymi kategoriami zagrożeń i odpowiadającym im atrybutom bezpieczeństwa. Druga natomiast klasyfikuje zagrożenia najczęściej występujące w odniesieniu do systemów internetowych.

2.5.1 Klasyfikacja zagrożeń STRIDE

Opracowana przez Microsoft klasyfikacja STRIDE [29] składa się z sześciu kategorii zagrożeń: Spoofing, Tampering, Repudiation, Information disclosure, Denial of Service i Elevation of privilege. Określenia angielskie zachowano celowo, ponieważ są one bardziej zwięzłe i precyzyjniejsze niż ich polskie odpowiedniki. Dodatkowo nazwa klasyfikacji (STRIDE) jest akronimem pierwszych liter tych zagrożeń. Poniżej krótko scharakteryzowano poszczególne kategorie tej klasyfikacji na podstawie [31].

T1: Spoofing to grupa ataków polegających na podszywaniu się pod innego użytkownika lub element systemu. Efekt ten jest osiągany przez celowe niepoprawne użycie protokołów komunikacji lub umieszczenie w sieci spreparowanych pakietów danych. Na poziomie protokołów TCP/IP najczęstszymi atakami się IP spoofing oraz ARP spoofing. W przypadku protokołu HTTP najczęściej spotykany jest referrer spoofing (podmiana nagłówka Referrer). Powszechnym atakiem jest spoofing adresu email – wykorzystywany przez wszystkie narzędzia rozsyłające spam – adres nadawcy jest tak podmieniany, aby przekonać odbiorcę o pochodzeniu wiadomości z zupełnie innego źródła niż jest w rzeczywistości (szczególnie bardziej zaufanego). Lukę tą wykorzystuje się szczególnie w atakach ukierunkowanych na nakłonienie do podania haseł bądź instalacji zawirusowanego oprogramowania. Również

w technologiach mobilnych możliwy jest spoofing nadawcy wiadomości (przy wykorzystaniu przygotowanych do tego celu bramek internetowych) lub informacji o nawiązującym połączenie.

T2: Tampering to grupa ataków polegających na nieautoryzowanej modyfikacji danych wprowadzonych przez autoryzowanego użytkownika. Najlepszym sposobem ochrony przed tą grupą zagrożeń jest stosowanie zabezpieczeń kryptograficznych takich jak podpisywane cyfrowo wiadomości lub skróty wiadomości oraz szyfrowane kanały komunikacji i magazyny danych.

T3: Repudiation to możliwość wyparcia się użytkownika wykonanej przez niego akcji w systemie. Zabezpieczenie tej sfery zagrożeń odbywa się poprzez zebranie odpowiedniego materiału dowodowego. Zależnie od stopnia krytyczności operacji może to być:

- fakt niemożności wykonania kolejnej akcji w systemie bez wykonania wcześniejszej (np. skorzystanie z serwisu bez akceptacji regulaminu),
- podpis cyfrowy złożony na dokumencie – tylko właściciel klucza prywatnego może złożyć prawidłowy podpis (nikt inny).

T4: Information disclosure to zagrożenie ujawnienia wrażliwych danych. Dotyczy złamania poufności danych składowanych, przetwarzanych lub przesyłanych. Najczęstszym sposobem ochrony przed tą rodziną zagrożeń jest stosowanie szyfrowanej komunikacji (protokół HTTPS oraz tunele kryptograficzne), szyfrowanie składowanych danych oraz zabezpieczenia przed wyciekami pamięci.

T5: Denial of Service lub **Distributed Denial of Service** sprowadza się do przeciążenia systemu informatycznego lub usługi sieciowej w celu uniemożliwienia jej prawidłowego działania. Realizowane jest przez wysycanie zasobów procesorowych, połączeniowych (np. wyczerpanie limitu wolnych gniazd WWW) lub pamięciowych (np. zapełnienie pamięci operacyjnej albo systemu plików). W efekcie system lub usługa są niedostępne dla użytkowników. Skuteczniejsze są ataki rozproszone z wykorzystaniem tzw. botnetów komputerów zombie – zainfekowanych maszyn, których właściciele nie są tego nawet świadomi. Trudno jest się przed takim atakiem obronić, ponieważ źródło jest bardzo rozproszone i często „wymieszane" z rzeczywistymi użytkownikami systemu.

T6: Elevation of privilege (Privilege escalation) to grupa zagrożeń polegających na zdobyciu uprawnień do zasobów niedostępnych w normalnym trybie. Sprowadza się do wykonania nieautoryzowanych operacji w systemie. Zagrożenia te wiążą się najczęściej z błędami popełnionymi podczas implementacji systemu. Wyróżnia się dwa typy tego rodzaju zagrożeń:

- vertical privilege escalation (privilege elevation) – ma miejsce, gdy użytkownik o niższych uprawnieniach zdobywa dostęp do funkcjonalności albo zasobów

zarezerwowanych dla użytkowników o wyższych uprawnieniach (np. zwykły użytkownik zdobywa dostęp do funkcji administracyjnych),

- horizontal privilege escalation – polega na dostępie na tym samym poziomie uprawnień do zasobów innego użytkownika (np. użytkownik uzyskuje dostęp do internetowego konta bankowego innego użytkownika).

Szczegółowy opis przytoczonych ataków można znaleźć w [97][98][96].

Klasyfikacja STRIDE wiąże poszczególne kategorie zagrożeń z odpowiadającym im atrybutom bezpieczeństwa (patrz pkt. 2.1) – prezentuje to Tabela 2.

Tabela 2 Mapowanie poszczególnych kategorii zagrożeń STRIDE na atrybuty bezpieczeństwa

Kategorie zagrożeń		**Atrybut bezpieczeństwa**
T1	**S**poofing	Uwierzytelnianie
T2	**T**ampering	Integralność
T3	**R**epudiation	Niezaprzeczalność
T4	**I**nformation disclosure	Poufność
T5	**D**enial of Service	Dostępność
T6	**E**levation of privilege	Autoryzacja

Poszczególne zagrożenia mogą występować w różnych obszarach pracy systemu. Klasyfikacja STRIDE [29] definiuje następujące obszary:

- Data Flows - przepływy danych (połączenia sieciowe, połączenia wewnątrz-systemowe)
- Data Stores - magazynowane dane (pliki, bazy danych, rejestry, ...)
- Processes - procesy biznesowe
- Interactors - aktorzy (ludzie, web-serwisy, serwery)

Tabela 3 przedstawia relację pomiędzy zagrożeniami, a obszarami, w których mogą wystąpić. Wyraźnie widać, że większość dotyczy kilku obszarów, wymaga więc wielopoziomowych mechanizmów zabezpieczeń.

Tabela 3 Relacja pomiędzy kategoriami zagrożeń, a obszarami, w których mogą wystąpić

Kategorie zagrożeń		Data Flows	Data Stores	Processes	Interactors
T1	Spoofing			X	X
T2	Tampering	X	X	X	
T3	Repudiation			X	X
T4	Information disclosure	X	X	X	
T5	Denial of Service	X	X	X	
T6	Elevation of privilege			X	

2.5.2 Klasyfikacja OWASP Top Ten

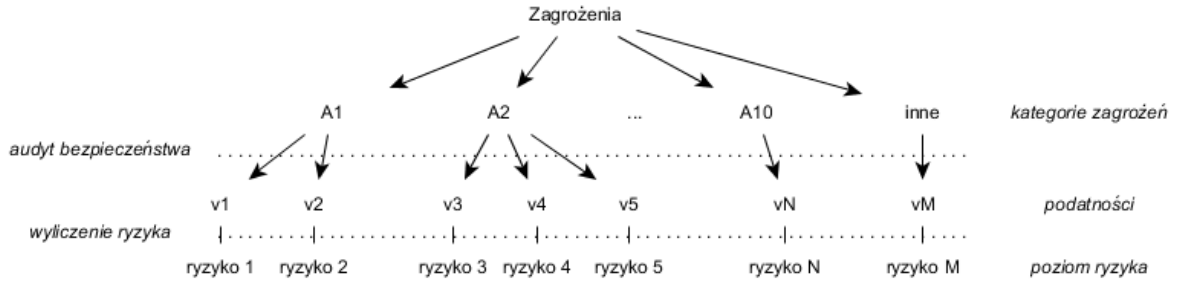
Klasyfikacja Open Web Application Security Project Top Ten (OWASP Top Ten) [95] wydaje się być obecnie najlepszą w odniesieniu do systemów internetowych. Budowana i aktualizowana jest ona na bieżąco przez wielu specjalistów z zakresu bezpieczeństwa. Co najważniejsze jest dostępna na otwartej licencji Creative Commons [99], a więc powszechna i może być na bieżąco korygowana przez bardzo dużą liczbę specjalistów, co znacząco wpływa na jej jakość. OWASP Top Ten składa się z 10 kategorii zagrożeń. Tabela 4 przedstawia ich skrócony opis. Szczegółowy można znaleźć w [95].

Tabela 4 Skrócony opis kategorii zagrożeń OWASP Top Ten

Kategorie zagrożeń	Opis
A1-Injection	Wszelkiego typu modyfikacje oryginalnych zapytań/poleceń SQL, OS, czy LDAP występujące, gdy przesłane przez atakującego dane nie są właściwie przetworzone i są interpretowane przez system jako element zapytania lub polecenia.
A2-Broken Authentication and Session Management	Wszystkie błędy związane z nieprawidłową implementacją procesu uwierzytelniania, przechowywania danych uwierzytelniających i obsługą identyfikatorów sesji.
A3-Cross-Site Scripting (XSS)	Podatności XSS występują, gdy system internetowy przetwarza lub przechowuje i wysyła do przeglądarki WWW użytkownika niezaufane dane, które następnie są interpretowane przez przeglądarkę użytkownika jako kod HTML lub JavaScript.
A4-Insecure Direct Object References	Podatność występuje, gdy system udostępnia dane na podstawie przesłanego identyfikatora obiektu bez weryfikacji, czy zgłaszający ma do nich uprawnienia.
A5-Security Misconfiguration	Bardzo często domyślna konfiguracja oprogramowania (frameworki, serwery aplikacji, serwery baz danych, …) zawiera niebezpieczne ustawienia ułatwiające prace implementacyjne, ale stanowiące w eksploatacji zagrożenie dla systemów produkcyjnych.
A6-Sensitive Data Exposure	Obejmuje wszystkie zagrożenia związane ze zbyt słabym zabezpieczeniem wrażliwych danych takich jak dane kart kredytowych, hasła, itp.
A7-Missing Function Level Access Control	Występuje, gdy kontrola do poszczególnych funkcjonalności odbywa się jedynie poprzez ukrycie na interfejsie linku lub przycisku, natomiast spreparowane odpowiednie żądanie nie jest już weryfikowane po stronie serwera.
A8-Cross-Site Request Forgery (CSRF)	Ataki opierające się na przygotowaniu takiej strony WWW, która zawiera ukryty kod akcji odwołującej się do innego systemu. Ten szkodliwy kod jest wykonany bez wiedzy użytkownika, gdy użytkownik odwiedzający spreparowaną stronę jest jednocześnie zalogowany w tym drugim systemie.

A9-Using Components with Known Vulnerabilities	Poszczególne dostępne powszechnie biblioteki powinny być wykorzystywane w najnowszych dostępnych wersjach, w których usunięte są wykryte wcześniej podatności, ponieważ istnieją narzędzia próbujące automatycznie wykonać ataki bazując na podstawie list odkrytych podatności.
A10-Unvalidated Redirects and Forwards	Występuje, gdy system generuje żądania redirect lub forward do stron na podstawie niesprawdzonych właściwie danych. Żądania takie powinny być generowane tylko na podstawie słownikowych wartości parametrów lub określonego ściśle ich wzorca.

Rysunek 8 przedstawia ogólną topologię zagrożeń, podatności i ryzyka jakie one niosą. Procedura wyznaczania ryzyka poszczególnych podatności i powiązanego z nimi poziomu bezpieczeństwa została opisana w kolejnym punkcie rozdziału.

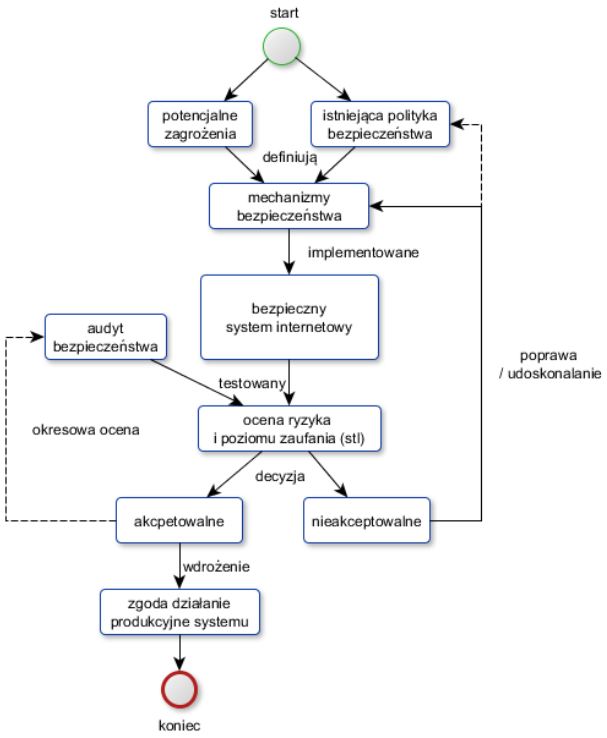


Rysunek 8 Ogólna topologia zagrożeń, podatności i ryzyka z nimi związanego

Jak już wspomniano klasyfikacja OWASP Top Ten pokrywa większość zagrożeń specyficznych dla systemów internetowych. Może się jednak zdarzyć, że zostanie wykryta podatność nie mieszcząca się w przytaczanej klasyfikacji. Mimo, że jest to bardzo rzadki przypadek, to po analizie związanej z okolicznościami jego wykrycia oraz sposobem implementacji sytemu może nastąpić aktualizacja przyjętej klasyfikacji.

2.5.3 Analiza ryzyka wykrytych podatności

Każda wykryta podatność (*ang. vulnerability*) powinna być analizowana pod kątem ryzyka jakie niesie jej wykorzystanie przez potencjalnego atakującego [4][100]. Na podstawie tak przygotowanego i na bieżąco aktualizowanego zestawienia określa się priorytety w usuwaniu podatności [25][45]. Warto zwrócić uwagę, że już na etapie tworzenia polityki bezpieczeństwa powinien zostać określony próg akceptowalnego poziomu ryzyka [32][101]. Główna trudność w analizie ryzyka to dobór odpowiednich metryk [102][103][104].



Rysunek 9 Proces "utwardzania" zabezpieczeń systemu internetowego

Rysunek 9 prezentuje proces „utwardzania" zabezpieczeń systemu internetowego składający się z dwóch cykli. Pierwszy z nich obejmuje poprawę i udoskonalanie zaimplementowanych mechanizmów bezpieczeństwa. Drugi z nich skupia się na okresowej ocenie zaufania do systemu (λ) – innymi słowami czy zidentyfikowany poziom ryzyka mieści się w założonych granicach przyjętego poziomu bezpieczeństwa systemu. Oba cykle mogą być powtarzane wielokrotnie, aż do osiągnięcia oczekiwanego poziomu zaufania do systemu (λ). Poniżej opisano dwa kluczowe działania tego procesu: ocena ryzyka związanego z wykrytymi podatnościami oraz szacowanie poziomu zaufania do systemu (λ).

Powstało wiele metod oraz technik analizy i oceny ryzyka wykrytych podatności. Warto wymienić 5 szerzej opisanych w [67]:

- Microsoft Security Response Center Security Bulletin Severity Rating System [105]
- US-CERT Vulnerability Metric [106]
- Microsoft DREAD threat-risk ranking model [107]
- SANS Critical Vulnerability Analysis Scale [108]
- Common Vulnerability Scoring System (CVSS) [109][110][108]

W chwili obecnej głównie rozwijana jest metoda CVSS (aktualnie w wersji 2). CVSS v2 składa się z 3 grup metryk:

- bazowych określających podstawowe cechy charakterystyczne dla podatności, które są niezmienne w czasie i niezależne od środowiska użytkownika/organizacji,

- czasowych – ich wartość zmienia się w czasie życia podatności od czasu jej ujawnienia do zastosowania zabezpieczeń,
- środowiskowych, które zależą od środowiska konkretnego użytkownika/organizacji (np. straty ekonomiczne utraty produktywności lub dochodu).

Pierwsza grupa składa się z 6 metryk: Access Vector (AV), Access Complexity (AC), Authentication (Au), Confidentiality Impact (CI), Integrity Impact (II), and Availability Impact (AI). Pierwsze trzy określają jak podatność jest dostępna i czy potrzebne są specjalne warunki do jej wykorzystania. Trzy metryki wpływu mierzą niezależnie jak dana podatność wpłynęłaby na poufność, integralność i dostępność systemu i danych w nim przetwarzanych. Przykładowo podatność może powodować częściową stratę w integralności i dostępności danych, ale nie redukcję poufności [111]. Pozostałe dwie grupy metryk są opcjonalne oraz bardzo specyficzne w zależności od organizacji i mogą zmieniać się w czasie. Dlatego nie zostały uwzględnione w niniejszej rozprawie.

CVSS v2 definiuje 3 dyskretne wartości dla każdej metryki. Szczegółowy ich opis można znaleźć w [111]. Dla bazowych metryk są to odpowiednio:

- Access Vector (AV): Local (AV:L), Adjacent Network (AV:A), Network (AV:N)
- Access Complexity (AC): High (AC:H), Medium (AC:M), Low (AC:L)
- Authentication (Au): Multiple (Au:M), Single (Au:S), None (Au:N)
- Confidentiality Impact (CI): None (CI:N), Partial (CI:P), Complete (CI:C)
- Integrity Impact (II): None (II:N), Partial (II:P), Complete (II:C)
- Availability Impact (AI): None (AI:N), Partial (AI:P), Complete (AI:C)

Metryki te służą do wyznaczenia numerycznej wartości oraz tzw. wektora CVSS v2, które wskazują na poziom ryzyka ρ, jaki związany jest z ew. wykorzystaniem podatności (luki). Wartość ta jest wyliczana za pomocą kalkulatora dostarczanego przez National Institute of Standards and Technology [112] z wykorzystaniem opracowanego przez jego pracowników wzoru podanego w równaniach 2.1 – 2.4.

$$\rho = (0{,}6 * \text{impact} + 0{,}4 * \text{exploitability} - 1{,}5) * f(\text{impact}) \quad (2.1)$$

$$\text{impact} = 10{,}41 * \left(1 - (1 - \text{CI}) * (1 - \text{II}) * (1 - \text{AI})\right) \quad (2.2)$$

$$\text{exploitability} = 20 * \text{AC} * \text{Au} * \text{AV} \quad (2.3)$$

$$f(\text{impact}) = \begin{cases} 0, & \text{impact} = 0 \\ 1{,}176, & \text{impact} \neq 0 \end{cases} \quad (2.4)$$

Tabela 5 przedstawia wartości numeryczne odpowiadające poszczególnym wartościom metryk bazowych. W celu wyznaczenia poziomu ryzyka należy wykorzystać wzory 2.1 – 2.3 oraz wartości podane w tabeli 5. Czynności te ułatwia wykorzystanie dostarczonego przez NIST kalkulatora [112].

Tabela 5 Wartości numeryczne dla poszczególnych wartości metryk bazowych

AV		AC		Au		CI		II		AI	
AV:L	0,395	**AC:H**	0,35	**Au:M**	0,45	**CI:N**	0	**II:N**	0	**AI:N**	0
AV:A	0,646	**AC:M**	0,61	**Au:S**	0,56	**CI:P**	0,275	**II:P**	0,275	**AI:P**	0,275
AV:N	1	**AC:L**	0,71	**Au:N**	0,704	**CI:C**	0,66	**II:C**	0,66	**AI:C**	0,66

Dla każdego systemu można wyznaczyć liczbę podatności z zerowym (podatność nie została zidentyfikowana w systemie), niskim, średnim i wysokim poziomem ryzyka, odpowiednio: φ_Z, φ_L, φ_M i φ_H. Tabela 6 przedstawia powiązanie pomiędzy poziomem ryzyka ρ, klasą tego poziomu (zgodnie z CVSS v2) oraz czynnikiem wpływu na zaufanie użytkownika do systemu.

Tabela 6 Powiązanie pomiędzy poziomem ryzyka, krytycznością podatności oraz zaufaniem do systemu

poziom ryzyka ρ	**0**	**(0 – 4.0)**	**[4.0 – 7.0)**	**[7.0 – 10.0]**
poziom ryzyka (klasa)	zerowe	niskie	średnie	wysokie
czynnik wpływu na zaufanie do systemu	1.0	0.6	0.3	0.1

Równanie 2.5 definiuje proponowany sposób wyznaczenia poziom zaufania do systemu (λ).

$$\lambda = \frac{\varphi_Z + \varphi_L * 0.6 + \varphi_M * 0.3 + \varphi_H * 0.1}{\varphi_Z + \varphi_L + \varphi_M + \varphi_H} \quad (2.5)$$

gdzie:

φ_Z – oznacza liczbę potencjalnych podatności, które były sprawdzane podczas audytu, ale nie zostały wykryte w systemie

φ_L /φ_M /φ_H – oznacza liczbę podatności wykrytych podczas audytu o klasie odpowiednio niskiej / średniej / wysokiej.

Minimalna możliwa wartość λ zgodnie z tabelą 6 wynosi 0,1 (w przypadku, gdy wszystkie sprawdzane potencjalne podatności zostały zidentyfikowane w audytowanym systemie), natomiast maksymalna wynosi 1,0 (w przypadku, gdy żadna ze sprawdzanych potencjalnych podatności nie została zidentyfikowana w audytowanym systemie). Poziom zaufania użytkownika do systemu λ jest proporcjonalny do poziomu bezpieczeństwa systemu β, więc na potrzeby niniejszej rozprawy przyjęto, że jest on tożsamy z poziomem bezpieczeństwa systemu β.

Zaprezentowana w tym rozdziale procedura oceny poziomu ryzyka ρ i zaufania do systemu λ została wykorzystana w rozdziale 5 do oceny zaproponowanego w rozdziale 3 modelu.

Rozdział 3. Modele CoRBAC i TCoRBAC

Zdefiniowano kontekst w szerokim znaczeniu uwzględniającym zachowania użytkownika. Zaprezentowano możliwe sposoby uwzględnienia kontekstu w polityce bezpieczeństwa oraz w modelach bezpieczeństwa. Zaproponowano formalny opis zorientowanych kontekstowo modeli bezpieczeństwa będących rozszerzeniem modelu RBAC. Modele te o nazwach CoRBAC i TCoRBAC uwzględniają szeroki zakres mechanizmów bezpieczeństwa oraz umożliwiają ich efektywną implementację. Zostały zaprezentowane i zweryfikowane na międzynarodowym forum specjalistów bezpieczeństwa [113].

3.1 Opis modelu RBAC

Model RBAC jest powszechnie wykorzystywany w dużych systemach informatycznych ze względu na jego akceptowalną efektywność w modelowaniu złożonych reguł polityk bezpieczeństwa [114]. Ponieważ jest on dodatkowo neutralny pod względem polityk bezpieczeństwa (patrz pkt. 2.4.2), został wybrany przez autora rozprawy jako najbardziej elastyczny do rozbudowy o parametry kontekstu i zaufanie do użytkownika. W rozprawie proponuje się model CoRBAC (Context-oriented Role Based Access Control model), jako konsekwencję rozwoju tradycyjnego modelu bezpieczeństwa RBAC, a następnie model TCoRBAC (Trust and Context-oriented Role Based Access Control model) będący rozwinięciem modelu CoRBAC o zaufanie do użytkownika. Tak więc model TCoRBAC zawiera w sobie model CoRBAC.

Model RBAC zbudowany jest w oparciu o relacje pomiędzy następującymi zbiorami artefaktów [38]:

Przyjmijmy, że mamy k użytkowników danego systemu internetowego. Zapisujemy to w sposób następujący:

$$\mathrm{U} = \{\mathrm{u}_1, \mathrm{u}_2, \mathrm{u}_3, \dots, \mathrm{u}_\mathrm{k}\},\ \mathrm{u}_\mathrm{i} \in \mathrm{U},\ \mathrm{i} = 1, 2, \dots, \mathrm{k} \quad (3.1)$$

Role każdego z użytkowników $u_i \in U$ określa się ze zbioru R:

$$\mathrm{R} = \{\mathrm{r}_1, \mathrm{r}_2, \mathrm{r}_3, \dots, \mathrm{r}_\mathrm{l}\},\ \mathrm{r}_\mathrm{j} \in \mathrm{R},\ \mathrm{j} = 1, 2, \dots, \mathrm{l} \quad (3.2)$$

Co oznacza, że każdy użytkownik u_i może posiadać role przynależące do zbioru $R_i \subseteq R$. Z kolei uprawnienia użytkowników definiuje się poprzez zbiór P:

$$\mathrm{P} = \{\mathrm{p}_1, \mathrm{p}_2, \mathrm{p}_3, \dots, \mathrm{p}_\mathrm{m}\},\ \mathrm{p}_\mathrm{x} \in \mathrm{P},\ \mathrm{x} = 1, 2, \dots, \mathrm{m} \quad (3.3)$$

Oznacza to, że każdy użytkownik u_i posiada uprawnienia z podzbioru $P_i \subseteq P$. Uprawnienia te umożliwiają wykonanie odpowiednich operacji systemowych. Pełne możliwości systemu określa zbiór *S*, zaś użytkownik u_i może realizować operacje/usługi należące do $S_i \subseteq S$:

$$\mathrm{S} = \{\mathrm{s}_1, \mathrm{s}_2, \mathrm{s}_3, \dots, \mathrm{s}_\mathrm{n}\},\ \mathrm{s}_\mathrm{y} \in \mathrm{S},\ \mathrm{y} = 1, 2, \dots, \mathrm{n} \quad (3.4)$$

Powyżej zdefiniowane zbiory mogą być różnoliczne, tzn. $k \neq l \neq m \neq n$ i dotyczą konkretnego systemu.

Relacje pomiędzy elementami zbiorów *U*, *R*, *P* i *S* można przedstawić za pomocą 4-warstwowego grafu skierowanego *G(Vert, E)*, gdzie wierzchołkami (*Vert*) są elementy kolejno wymienionych wyżej zbiorów, natomiast krawędzie reprezentują relacje skierowane pomiędzy wierzchołkami dwóch sąsiednich warstw. Rysunek 10 prezentuje fragment przykładowego grafu warstwowego dla modelu RBAC. Przyjmując taki model kontroli bezpieczeństwa weryfikacja czy użytkownik u_x może wykonać operację s_y sprowadza się do sprawdzenia, czy istnieje ścieżka łącząca te dwa wierzchołki w grafie *G(Vert, E)*. Zbiór wierzchołków *Vert* wynosi:

$$\mathrm{Vert} = \mathrm{U} \cup \mathrm{R} \cup \mathrm{P} \cup \mathrm{S} \quad (3.5)$$

zaś zbiór krawędzi:

$$\mathrm{E} = \mathrm{UR} \cup \mathrm{RP} \cup \mathrm{PS} \quad (3.6)$$

gdzie:

$$\mathrm{UR} \subseteq \mathrm{U} \times \mathrm{R} \quad (3.7)$$

$$\mathrm{RP} \subseteq \mathrm{R} \times \mathrm{P} \quad (3.8)$$

$$\mathrm{PS} \subseteq \mathrm{P} \times \mathrm{S} \quad (3.9)$$

Niech $G_z(Vert_z, E_z)$ oznacza podgraf grafu *G*, tzn. $G_z \subset G$ dotyczy warstwy *z*, $z = 1, 2, 3$ wtedy $Vert_z \subset Vert$, $E_z \subset E$. Więc:

$$\mathrm{Vert}_1 \subseteq \mathrm{U} \cup \mathrm{R},\ \mathrm{E}_{\mathrm{UR}} = \mathrm{E}_1 \subseteq \mathrm{U} \times \mathrm{R} \quad (3.10)$$

$$\mathrm{Vert}_2 \subseteq \mathrm{R} \cup \mathrm{P},\ \mathrm{E}_{\mathrm{RP}} = \mathrm{E}_2 \subseteq \mathrm{R} \times \mathrm{P} \quad (3.11)$$

$$\mathrm{Vert}_3 \subseteq \mathrm{P} \cup \mathrm{S},\ \mathrm{E}_{\mathrm{PS}} = \mathrm{E}_3 \subseteq \mathrm{P} \times \mathrm{S} \quad (3.12)$$

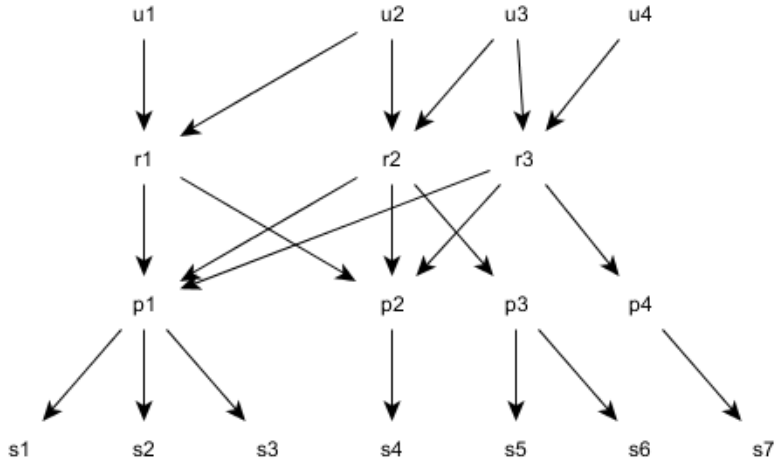
$$\mathrm{E} = \mathrm{E}_1 \cup \mathrm{E}_2 \cup \mathrm{E}_3 \quad (3.13)$$

Tak więc grafy G_z są grafami dwudzielnymi $G_z(Vert_z, Vert_{(z+1) \bmod 3}, E_z)$, *z = 1, 2, 3*.

Macierze reprezentacji tego grafu można przedstawić zgodnie z przyjętym oznaczeniem w postaci $M_z = [m_{xyz}]$, gdzie:

$$\mathrm{m}_{\mathrm{xyz}} = \begin{cases} 1, & \mathrm{xy} \in \mathrm{E}_\mathrm{z} \\ 0, & \text{w pozostałych przypadkach} \end{cases} \quad (3.14)$$

Rysunek 10 przedstawia czterech użytkowników systemu Moja PG. Użytkownik u_1 jest studentem i ma przypisaną rolę r_1 (student). Użytkownik u_2 jest przykładem doktoranta, który występuje w 2 rolach jednocześnie: studenta i nauczyciela, ponieważ równocześnie uczęszcza na zajęcia oraz je prowadzi. Pozostali dwaj użytkownicy (u_3 i u_4) to pracownicy z przypisaną rolą r_3 (pracownik administracyjny). Dodatkowo użytkownik u_3 jest nauczycielem akademickim i ma przypisaną rolę r_2 (nauczyciel), więc jest najprawdopodobniej osobą funkcyjną, np. dziekanem. Roli tej nie ma użytkownik u_4 będący jedynie pracownikiem administracyjnym (np. dziekanatu). Wszystkie z przytoczonych ról umożliwiają podstawowe operacje na koncie użytkownika takie jak: historia swoich logowań do systemu, zmiana hasła, zmiana adresu do powiadomień (odpowiednio s_1, s_2 i s_3). Jest to możliwe, ponieważ posiadają przypisane uprawnienie p_1 (operacje na koncie). Operacja s_4 (pobranie oceny) jest chroniona osobnym uprawnieniem p_2 (odczyt ocen), które w przytaczanym przykładzie również jest przypisane do wszystkich trzech ról, ponieważ każda rola może te oceny przeglądać w różnym celu. Operacje s_5 (dodawanie oceny) oraz s_6 (zmiana oceny) chronione są osobnym uprawnieniem p_3 (edycja ocen), które jest powiązane jedynie z rolą r_2 (nauczyciel). Ostatnia operacja dla rozważanego przykładu s_7 (archiwizacja ocen) strzeżona jest przez dedykowane jej uprawnienie p_4 (archiwizacja ocen) przypisane jedynie do roli r_3 (pracownika administracyjnego).



Rysunek 10 Przykład fragmentu warstwowego grafu G(Vert, E) wykorzystanego do opisu bezpieczeństwa danego systemu internetowego

Tabela 7 przedstawia macierz M_{UR} dla rozważanego przykładu.

Tabela 7 Macierz M_{UR} (pary użytkownik-rola) dla rozważanego przykładu

Rola \ Użytkownik	u_1	u_2	u_3	u_4
r_1	1	1	0	0
r_2	0	1	1	0
r_3	0	0	1	1

Tabela 8 przedstawia macierz M_{RP} dla rozważanego przykładu.

Tabela 8 Macierz M_{RP} (pary rola-uprawnienie) dla rozważanego przykładu

Uprawnienie \ Rola	r_1	r_2	r_3
p_1	1	1	1
p_2	1	1	1
p_3	0	1	0
p_4	0	0	1

Tabela 9 przedstawia macierz M_{PS} dla rozważanego przykładu.

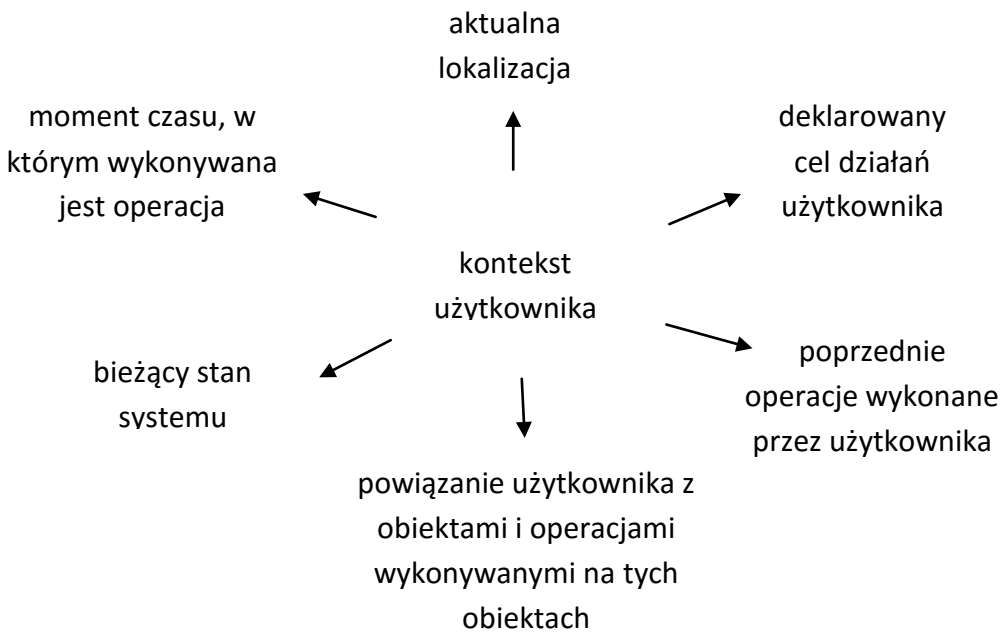
Tabela 9 Macierz M_{PS} (pary uprawnienie-operacja) dla rozważanego przykładu

Operacja \ Uprawnienie	p_1	p_2	p_3	p_4
s_1	1	0	0	0
s_2	1	0	0	0
s_3	1	0	0	0
s_4	0	1	0	0
s_5	0	0	1	0
s_6	0	0	1	0
s_7	0	0	0	1

3.2 Pojęcie kontekstu działań użytkownika

Wraz z postępującą informatyzacją społeczeństwa i rosnącym proporcjonalnie zagrożeniem dla bezpieczeństwa systemów standardowe modele kontroli uprawnień powinny ewaluować do wersji bardziej dynamicznych, dzięki czemu użytkownik, poza krótkim przedziałem czasu, nie ma pełnego kompletu uprawnień, szczególnie tych najbardziej krytycznych [100]. Niezwykle istotny jest tu aspekt bezpiecznego uwierzytelniania cyfrowych tożsamości, w oparciu o które użytkownicy uzyskują pozwolenie na wykonanie akcji w systemach [37]. Dalszym rozwinięciem jest kontekstowa kontrola dostępu do zasobów i operacji na nich, która wiąże poziom uprawnień z bieżącą sytuacją w środowisku pracy użytkowników [115][116][117].

Sliman, Biennier i Badr wskazują na 5 elementów opisujących kontekst: kto (tożsamość – *ang. Identity*), co (czynność), gdzie (lokalizacja), kiedy (czas) i dlaczego (cel) [75]. Innymi słowy ważne jest kto jest użytkownikiem (aktorem) wykonującym operacje (czynności) w systemie oraz jakie posiada on cechy predysponujące go do wykonywania pożądanych czynności. Kolejnymi elementami są fizyczna i logiczna lokalizacja użytkownika oraz określenie czasu, w którym wykonywana jest ta czynność. Ostatni jest cel (intencje) takiego działania użytkownika.



Rysunek 11 Zakres kontekstu działań użytkownika w aspekcie bezpieczeństwa

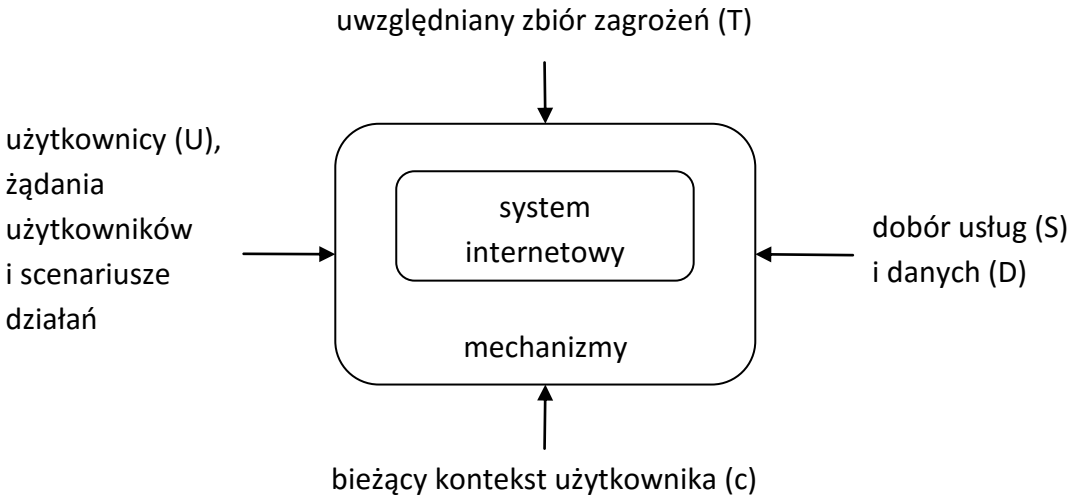
Przyjmijmy więc, że kontekst jest to zbiór wyszczególnionych reguł w systemie warunkujących działanie użytkownika. Rysunek 11 ilustruje rozpatrywane typy kontekstu w aspekcie bezpieczeństwa systemu. W pierwszej kolejności jest to moment czasu, w którym wykonywana jest przez użytkownika dana operacja. Następnie istotna jest lokalizacja fizyczna i/lub logiczna użytkownika. Wiele prac uwzględnia tego typu kontekst [35][118][119][120][121]. Kolejnym aspektem jest deklarowany przez użytkownika cel jego działań. Wiąże się to przeważnie z nadaniem dodatkowych uprawnień w sytuacjach wyjątkowych [122]. Bardzo często wpływ mają wcześniej wykonane operacje w ramach pewnego scenariusza działań [123]. To właśnie scenariusze (mniej lub bardziej złożone) określają zbiór dopuszczalnych operacji użytkownika. Logowanie (zapisywanie w logach) tych operacji użytkownika pozwala dodatkowo zbudować profil jego działań [124], na podstawie którego możliwe będzie wychwytywanie i blokowanie sytuacji wyjątkowych, np. użytkownik, który zawsze loguje się do systemu z Polski, chce nagle zalogować się z Chin lub Rosji. Jest to sytuacja nietypowa, zwłaszcza, że stamtąd pochodzi większość ataków sieciowych. Kolejnym typem kontekstu, który wykorzystywany jest często w dużych systemach o charakterze ewidencyjnym, jest powiązanie użytkownika z obiektami (danymi) i operacjami, jakie może na nich wykonać. Relacje te określa się niezależnie pomiędzy:

- użytkownikami (U), a operacjami/usługami (S),
- użytkownikami (U), a obiektami (O),
- obiektami/danymi (O), a operacjami/usługami (S).

Określają one dopuszczalny zakres działań konkretnego użytkownika (u) na określonym zbiorze obiektów/danych (O). Generalnie użytkownik nie ma pełni uprawnień dostępu do wszystkich obiektów w systemie - są one w czasie wyznaczane i ograniczane do określonych podzbiorów w zależności od zaistniałego kontekstu (np. lokalizacji). Należy też zwrócić uwagę na delegację uprawnień z jednego użytkownika na drugiego użytkownika, np. przyjęcie innego pracownika na zastępstwo z uwagi na wykluczenie z pracy stałego pracownika z powodu choroby, itp. [125], co znacznie komplikuje model. Poza tym wszystkie te typy kontekstu osadzone są w bieżącym stanie systemu określanym przez tryb pracy, obciążenie serwerów, liczbę aktywnych połączeń, liczbę aktywnych węzłów aplikacyjnych, dostępność różnych źródeł danych, itp.

Poszczególne typy kontekstu są ściśle ze sobą powiązane, co powoduje, że interferują na siebie wzajemnie [115]. Kontekst bieżącego stanu systemu uwzględnia również aktualny globalny tryb pracy: normalny lub sytuacji wyjątkowej. W ten sposób możliwe jest przełączanie pomiędzy kilkoma scenariuszami pracy systemu zdefiniowanymi w polityce bezpieczeństwa. Sama polityka natomiast może uwzględniać również kontekst, np. w przypadku urządzeń mobilnych logowanie do nich w określonych lokalizacjach może być uproszczone lub zakazane [126].

Uwzględnienie tych wszystkich parametrów powiązanych z wykonywaną operacją pomaga zwiększyć wykrywalność i ułatwia odfiltrowanie prób ataku na system, co w efekcie podnosi poziom bezpieczeństwa tego systemu. Jak już wcześniej wspomniano, przekłada się to także na wzrost zaufania do systemu. Oczywiście przy jednoczesnym założeniu, że nie ogranicza się w ten sposób jego użyteczności.



Rysunek 12 Architektura logiczna systemu zorientowanego na usługi z uwzględnieniem kontekstu

Rysunek 12 ilustruje architekturę logiczną systemu zorientowanego na usługi uwzględniającego kontekst. Obrazuje on system operujący na danych (D) i udostępniający usługi (S). Z systemem łączy się użytkownik (U). Działanie systemu i interakcja użytkownika z systemem osadzona jest w kontekście (C). Na system oddziałują również zagrożenia (T), przed którymi system jest chroniony za pomocą mechanizmów bezpieczeństwa. Mechanizmy te mogą (lub nie) uwzględniać bieżący kontekst.

Tabela 10 przedstawia porównanie elementów bezpieczeństwa w systemach bez i z uwzględnieniem kontekstu. Różnica pomiędzy tradycyjnym modelem RBAC oraz proponowanymi modelami kontekstowymi CoRBAC (Context-oriented Role Based Access Control) i TCoRBAC (Trust- and Context-oriented Role Based Access Control) uwidacznia się w możliwościach realizacji dynamicznych polityk bezpieczeństwa, tj. takich, które uwzględniają szerokie tło działań i dostosowują reguły bezpieczeństwa do zaistniałej sytuacji.

Tabela 10 Różnice w bezpieczeństwie systemów bez i z kontekstem

	Bez kontekstu	Z kontekstem
Polityka bezpieczeństwa	Polityka statyczna Mechanizmy zabezpieczeń działają według stałych, z góry określonych reguł.	Polityka bardziej dynamiczna Mechanizmy zabezpieczeń zmieniają się w trakcie działania systemu w zależności od bieżącego kontekstu.
Model bezpieczeństwa	RBAC	CoRBAC / TCoRBAC
Mechanizmy bezpieczeństwa	Mechanizmy bezkontekstowe	Mechanizmy kontekstowe

W kolejnych sekcjach rozdziału przedstawiono możliwe sposoby uwzględnienia kontekstu w systemie oraz politykę bezpieczeństwa na poszczególnych poziomach jego architektury. Ostatecznie zaproponowano formalny opis zorientowanego kontekstowo modelu bezpieczeństwa będącego rozszerzeniem modelu RBAC.

Kontekst systemu obejmuje zarówno kontekst, w którym działa użytkownik, jak i kontekst związany z funkcjonowaniem systemu. Dotyczy to zwłaszcza takich newralgicznych warstw jak warstwa logiki biznesowej czy warstwa danych.

Rozpatrzmy kontekst użytkownika dotyczący poszczególnych warstw systemu o przyjętej architekturze opisanej w rozdziale 2.3.1. Kontekst ten ma wpływ na bezpieczeństwo w sferze kontroli dostępu (*ang. security*) jak i zapewnienia ciągłości działania (*ang. safety*). W dalszych rozważaniach ograniczono się tylko do problemów bezpieczeństwa. Poniżej przedstawiono ogólny opis przykładowych kontekstów występujących w poszczególnych warstwach internetowego systemu usługowego (ISS). Szczegółowe przykłady kontekstów oraz zaimplementowanych w systemie Moja PG mechanizmów bezpieczeństwa zostały przedstawione w rozdziale 4.

Kontekst w warstwie zarządzania i bezpieczeństwa (sfera kontroli dostępu) dotyczy:

- Fizycznej lokalizacji użytkownika/usługi – np. kraj, miasto, kampus uczelniany. Lokalizacja może być wyznaczana z wykorzystaniem pozycjonowania GPS, w oparciu o dostępne sieci bezprzewodowe [127], proste skanowanie kodów QR wskazujących lokalizację (np. tabliczki na drzwiach do sal wykładowych/pokoi prowadzących) czy bardziej zaawansowane techniki typu rzeczywistość rozszerzona (ang. *Augmented Reality*) [128][129]. Kontekst ten jest bardzo często wykorzystywany w różnych systemach, np. nawigacji, wskazywania najbliższych POI (*ang. point of interest*) [130]. Pozwala również na ograniczenie geograficzne zasięgu pewnych usług. Przykładem zastosowania lokalizacji wyznaczanej na podstawie geolokalizacji adresu IP użytkownika [131] jest oferowanie wybranych materiałów video tylko na obszarze kraju, co wynika z ograniczeń prawnych.
- Logicznej lokalizacji użytkownika/usługi – wyznaczanej najczęściej poprzez sieć, za pomocą której komunikuje się użytkownik. Mogą być to z góry określone sieci takie jak np. instytucjonalna sieć komputerowa, Internet, wydzielona sieć wewnętrzna. Szczególnie ten ostatni typ sieci wykorzystuje się w przypadku krytycznych systemów. Istotne jest, że logiczna lokalizacja może zupełnie nie korespondować z lokalizacją fizyczną. Wiąże się to z możliwością wykorzystania połączeń VPN [5]. Warunkiem koniecznym jest brak fizycznego odseparowania sieci od innych (w tym Internetu). Warto zwrócić uwagę, że już samo tworzenie wydzielanych sieci jest mechanizmem podnoszenia bezpieczeństwa poprzez kontrolę dostępu – aby uzyskać dostęp do systemu, należy w pierwszej kolejności uzyskać dostęp do odpowiedniej sieci, z której system ten jest osiągalny [32].
- Urządzenia, za pomocą którego komunikuje się użytkownik – dotyczy to zarówno sprzętu, jak i platformy systemowej. W chwili obecnej na równi z komputerami klasy PC popularne są urządzenia przenośne, takie jak różnego rodzaju laptopy, netbooki, notebooki, tablety (w tym popularne iPady), czytniki e-booków, smartphone'y, feature-phone'y i inne. Na wszystkich tych urządzeniach uruchamiane są różne systemy operacyjne i aplikacje [132]. Wiążą się one również z różnymi poziomami bezpieczeństwa [124]. Można więc ograniczać pewną funkcjonalność do konkretnego podzakresu obsługiwanych urządzeń i wspieranych wersji oprogramowania natywnego. Dobrym przykładem są aplikacje mobilne bankowości internetowych, które oferują zdecydowanie węższą funkcjonalność niż ich odpowiedniki na komputerowe przeglądarki WWW. Przeważnie już sam interfejs jest zbudowany zupełnie inaczej. Wynika to prawdopodobnie z faktu, że urządzenie to łatwiej ukraść. Dodatkowo w jednym urządzeniu ponownie skupiają się dwa kanały komunikacji, które z założenia miały być niezależne (dostęp do systemu i smsy potwierdzające operacje).
- Czasu – rozpatrywanego w różny sposób. Pierwszym, najpopularniejszym podejściem jest uwzględnienie momentu wykonania operacji, np. akcje możliwe do wykonania

tylko w godzinach pracy, w dni robocze, itp. Kolejnym jest kolejność zdarzeń następujących po sobie w czasie z uwzględnieniem odstępu czasu pomiędzy poszczególnymi operacjami. Badanie tego wskaźnika wykorzystuje się podczas obrony przed atakami DoS (*ang. Denial of Service*). Inne mechanizmy uwzględniające dopuszczalne przedziały czasu to okresowe wymuszanie zmiany hasła i deaktywacja niewykorzystywanych przez dłuższy czas kont.

- Scenariusza działań, w ramach którego wykonywana jest operacja [133]. W tym przypadku analizowana jest historia poprzednich operacji wykonanych w systemie. Tego typu zabezpieczenia istotne są w systemach przepływu pracy (*ang. workflow*), gdzie kontrolowana jest właściwa kolejność wykonywanych działań. Tego typu mechanizmy mogą analizować np. poprzednią lokalizację użytkownika, z której użytkownik logował się do systemu, porównując z bieżącą lokalizacją w celu detekcji nietypowej zmiany jego położenia.

Kontekst w warstwie logiki biznesowej (sfera zapewnienia ciągłości działania) dotyczy:

- Trybu pracy systemu – możliwe jest stosowanie różnych wariantów polityki dostępu zależnie od sytuacji czy jest to praca normalna, serwisowa czy awaria, itp.
- Obciążenia systemu – mierzonego przez liczbę aktywnych użytkowników, liczbę aktywnych połączeń, wielkość wykorzystanych zasobów sprzętowych dostępnych serwerów, itp. Często obciążenie systemu będzie się różnić podczas normalnej pracy i przy zwiększonym zainteresowaniu użytkowników, albo będzie zupełnie inne podczas ataku DoS.
- Stanu rozproszenia systemu – poprzez określenie czy wszystkie kluczowe węzły klastra są dostępne i pracują normalnie.

Kontekst w warstwie dostępu do danych (sfera kontroli dostępu) dotyczy:

- Lokalizacji usługi przy dostępie do źródła danych. Interoperacyjne usługi komunikują się z innymi usługami. Mogą również wykorzystać kilka różnego typu źródeł danych (*ang. datasource*). W tym momencie można mówić o wzajemnej lokalizacji usług i źródeł danych, na których pracują. Z punktu widzenia źródła danych możemy mówić o usługach lokalnych (zwykle uprzywilejowanych) oraz usługach zewnętrznych, czy inaczej zdalnych. Przeważnie tylko usługi lokalne są uprawnione do modyfikacji danych, natomiast zewnętrzne mają uprawnienia tylko do odczytu.
- Powiązania danych z usługą. Prawidłowo zaprojektowana architektura, w której do jednego źródła danych odwołuje się wiele usług powinna przewidywać gradację dostępu do danych. Tylko wyróżnione uprzywilejowane usługi (najczęściej jedna) mają możliwość modyfikowania danych zarządzanych przez tą usługę. Pozostałe usługi mają dostęp do tych danych w trybie „tylko-do-odczytu". Takie podejście podyktowane jest dbałością o integralność danych. W ramach usług uprzywilejowanych implementowane są spójne algorytmy weryfikacji poprawności danych i zapisu ich do źródłowej bazy danych. Jest to istotne szczególnie w trakcie

życia systemu, gdy usługi ewaluują i trudno byłoby zapewnić spójność reguł walidacyjnych wdrażanych w różnym czasie, dla różnych usług operujących na tym samym zakresie danych.

Kontekst w warstwie danych (strefa kontroli dostępu) dotyczy:

- Powiązania danych z użytkownikiem – wiele danych w systemach jest powiązanych semantycznie z użytkownikami (np. preferencje użytkownika, dane osobowe użytkownika). W takim przypadku mówi się o danych powiązanych wprost z użytkownikiem. Jednakże istnieją również przypadki bardziej rozbudowane. Przykładem może być system szpitalny, w którym każdy lekarz może uzyskać dostęp do kart historii chorób tylko pacjentów będących pod jego opieką [122]. Relacje te mogą być bardzo skomplikowane. Mogą mieć one charakter stały lub zmienny w czasie.
- Metadanych opisujących dokument cyfrowy – ściśle powiązanych z polityką kontroli dostępu MAC (Mandatowy Access Control). Różne cechy opisujące bardzo szeroko rozumiany dokument cyfrowy (np. typ dokumentu, stan dokumentu, poziom tajności dokumentu, itp.) pozwalają na zawężanie dostępu do niego. Jest to szczególnie widoczne w systemach elektronicznego obiegu dokumentów i systemach związanych z przepływem pracy (*ang. workflow*) – zależnie od stanu dokumentu poszczególni użytkownicy o różnych rolach mogą na nim wykonać różne operacje.
- Roli/cechy użytkownika – czyli uwzględnienia kim jest użytkownik i jaką pełni rolę. To podejście jest podstawą modelu kontroli dostępu RBAC w warstwie logiki biznesowej, gdzie uprawnienia są nadawane na podstawie cech (ról) użytkownika. Również w warstwie danych można wyznaczyć analogiczne uprawnienia (np. zwykły użytkownik, użytkownik uprzywilejowany, administrator, itp.). Warto podkreślić, że kontekst ten nie odnosi się jednak tylko do ról użytkownika. Możemy analizować dowolną cechę użytkownika, np. płeć, narodowość, czy język jakim się on posługuje.
- Częstości zmian konkretnych danych – różne dane są zmieniane z różną częstotliwością. Analizując historię zmian danych można wychwycić nietypowe „ożywienie" w aktualizacji danych, co może oznaczać przełamanie zabezpieczeń i nieautoryzowaną modyfikację danych.

Tak szeroko rozumiany kontekst wpływa na definicję polityki bezpieczeństwa, jak i budowę mechanizmów zabezpieczających. W konsekwencji determinuje przyjęty model bezpieczeństwa.

Model bezpieczeństwa jest matematycznym zapisem reguł zawartych w polityce bezpieczeństwa. Formalizuje więc uwzględnienie kontekstu w regułach polityki. Powstało wiele opracowań dotyczących modelowania kontekstowych polityk bezpieczeństwa. Mouatidis i Jurjens proponują integrację ukierunkowanego na cel procesu analizy wymagań

bezpieczeństwa GDSRE (nazywanym Secure Tropos) z podejściem opartym na modelu MBSE (nazywanym UMLsec) [134]. Wskazują oni również na dwuaspektowość bezpieczeństwa – czynniki techniczne i bardzo często zaniedbywany wpływ czynników ludzkich [135][136].

W innym aspekcie ujmują kontekst Haibo i Fan proponując model CGRBAC [19]. Skupiają się oni na globalnych rolach, tak aby zapewnić spójne zarządzanie rolami w heterogenicznych systemach. Rozwiązanie to obejmuje jednak również dynamiczne zmiany kontekstu. Sama notacja dotycząca kontekstu, w szczególności w modelowaniu obsługi procesów (*ang. event processing modeling*), została omówiona w pracy O. Etziona i in. [137].

W niniejszej rozprawie zaproponowano model kontekstu jako zbiór parametrów kontekstu o dyskretnych wartościach (patrz punkt 3.3). Podejście takie pozwala zamodelować praktycznie każdy rodzaj kontekstu. Dodatkowo model rozszerzono o zaufanie do użytkownika (patrz punkt 3.4). W rozdziale 4 zaprezentowano przykładowe mechanizmy wykorzystujące różne parametry kontekstu.

3.3 Opis modelu CoRBAC

Istnieją dwie metody uwzględnienia kontekstu w warstwie kontroli dostępu. Pierwszym z nich jest dynamiczna zmiana uprawnień w zależności o czynników kontekstu (*ang. dynamic role activation*) [138]. Drugim natomiast są uprawniania warunkowe, czyli nadawane tylko wtedy, gdy zaistnieją postawione warunki kontekstu [17][139]. Co więcej możemy wyróżnić warunki statyczne, weryfikowane każdorazowo w momencie przypisywania użytkownikowi konkretnej roli, bądź warunki dynamiczne wyliczane na bieżąco podczas bieżącej interakcji użytkownika z systemem.

Istnieje kilka możliwości formalnego opisu kontekstu w modelu bezpieczeństwa. Przykładowo w tym celu można wykorzystać tzw. funkcje [140] albo warunki kontekstu [141]. W dalszej części pracy zaprezentowano jednak rozwiązanie oparte o graf skierowany oraz macierze wzajemnych relacji poszczególnych elementów, z uwzględnieniem kontekstu jako „filtra uprawnień”. To podejście będzie podstawą dalszych rozważań w rozprawie.

Kontekst (*C*) to zbiór parametrów kontekstu (*CP*):

$$\mathrm{C} = \mathrm{CP}_1 \times \mathrm{CP}_2 \times \mathrm{CP}_3 \times \ldots \times \mathrm{CP}_\mathrm{w} \quad (3.15)$$

Parametr kontekstu (*CP*) to skończony dyskretny zbiór możliwych wartości danego parametru:

$$\mathrm{CP_i} = \left\{\mathrm{cp_{i1}}, \mathrm{cp_{i2}}, \mathrm{cp_{i3}}, \ldots, \mathrm{cp_{iv_i}}\right\}, \quad \mathrm{i} = 1, 2, \ldots, \mathrm{w} \quad (3.16)$$

$$\mathrm{v_i} = |\mathrm{CP_i}|, \quad \mathrm{i} = 1, 2, \ldots, \mathrm{w} \quad (3.17)$$

Przyjęte założenia:

- Zbiór parametru kontekstu CP_i pokrywa wszystkie możliwe wartości tego parametru, co oznacza, że dziedzina parametru została skwantyfikowana w całości.
- Zbiory parametrów kontekstu mogą być różnoliczne.

$$|\mathrm{CP_a}| = \mathrm{v_a}, |\mathrm{CP_b}| = \mathrm{v_b},\ \mathrm{v_a} = \mathrm{v_b} \vee \mathrm{v_a} \neq \mathrm{v_b} \quad (3.18)$$

- Wartości danego parametru kontekstu (elementy zbioru CP_i) są rozłączne, co oznacza, że w danej, określonej sytuacji (wykonywania konkretnej operacji ($s \in S$) przez użytkownika ($u \in U$)) tylko jeden element zbioru opisuje dany parametr kontekstu.

$$\forall \mathrm{cp_{ja}} \in \mathrm{CP_j} \ \wedge \ \forall \mathrm{cp_{jb}} \in \mathrm{CP_j} \implies \mathrm{cp_{ja}} \neq \mathrm{cp_{jb}} \quad (3.19)$$

- W danej sytuacji wszystkie parametry kontekstu są określone przez dopuszczalne wartości, co oznacza, że dla każdego CP_i wskazywany jest element zbioru wartości opisujący dany parametr kontekstu.

Tak zdefiniowany model kontekstu wraz z powyższymi założeniami jest bardzo uniwersalny – pozwala na zamodelowanie praktycznie dowolnego rodzaju kontekstu, niezależnie od jego charakteru. Cechuje go duża elastyczność – uwzględnia proste konteksty składające się z kilku parama terów kontekstu, jak i te bardzo złożone. Dodatkowo jest on łatwy w implementacji w systemie.

Niektóre parametry kontekstu mogą mieć charakter ciągły (np. czas), więc na potrzeby analizy kontekstu zostają one skwantyfikowane do pewnych przedziałów, np. dni tygodnia. Analogiczna sytuacja dotyczy parametrów kontekstu o wartościach dyskretnych, np. adres IP użytkownika. Ponieważ zbiór możliwych wartości adresów IP jest bardzo liczny i bezcelowe jest analizowanie tak licznego zbioru, zakłada się podzielenie pełnego zbioru na pewne zakresy (grupowanie) według z góry określonych reguł, np. wydzielona sieć kampusowa o masce 19-bitowej i pozostałe adresy (Internet). **W dalszej części rozprawy każdorazowo wartość parametru kontekstu jest rozumiana jako taki zakres (grupa) wartości, chyba że zaznaczono inaczej.**

Rozpatrzmy przykładowy zbiór parametrów kontekstu (*CP*):

CP_1 – zbiór możliwych lokalizacji użytkownika o elementach: cp_{11} = wydzielona sieć wewnętrzna, cp_{12} = sieć uczelniana, cp_{13} = Internet.

CP_2 – zbiór możliwych dni tygodnia o elementach: cp_{21} = pon. - pt. , cp_{22} = sob., cp_{23} = niedz.

Przyjęto, że kontekst to element zbioru C określonego w następujący sposób:

$$C = \{c_1, c_2, c_{3,} \dots, c_z\}, \quad c \in C \quad (3.20)$$

Implikuje to stwierdzenie, że dla powyższego przykładu kontekst (*C*) to zbiór dwójek:

$$C = \begin{Bmatrix} (cp_{11}, cp_{21}), (cp_{12}, cp_{21}), (cp_{13}, cp_{21}), \\ (cp_{11}, cp_{22}), (cp_{12}, cp_{22}), (cp_{13}, cp_{22}), \\ (cp_{11}, cp_{23}), (cp_{12}, cp_{23}), (cp_{13}, cp_{23}) \end{Bmatrix} \quad (3.21)$$

Moc tego zbioru to iloczyn liczebności wszystkich zbiorów parametrów kontekstu:

$$|C| = \prod_{i=1}^{w} v_i \quad (3.22)$$

W przytaczanym przypadku wynosi on 3 * 3 = 9. Oznacza to, że możliwych jest 9 wartości kontekstu. Wśród tych wszystkich możliwych wartości nie wszystkie muszą przejawiać się z tą samą częstotliwością w systemie.

Kontekst zidentyfikowany w rozdziale 3.2 może dotyczyć zarówno użytkownika (U), operacji/usługi (S), roli użytkownika (R), relacji z innymi obiektami (O), jak i uprawnień (P). Analiza jego wpływu na bezpieczeństwo prowadzi jednak do wniosku, że uwzględnienie kontekstu sprowadza się do wystąpienia (lub nie) uprawnienia (P) do wykonania operacji/usługi (S). Wszystkie mechanizmy uwzględniające kontekst będą zabezpieczały przed zagrożeniami (T) w sferze kontroli dostępu (*ang. security*) poprzez blokadę dostępu. Analogicznie mechanizmy kontekstowe operujące w sferze zapewnienia ciągłości działania (*ang. safety*) będą reagowały przez blokowanie dostępu zagrażającego bezpieczeństwu i stabilności systemu (np. atak DDoS).

Można więc przyjąć, że uwzględnienie kontekstu c prowadzi do rozbudowy modelu RBAC poprzez modyfikację (zawężanie) zbioru uprawnień *P* stosując funkcję f_C. Formalnie można to zapisać następująco:

$$f_C: P \longrightarrow P(c), \quad P(c) \subseteq P, \quad c \in C \quad (3.23)$$

P(c) to zmodyfikowany zbiór uprawnień uwzględniający kontekst.

Funkcję tą można przedstawić zgodnie z przyjętym oznaczeniem w postaci macierzy $M_{PC} = [m_{pc}]$, gdzie:

$$\text{dla } \forall c \in C, \forall p \in P \qquad m_{pc} = \begin{cases} 1, & c \in C \ \wedge p \in P(c) \\ 0, & \text{w pozostałych przypadkach} \end{cases} \quad (3.24)$$

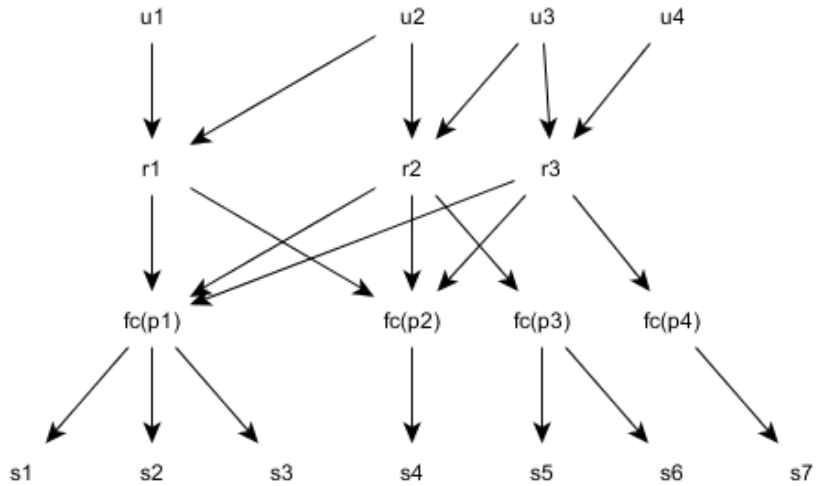
Tabela 11 ilustruje przykładowy wpływ kontekstu na uprawnienia w postaci macierzy M_{PC}.

Tabela 11 Przykładowa macierz M_{PC} (uprawnienia zależne od kontekstu)

Uprawnienie \ Kontekst	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9
p_1	1	1	1	1	1	1	1	1	1
p_2	1	1	1	1	1	1	1	1	1
p_3	1	1	0	1	1	0	1	1	0
p_4	1	0	0	0	0	0	0	0	0

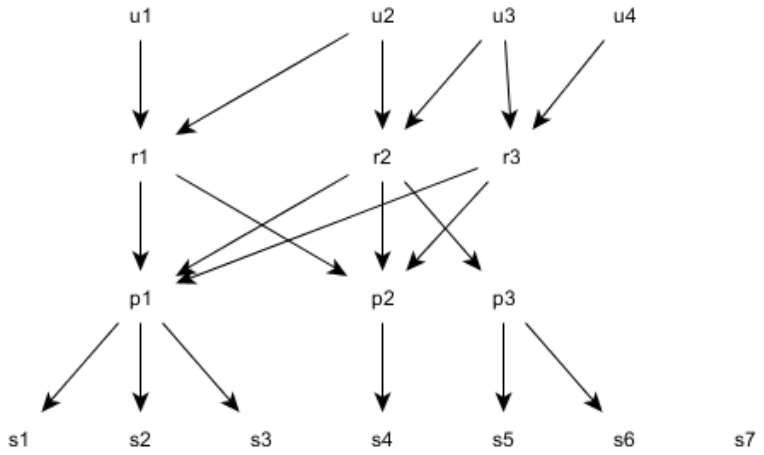
Warto zauważyć, że przyjęta definicja kontekstu (w każdej sytuacji wszystkie parametry kontekstu przyjmują wartości) oraz kwantyzacja parametrów kontekstu (wartości parametru kontekstu są rozłączne i pokrywają wszystkie możliwe wartości) powoduje, że **w danej analizowanej sytuacji kontekstu dokładnie tylko jedna kolumna macierzy M_{PC} będzie miała zastosowanie**. Kolumna ta odpowiada wartości kontekstu dla danej sytuacji wykonywania konkretnej operacji ($s \in S$) przez użytkownika ($u \in U$).

Rysunek 13 prezentuje graf *G'(Vert(c), E(c))* będący modyfikacją grafu *G (Vert, E)* (patrz Rysunek 10) za pomocą funkcji f_C, w którym uprawnienia $p_x \in P(c)$ występują zależnie od wartości parametrów kontekstu. W przypadku, gdy dla danej sytuacji kontekstu macierz M_{PC} dla danego uprawnienia p_x przyjmuje wartość zero, wierzchołek ten nie istnieje w grafie, więc nie można poprowadzić przez niego ścieżki łączącej użytkownika u_i z operacją/usługą s_j.

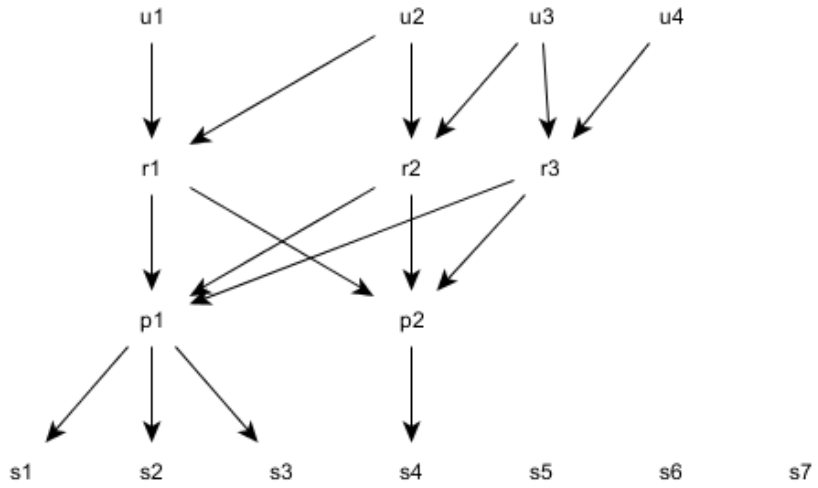


Rysunek 13 Przykład fragmentu warstwowego grafu G'(Vert, E) uwzględniającego kontekst C wykorzystanego do opisu bezpieczeństwa danego systemu

Rysunek 14 przedstawia graf *G'(Vert(c), E(c))* dla wartości kontekstu c_2, c_4, c_5, c_7, c_8 (macierz M_{PC} zgodnie z tabelą 11). Rysunek 15 natomiast prezentuje graf *G'(Vert(c), E(c))* dla wartości kontekstu c_3, c_6, c_9 (macierz M_{PC} zgodnie z tabelą 11).



Rysunek 14 Przykładowy graf G'(Vert(c), E(c)) dla bieżącego kontekstu c_2, c_4, c_5, c_7, c_8 zgodnie z tabelą 11



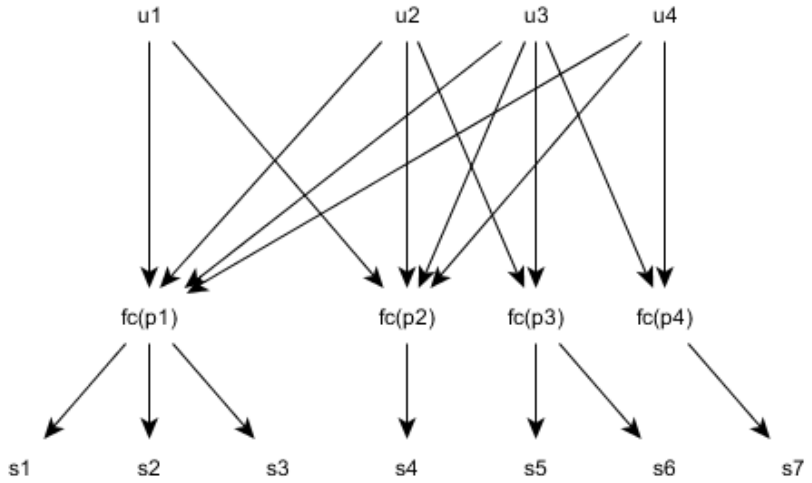
Rysunek 15 Przykładowy graf G'(Vert(c), E(c)) dla bieżącego kontekstu c_3, c_6, c_9 zgodnie z tabelą 11

Wielowarstwowe, rozproszone systemy internetowe wykorzystują mechanizmy bezpieczeństwa, które realizują kontrolę bezpieczeństwa w różnych miejscach architektury systemu, na różnych etapach obsługi żądania. Jest to podyktowane właśnie rozproszeniem (często nawet fizycznym) różnych elementów systemu. Tak więc dostęp może być kontrolowany już na początku obsługi żądania, tj. w warstwie interfejsu użytkownika. Może też być kontrolowany w logice biznesowej przy dostępie do usług (zarówno od strony interfejsu użytkownika, jak i od innych interoperacyjnych usług). Również przy dostępie do danych bardzo często stosowana jest niezależna kontrola dostępu.

Warto zwrócić uwagę, że nie we wszystkich mechanizmach bezpieczeństwa (na różnych warstwach) musi być wykorzystywany pełny model RBAC. Można go uprościć poprzez pominięcie ról (R) wiążąc bezpośrednio użytkownika (U) z uprawnieniami (P). Pominięcie

takie **nie** skutkuje zmniejszeniem liczby ścieżek pomiędzy użytkownikami (U) i uprawnieniami (P). Wpływa jedynie na wygodę administrowania (nadawania bądź odbierania uprawnień zgrupowanych w rolach) – nie wpływa w żaden sposób na kontrolę uprawnień. Podejście takie stosowane jest przeważnie w przypadkach, gdy liczność zbioru ról (R) jest zbliżona do liczności zbioru użytkowników (U) lub zbioru uprawnień (P) – każdy użytkownik posiada swoją rolę lub każda rola odpowiada jednemu uprawnieniu.

Idąc tym tokiem rozumowania otrzymujemy również uogólniony model kontekstowy. Rysunek 16 przedstawia modyfikację przykładowego grafu G' – graf G'' dla uogólnionego modelu kontroli bezpieczeństwa. W grafie tym pominięto role (R).



Rysunek 16 Przykład fragmentu uogólnionego warstwowego grafu G''(Vert(c), E(c)) uwzględniającego kontekst c wykorzystanego do opisu bezpieczeństwa danego systemu

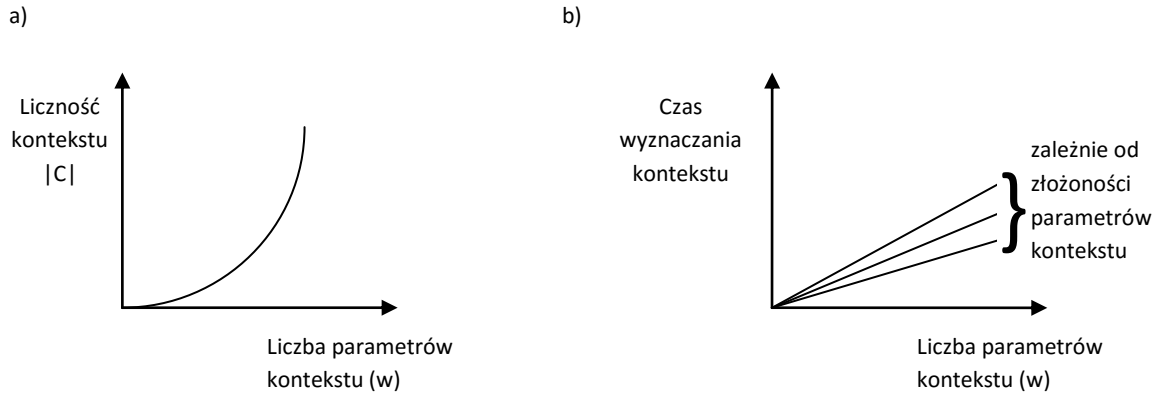
W takim modelu algorytm wyznaczania decyzji o przyznaniu dostępu nie ulega zmianie – konieczne jest istnienie ścieżki w grafie łączącej użytkownika (u) z operacją/usługą (s).

Tak uogólniony model pozwala na modelowanie praktycznie każdego mechanizmu bezpieczeństwa uwzględniającego kontekst. Przykładowe mechanizmy zostały przedstawione w dalszej części pracy w rozdziale 4.5.

Zastosowanie uogólnienia nie wyklucza możliwości korzystania z oryginalnego modelu CoRBAC zawierającego również role wykorzystując wszystkie zalety modelu RBAC. Podejście to sprawdza się jednak przeważnie w mechanizmach, w których występuje mnogość użytkowników (U), uprawnień (P) i usług (S), ponieważ pozwala na wygodne administrowanie uprawnieniami użytkowników/klientów [142].

Przyjęty model kontekstu (C) zakłada kwantyzację wartości parametrów kontekstu (CP) oraz całkowite pokrycie dziedziny każdego parametru. Przyjmując nawet wspomnianą wcześniej kwantyzację wartości parametrów kontekstu do pewnych zakresów, problematyczna staje

się liczność możliwych kombinacji wartości kontekstu. Definiowanie kontekstu jako iloczynu kartezjańskiego wszystkich wartości parametrów (wszystkie kombinacje) implikuje wykładniczy przyrost wartości |C|. Z kolei czas potrzebny na wyznaczenie aktualnej wartości kontekstu odpowiadającej wartości wszystkich parametrów kontekstu przybiera na ogół charakter liniowy. Te dwie zależności ilustruje Rysunek 17 a i b.



Rysunek 17 Wykładniczy wzrost liczności kontekstu (a) oraz liniowy czas wyznaczania bieżącego kontekstu (b)

Rosnący czas wyznaczania aktualnych wartości parametrów kontekstu *c* wpływa negatywnie na wydajność systemu. Problematyczny jest natomiast wykładniczy przyrost możliwych kombinacji wszystkich wartości parametrów kontekstu, a co za tym idzie budowania macierzy M_{PC}.

Czas analizy bieżącego kontekstu jest liniowo zależny od liczby analizowanych parametrów kontekstu oraz złożoności samych parametrów kontekstu (liczby operacji niezbędnych do wykonania w celu wyznaczenia aktualnej wartości danego parametru kontekstu). Można go optymalizować dwutorowo. Po pierwsze osiąga się to zmniejszając liczbę analizowanych parametrów kontekstu, co niestety może prowadzić do obniżenia poziomu bezpieczeństwa. Poza tym warto skupić się na implementacji algorytmów wyznaczających aktualne wartości poszczególnych parametrów kontekstu, co wymaga odpowiedniej rewizji kodu.

Drugie zagadnienie dotyczy redukcji rozmiaru macierzy M_{PC}, co sprowadza się do wypracowania odpowiednich algorytmów zawężania uprawnień z uwzględnieniem bieżącego kontekstu. Operacja ta nie wpływa wprost na czas obsługi pojedynczego żądania – przeważnie wykonywana jest jednorazowo podczas konfiguracji systemu i później ew. aktualizacji tej konfiguracji. Problemem jest jednak uniknięcie błędów przy wyznaczaniu macierzy podczas konfiguracji uprawnień w systemie. Błędy te mogą odzwierciedlać się w obniżeniu poziomu bezpieczeństwa – nieświadomie zostaną wyłączone pewne mechanizmy bezpieczeństwa. Dlatego zaproponowano wyznaczanie wartości tylko dla najbardziej istotnych przypadków (wariantów) kontekstu. Dla wszystkich pozostałych przyjęto algorytm „przybliżania" opisany poniżej.

Celem umożliwienia kwalifikacji takich nieokreślonych wprost w macierzy M_{PC} przypadków przyjęto 2 poziomy przybliżania approx (*ang. approximation*) o wartościach od 1 (najniższy) do 2 (najwyższy): *approx1* i *approx2*. Liczba poziomów przybliżania *approx* została wybrana na podstawie przeprowadzonych analiz na danych zebranych w systemie Moja PG [143] i dopasowana do omawianego przykładu. Dla innych systemów może przyjąć inną wartość, co nie wpływa na logikę dalszych rozważań.

Następnie każdej wartości każdego parametru kontekstu przyporządkowano jeden z poziomów przybliżania *approx*. Tabela 12 prezentuje przykładowe przyporządkowanie poziomów zaufania do wartości parametrów kontekstu CP_1.

Tabela 12 Przykładowe przyporządkowanie poziomów przybliżania (approx) do wartości parametrów kontekstu CP_1

CP_1	cp_{11} = wydzielona sieć wewnętrzna	cp_{12} = sieć uczelniana	cp_{13} = internet
approx(CP_1)	2	1	1

Zależnie od przyjętej polityki bezpieczeństwa poziom przybliżania dla wszystkich parametrów kontekstu (*approx(c)*) może być wyznaczany np. na jeden z poniższych sposobów:

1. Jako wartość maksymalna wszystkich *approx(CP_i)*
2. Jako średnia arytmetyczna/ważona wszystkich *approx(CP_i)*
3. Jako wartość minimalna wszystkich *approx(CP_i)*

W kontekście dalszych rozważań wariant trzeci wydaje się być najwłaściwszym wyborem.

Bazując na powyższych założeniach możemy wprowadzić zmodyfikowaną macierz M_{PCN} rozbudowaną o przybliżanie – patrz Tabela 13 (porównaj z Tabela 11).

Tabela 13 Przykładowa macierz M_{PCN} (uprawnienia zależne od kontekstu rozbudowane o przybliżanie)

	przypadki określone dokładnie (najbardziej istotne)			pozostałe przypadki („przybliżenia”)	
Uprawnienie \ Kontekst	c_3	c_6	c_9	approx1	approx2
p_1	1	1	1	1	1
p_2	1	1	1	1	1
p_3	0	0	0	1	1
p_4	0	0	0	0	1

Algorytm wyznaczania kolumny z macierzy M_{PCN} odpowiadającej bieżącemu kontekstowi w pierwszej kolejności próbuje odnaleźć kolumnę w części przypadków dokładnie określonych. Dopiero jeżeli nie znajdzie w pierwszej części, wyznacza wartość poziomu przybliżania *approx* (zgodnie z metodą określoną w polityce bezpieczeństwa) i odczytuje kolumnę odpowiadającą temu poziomowi.

Proponowane rozwiązanie pozwala na dowolną redukcję rozmiaru (liczby kolumn) macierzy M_{PC} – od maksymalnego odpowiadającego |C| do minimalnego odpowiadającego przyjętej liczbie poziomów bezpieczeństwa. Optymalnym jest wyznaczenie „przypadków dokładnie określonych" dla najważniejszych wartości parametrów kontekstu (np. dni robocze, sieć uczelniana, duże obciążenie systemu). Dla pozostałych zaś przypadków zastosowanie metody „przybliżania".

3.4 Opis modelu TCoRBAC

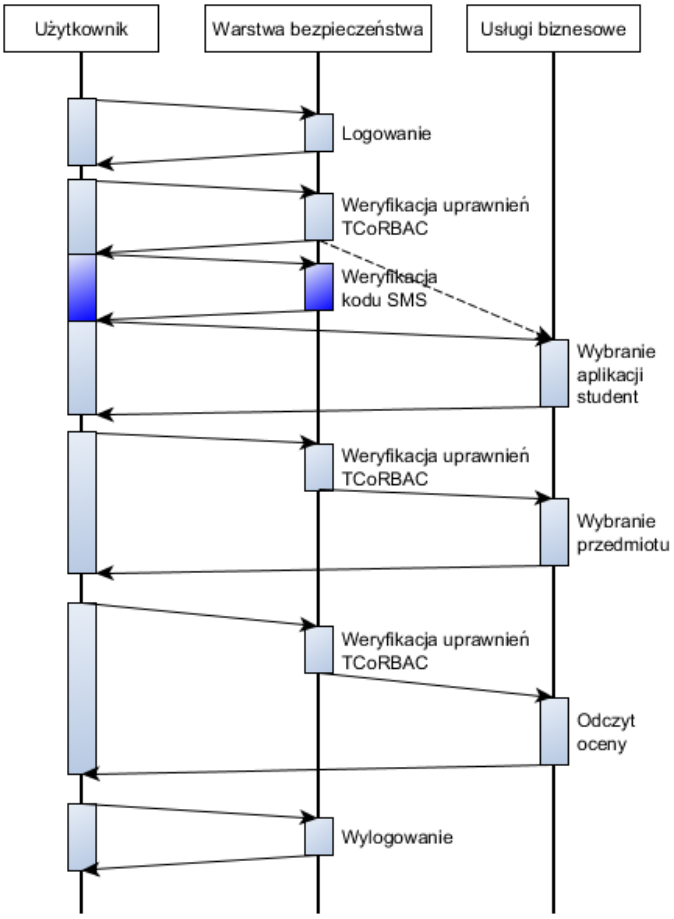
Wprowadzony powyżej model CoRBAC ukierunkowany jest na dynamiczne zawężanie uprawnień w zależności od wartości różnych parametrów kontekstu. Jest on jednak powtarzalny – w takich samych wartościach parametrów kontekstu system zachowa się tak samo. Naturalnym rozwinięciem i uzupełnieniem tego modelu jest model TCoRBAC. Pozwala on na bardziej dynamiczne analizowanie bieżącej interakcji użytkownika z systemem (podczas obsługi poszczególnych żądań użytkownika) i podejmowanie decyzji dodatkowo w oparciu o profil zachowań użytkownika.

W tym momencie niezbędne jest wprowadzenie pojęcia sesji użytkownika jako ciągu interakcji użytkownika z systemem zgodnie z pewnym scenariuszem działań. Jest ono tożsame z sesją użytkownika w systemach internetowych wprowadzoną na potrzeby powiązania ze sobą bezstanowych żądań HTTP(S) w ramach jednego ciągu interakcji użytkownika z systemem od momentu zalogowania do wylogowania lub zaprzestania tej interakcji przez czas dłuższy niż określony w konfiguracji. Rysunek 18 przedstawia przykładowy prosty scenariusz działań w ramach jednej sesji użytkownika. Scenariusz ten w wersji uproszczonej jest analizowany w rozdziale 5 pod kątem użyteczności ε.

Tak więc sesja użytkownika to ciąg pozornie niepowiązanych ze sobą (ze względu na bezstanowość protokołu HTTP(S)) żądań użytkownika. Żądania te są wykonywane w ramach pewnego scenariusza działań użytkownika uis (*ang. user interaction scenario*). W zaproponowanym wcześniej kontekstowym modelu bezpieczeństwa CoRBAC każde żądanie przechodzi przez warstwę bezpieczeństwa, która wyznacza aktualne wartości wszystkich parametrów kontekstu i na podstawie tak wyznaczonego kontekstu przydziela uprawnienia do wykonania operacji/usługi (s) lub odmawia dostępu. Rozwinięciem zaproponowanego bezstanowego modelu kontekstowego jest wprowadzenie modelu stanowego. Polega on na każdorazowym (przy każdym żądaniu) wyznaczaniu poziomu zaufania do użytkownika tl (*ang. trust level*) oraz wplataniu w scenariusz działań użytkownika (uis), związanym z wywoływaniem usług oferujących poszczególne funkcjonalności użytkowe, scenariuszy bezpieczeństwa ss (*ang. security scenario*) – patrz przykład: Rysunek 18.

Poziom zaufania systemu do użytkownika (tl) to miara określająca w jakim stopniu użytkownik jest wiarygodny dla systemu (jest tym, za kogo się podaje). Wiarygodność ta

wyznaczana jest na podstawie informacji na ile bieżący kontekst c pasuje do historii kontekstów poprzednich żądań użytkownika. Zdarzają się sytuacje, gdy użytkownik uwierzytelni się prawidłowo, ale zachowuje się nietypowo dla niego, czyli podejrzanie. W takim przypadku poziom zaufania systemu do użytkownika ulega odpowiedniemu obniżeniu.



Rysunek 18 Przykładowy scenariusz działań użytkownika w ramach jednej sesji z zaznaczonym wymuszonym wywołaniem dodatkowej weryfikacji kodu SMS

Przyjęto, że poziom zaufania przyjmuje wartości od 1 (najniższa) do 4 (najwyższa): *tl1*, *tl2*, *tl3* i *tl4*. Poziom najwyższy *tl4* i oznacza, że użytkownik zachowuje się w sposób dla niego typowy. Każdy wcześniejszy poziom wskazuje na coraz większą nietypowość zachowania użytkownika. Inicjalnie przyjęto poziom najniższy *tl1*, ponieważ nie jest możliwe określenie, czy weryfikowane zachowanie jest typowe dla danego użytkownika, z powodu braku wcześniejszych jego działań w systemie. Liczba poziomów zaufania *tl* została wybrana na podstawie przeprowadzonych analiz na danych zebranych w systemie Moja PG [143], ale dla innych systemów może przyjąć inną wartość, co nie wpływa na logikę dalszych rozważań.

Scenariuszem bezpieczeństwa (ss) może być dowolny dodatkowy ciąg operacji potwierdzających, że użytkownik jest tą osobą, za którą się podaje. Najczęściej będą to proste interakcje wymagające dodatkowego potwierdzenia, np.:

- ss_1 - przepisanie kodu CAPTCHA,
- ss_2 - ponowne wprowadzenie hasła,
- ss_3 - podanie hasła wysłanego innym kanałem (np. SMS),
- ss_4 - wprowadzenie danych z tokena,
- ss_5 - odpowiadanie na kolejne pytania o dane osobowe.

Możliwe są również bardziej skomplikowane scenariusze realizujące mechanizmy typu challenge-response. Przykładem jest ss_5. Polega on na wprowadzaniu przez użytkownika odpowiedzi na poszczególne pytania o pewne fragmenty jego danych osobowych do momentu, aż uzyska on co najmniej 3 poprawne odpowiedzi przy maksymalnie jednej złej. Jest to zabezpieczenie na przypadek pomyłki użytkownika przy wprowadzaniu danych. Każdy scenariusz bezpieczeństwa jest wykonywany tylko raz przy pierwszym żądaniu o danym poziomie zaufania do użytkownika (tl) oraz przy każdorazowej zmianie tego poziomu w ramach sesji użytkownika.

Tabela 14 przedstawia przykładową macierz określającą reguły wplatania poszczególnych scenariuszy bezpieczeństwa (ss) pomiędzy żądania wynikające ze standardowego scenariusza działań użytkownika (uis). Praktyczny przykład wyznaczania aktualnego poziomu zaufania do użytkownika (tl) oraz wykorzystania modelu TCoRBAC przedstawiono w analizie w rozdziale 5.

Tabela 14 Przykładowa macierz wplatania scenariuszy bezpieczeństwa (ss) pomiędzy żądania użytkownika (uis) uwzględniająca poprzedni i bieżący tl

Poprzedni tl \ bieżący tl	tl1	tl2	tl3	tl4
tl1	uis	ss_1	ss_1	ss_1
tl2	ss_4	uis	ss_1	ss_1
tl3	ss_4	ss_3	uis	ss_1
tl4	ss_5	ss_3	ss_2	uis

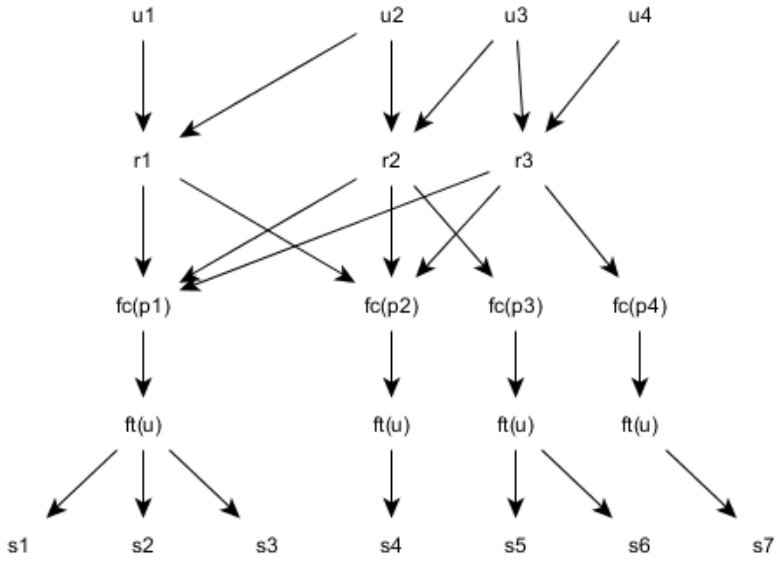
Podążając tym tokiem rozumowania naturalnym jest rozszerzenie rozpatrywanego scenariusza działań użytkownika (uis) o całą historię jego interakcji z systemem (nie tylko ograniczoną do pojedynczej sesji). W ten sposób tworzony jest profil działań użytkownika – sprowadza się do analizy dużych wolumenów danych (*ang. big data*) na potrzeby zapewnienia bezpieczeństwa. Oczywiście profil ten nie jest jedynie logiem wykonywanych operacji, a rejestrem analizowanego kontekstu każdej z tych operacji (wołań usługi).

Rysunek 19 przedstawia przykładowy graf G'''(Vert(c,tl), E(c,tl)) dla modelu TCoRBAC. W porównaniu do modelu CoRBAC (Rysunek 13) zawiera on dodatkową warstwę wierzchołków $f_t(u)$. Jest to funkcja, która odpowiada za wyznaczenie (na podstawie

bieżącego kontekstu c oraz historii kontekstów towarzyszących poprzednim żądaniom użytkownika) bieżącego i poprzedniego poziomu zaufania do użytkownika (tl), wybór właściwego scenariusza bezpieczeństwa (ss) oraz weryfikację, czy został on wykonany prawidłowo – tylko w takim przypadku zwraca ona 1 (w pozostałych zwraca 0). Funkcję tę można zdefiniować następująco:

$$f_t(u) = \begin{cases} 1, & \text{zmiana tl oraz odpowiedni ss wykonany prawidłowo} \\ 0, & \text{w pozostałych przypadkach} \end{cases} \quad (3.25)$$

Analogicznie do modelu CoRBAC, w modelu TCoRBAC musi istnieć ścieżka dla wskazanej pary (u, s) odpowiadającej żądaniu użytkownika u do wykonania usługi s.

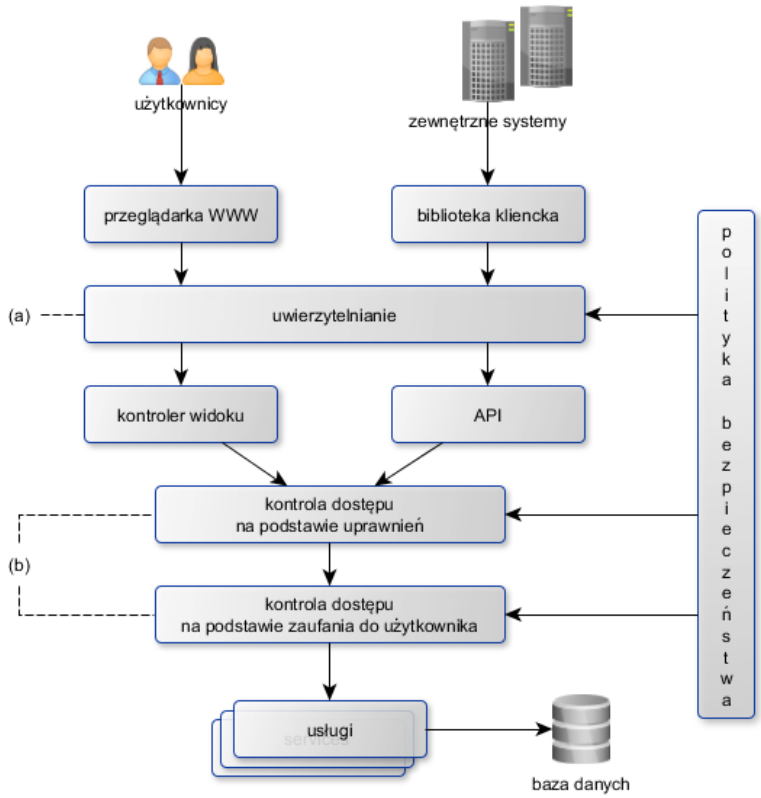


Rysunek 19 Przykładowy graf G'''(Vert(c,tl), E(c,tl)) dla modelu TCoRBAC

Rozwiązanie takie pozwala na zaprojektowanie rozbudowanych mechanizmów bezpieczeństwa, które będą podnosiły bezpieczeństwo systemu, nie wpływając w znacznym stopniu na użyteczność systemu. S. Furnell dowodzi, że najlepszym rozwiązaniem jest minimalna ilość widocznych dla użytkownika zabezpieczeń do czasu, gdy weryfikowany jest on poprawnie przez kolejne mechanizmy zabezpieczeń. Dopiero w momencie, gdy pojawiają się problemy weryfikacyjne i/lub sytuacje wątpliwe, podejrzane, to uwidaczniają się dla niego kolejne mechanizmy bezpieczeństwa [11]. W tym momencie trzeba odwołać się do miary użyteczności systemu ε zdefiniowanej w punkcie 1.4 wzorem 1.1. Z punktu widzenia użytkownika τ_P i τ_C są bliskie zeru, więc najistotniejszy wpływ na ten wskaźnik będzie miał τ_T i jego relacja do τ_B. Należy zatem dążyć do minimalizacji τ_T. Szczegółową analizę w tym zakresie przedstawiono w rozdziale 5.

Takie analizowanie zachowań użytkownika jest w pewnym zakresie analogiczne do mechanizmów IDS (*ang. Intrusion Detection Systems*) oraz IDPS (*ang. Intrusion Detection and Prevention Systems*) [144]. Opierają się one o bieżący monitoring działań użytkownika celem

wychwycenia sytuacji nietypowych. Rozwiązania te mają dwie podgrupy: NIDS (*ang. Network based Intrusion Detection Systems*) oraz HIDS (*ang. Host based Intrusion Detection Systems*). Pierwsze z nich polega na monitoringu przesyłanych danych. Umieszczane jest w strategicznych punktach infrastruktury sieciowej. Cały przekazywany ruch sieciowy jest analizowany i porównywany z bazą znanych sygnatur możliwych ataków. Druga podgrupa działa analogicznie, przy czym skupia się na monitoringu działań systemowych i zmiany konfiguracji systemu operacyjnego. Systemy HIDS umieszczane są w konkretnych systemach operacyjnych. Analizowane jest kto i jakie zmiany próbuje wykonać w konfiguracji systemu operacyjnego. Rozszerzeniem rozwiązań klasy IDS, które tylko wykrywają anomalie i powiadamiają o nich administratora, są rozwiązania IDPS. Zawierają one dodatkowo reguły, które na podstawie wykrytych zagrożeń, blokują je i tym samym chronią system przed intruzem.

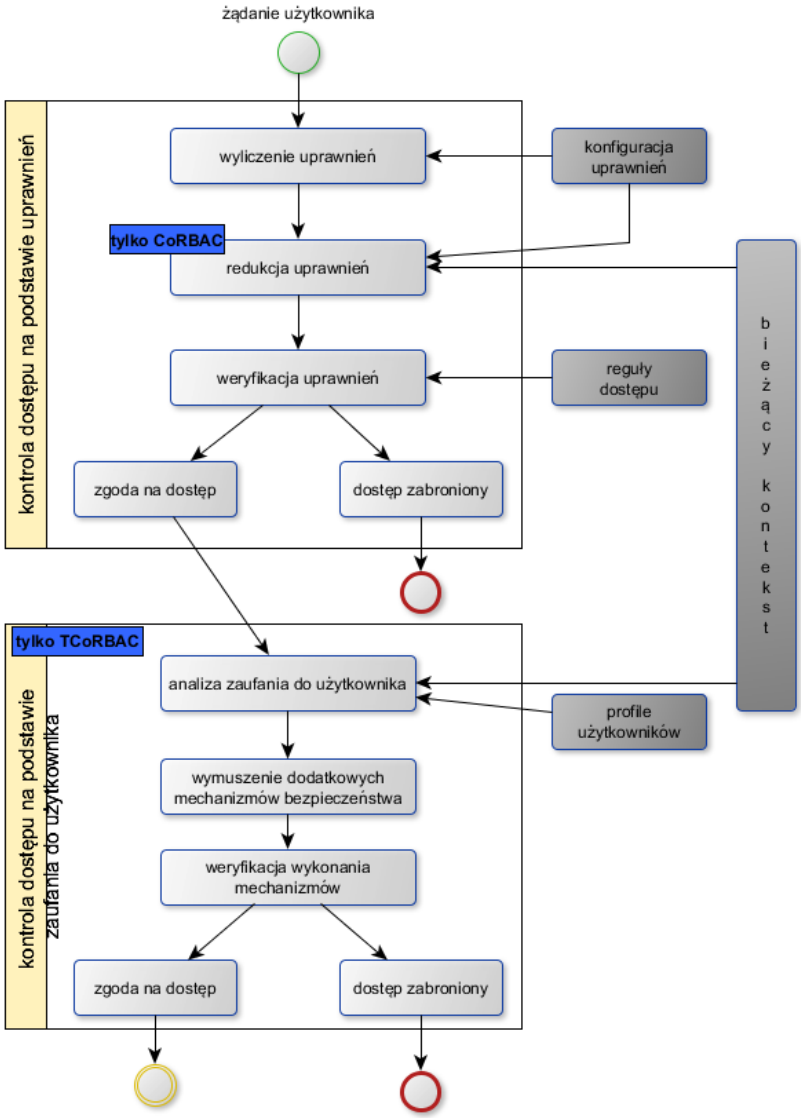


Rysunek 20 Architektura logiczna warstwy bezpieczeństwa systemu internetowego obejmująca: a) uwierzytelnianie, b) dwustopniową kontrolę dostępu

Proponowane rozwiązanie oparte o zaufanie do użytkownika na podstawie profilu jego działań wykracza daleko poza ramy IDPS. Po pierwsze działa na poziomie logiki systemu internetowego o charakterze usługowym. Kolejną różnicą jest odejście od list znanych sygnatur działań niepożądanych (ataków) na rzecz mechanizmu „uczącego się" indywidualnie każdego użytkownika. Jednak najbardziej znaczącą różnicą jest zakres analizowanych danych – przytaczane rozwiązania analizują wąski zakres danych (ruch sieciowy, zdarzenia systemu

operacyjnego), natomiast profil działań użytkownika uwzględnia szeroki kontekst tych działań.

Efektem powyższych rozważań jest wynikowa architektura logiczna warstwy bezpieczeństwa systemu internetowego uwzględniającej bieżący kontekst działania. Rysunek 20 ilustruje właśnie taką architekturę. Obejmuje ona obowiązkowe uwierzytelnianie (a) oraz dwustopniową kontrolę dostępu do usług (b). Oba te elementy są takie same zarówno dla zwykłych użytkowników, jak i zewnętrznych systemów.



Rysunek 21 Poszczególne czynności dwuetapowej kontroli dostępu w modelu CoRBAC

Rysunek 21 przedstawia poszczególne czynności w dwuetapowej kontroli dostępu w modelu RBAC, CoRBAC i TCoRBAC (zaznaczono czynności rozszerzające standardowy model RBAC). Model TCoRBAC składa się z dwóch grup czynności: kontroli dostępu na podstawie uprawnień oraz kontroli dostępu na podstawie zaufania do użytkownika. Pierwsza z nich jest rozszerzeniem tradycyjnego modelu RBAC do modelu CoRBAC (uwzględnia bieżący kontekst

w połączeniu z konfiguracją uprawnień), natomiast druga występuje tylko w modelu TCoRBAC (uwzględnia bieżący kontekst w połączeniu z profilem użytkownika). Obie kontrole dostępu są niezależne od siebie i mogą być wykonywane równolegle. Dzięki takiemu posunięciu zyskuje się pewne skrócenie czasu potrzebnego na kontrolę bezpieczeństwa, jednak z punktu widzenia użytkownika jest to niezauważalny uzysk. Dla czytelności dalszych rozważań przyjęto sekwencyjne ich wywołanie.

Podczas kontroli dostępu na podstawie uprawnień w pierwszej kolejności ma miejsce wyliczenie uprawnień przypisanych do użytkownika na podstawie konfiguracji uprawnień (macierze M_{UR} i M_{RP}). Odbywa się to identycznie, jak w tradycyjnym modelu RBAC. Następnie następuje redukcja (zawężenie) zbioru przypisanych do użytkownika uprawnień. Czynność ta występuje tylko w modelu CoRBAC i jest wykonywana na podstawie konfiguracji powiązania uprawnień z bieżącym kontekstem (macierze M_{PC} i M_{PCN}). Etap ten jest bardzo istotny podczas analizy ryzyka – redukuje zakres wpływu potencjalnego incydentu bezpieczeństwa. Ostatecznie ma miejsce weryfikacja uprawnień na podstawie reguł dostępu (macierz M_{PS}). Jeżeli weryfikacja jest pozytywna, to w modelu CoRBAC następuje dostęp do usługi, natomiast w modelu TCoRBAC uruchamiana jest kontrola dostępu na podstawie zaufania do użytkownika. Kontrola ta bierze pod uwagę ponownie bieżący kontekst działania oraz profil użytkownika (historię interakcji użytkownika z systemem). Głównym celem tej czynności jest wychwycenie nietypowego zachowania. Na podstawie wyliczonego poziomu zaufania do użytkownika (tl) uruchamiany jest odpowiedni scenariusz bezpieczeństwa. Dopiero prawidłowe wykonanie przez użytkownika uruchomionego scenariusza bezpieczeństwa pozwala na dostęp do usługi. Istotne jest zachowanie się użytkownika – częstość zmian lokalizacji i innych parametrów kontekstu, na podstawie których wyznaczany jest poziom zaufania do użytkownika (tl). Jeżeli zachowanie użytkownika jest stabilne, z jego punktu widzenia TCoRBAC jest niezauważalny.

Przytoczona dwuetapowa kontrola dostępu może być zaprezentowana również za pomocą następującego pseudo-kodu dla poszczególnych etapów:

```
permissionBasedAccessControl (user, service)
1  roles = getRolesOfUser(user)
2  userPermissions = getPermissionsOfRoles(roles)
3  currentContext = getCurrentContext()
4  contextPermissions = getContextPermissions(currentContext)
5  userPermissions = userPermissions ∩ contextPermissions
6  servicePermissions = getServicePermissions(service)
7  if(userPermissions ∩ servicePermissions != Ø)
8     then return GRANT_ACCESS (skip to userTrustBasedAccessControl)
9     else return DENY_ACCESS
```

```
userTrustBasedAccessControl (user, service)
1  currentContext = getCurrentContext()
2  userProfile = getUserProfile(user)
3  trustLevel = computeUserTrustLevel(userProfile, currentContext)
4  result = performUserSecurityCheck(trustLevel)
5  if(result == true)
6     then return GRANT_ACCESS
7     else return DENY_ACCESS
```

W rozdziale 5 zaprezentowano analizę profili użytkowników rzeczywistego systemu działającego na Politechnice Gdańskiej. Wykazano, że dzięki analizie kontekstu działań i porównywania go z profilem działań użytkownika zgodnie z modelem TCoRBAC opisanym powyżej, możliwe jest wdrożenie mechanizmów podnoszących bezpieczeństwo i wiarygodność systemu, które wyczulone są na sytuacje nietypowe. Odbywa się to bez znaczącej straty na użyteczności tegoż systemu.

Rozdział 4. Implementacja modelu kontekstowego

Zaprezentowano praktyczną implementację zaproponowanego modelu kontekstowego w Internetowej Platformie Integracji Politechniki Gdańskiej (IP^2). Przedstawiono zarówno wykorzystaną technologię jak i przyjętą architekturę systemu internetowego opartego na platformie IP^2 oraz opisane w poprzednich rozdziałach kontekstowe mechanizmy bezpieczeństwa. Podano zarówno metodę pozyskiwania kontekstu jak i jego efektywne wykorzystanie w rozpatrywanym systemie.

4.1 Platforma IP^2

Weryfikacja praktyczna zaproponowanego modelu kontekstowego została wykonana w ramach Internetowej Platformy Integracji Politechniki Gdańskiej IP^2. Jest to rozwiązanie integrujące większość systemów internetowych uczelni. Główny dostęp użytkownika do usług platformy IP^2 odbywa się przez system Moja PG [26]. Możliwy jest również dostęp do centralnych usług i danych poprzez inne systemy dziedzinowe zintegrowane z platformą. Niezwykle istotne było wprowadzenie jednej, integralnej bazy osób [145], w oparciu o którą działa system uwierzytelniania spójny dla wszystkich systemów [49]. Dzięki temu możliwa jest jednoznaczna identyfikacja użytkowników pomiędzy systemami.

Kolejnym etapem rozwoju systemu było wydzielenie modułu uwierzytelniającego i wdrożenie mechanizmu SSO (*ang. Single Sign On*). Jest to niezależna usługa, z którą integrowane są wszystkie systemy (w tym system Moja PG). Rozwiązanie takie posiada kilka zalet. Pierwszą jest wygoda użytkownika – użytkownik posiada tylko jeden login i hasło – nie musi pamiętać danych uwierzytelniających do każdego systemu osobno. Jest to istotne również pod kątem bezpieczeństwa – w przypadku, gdy trzeba pamiętać loginy i hasła do wielu systemów, w szczególności, gdy hasła te muszą być często zmieniane, dane te są trudne do zapamiętania i często zostają zapisane w sposób jawny, co znacząco obniża poziom bezpieczeństwa. Sam mechanizm jednokrotnego logowania (SSO) dodatkowo wpływa na wygodę użytkownika – dane uwierzytelniające podaje tylko raz, natomiast do kolejnych systemów w ramach jednej sesji jest już automatycznie logowany. Mechanizm ten jest nieoceniony w przypadku osadzania interfejsu jednych usług w interfejsach innych [146][147]. W ramach systemu Moja PG osadzonych jest kilka niezależnych usług, zlokalizowanych w innych systemach, do których logowanie odbywa się w sposób niezauważalny dla użytkownika. Uzupełnieniem mechanizmu jednokrotnego logowania SSO jest mechanizm Single Sign Out – jednoczesnego wylogowania we wszystkich systemach, w których użytkownik logował się w ramach danej sesji i wykonywał tam określone zadania.

Centralny Punkt Logowania PG (CPL PG) zrealizowany jest w oparciu o rozwiązanie Jasig CAS (Central Authentication Service) [54]. Dane uwierzytelniające przechowywane są w katalogu LDAP. Oryginalny serwer CAS został dostosowany do indywidualnych potrzeb Politechniki Gdańskiej. Wdrożono wiele standardowych modułów rozszerzających. Wprowadzono również autorskie modyfikacje mające na celu podniesie poziomu bezpieczeństwa tego kluczowego komponentu architektury bezpieczeństwa. Wyniesienie uwierzytelniania poza systemy do CPL PG umożliwiło wprowadzenie standardu uwierzytelniania na oczekiwanym poziomie. W ten sposób można się skupić na zabezpieczaniu pojedynczego miejsca w całej architekturze i uniknąć powielania tych samych błędów w wielu systemach. Jest to szczególnie istotne, gdy systemy są rozwijane przez niezależne, często zewnętrzne zespoły.

Każdy pojedynczy punkt dostępu w systemie może stać się wąskim gardłem. Dotyczy to zarówno wydajności, jak i dostępności w przypadku awarii. Wraz ze wzrostem liczby systemów zintegrowanych z CPL PG konieczne było podniesienie niezawodności tego systemu. Wykorzystano podejście polegające na skalowaniu poziomym poprzez klastrowanie n-węzłowe typu multi-master. To oznacza, że obciążenie jest rozkładane równo na wszystkie węzły. Jest to metoda równoważenia obciążania prowadząca do wzrostu wydajności całego systemu. Dodatkowo w momencie, gdy dowolny węzeł ulega awarii lub musi być wyłączony w związku z pracami serwisowymi, pozostałe węzły przejmują jego zadania. Również źródło danych zostało zwielokrotnione w trybie multi-master. Poszczególne węzły CPL PG korzystają z kilku źródeł danych, zależnie od ich obciążenia. Co również zwiększa bezpieczeństwo funkcjonowania systemu.

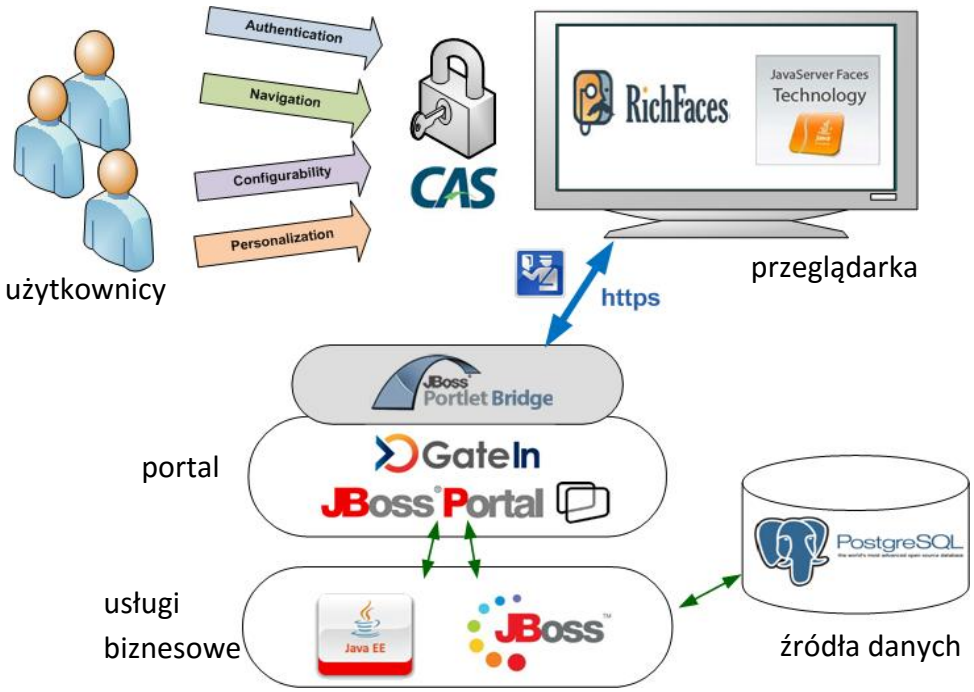
Zarządzanie danymi uwierzytelniającymi realizowane jest centralnie przez moduł bezpieczeństwa platformy IP^2. W ten sposób zachowana jest separacja uprawnień – CPL PG posiada dostęp do danych uwierzytelniających w trybie tylko do odczytu. Zapis (dodawanie i modyfikacja) natomiast jest możliwy tylko poprzez platformę IP^2. To w ramach niej znajdują się mechanizmy dostępne dla użytkownika typu zmiana hasła, aktywacja, reset hasła. Odpowiednio zabezpieczony interfejs administracyjny pozwala na sterowanie dowolnymi kontami. Co istotne: każda operacja na koncie, niezależnie czy wykonywana przez użytkownika, czy administratora, zostaje odnotowana w rejestrze zdarzeń. Pozwala to na śledzenie zmian na koncie użytkownika, szczególnie pomocne podczas analizy powłamaniowej. Dodatkowo każda operacja logowania (zarówno poprawnego, jak i odrzuconego) oraz wylogowania jest odnotowywana w stosownym rejestrze powiązanym z tym kontem. Na podstawie tych danych budowany jest profil bezpieczeństwa użytkownika.

Cała platforma została wykonana w technologiach otwartych (*ang. open source*). Główną technologią jest Java Enterprise Edition 6 [27]. W niej zaimplementowano całą logikę biznesową (usługi). Dostarcza ona również mechanizmy katalogowania i wyszukiwania usług. Ze względu na wielkość systemu usługi zostały pogrupowane w komponenty tematyczne. Najistotniejsze to te realizujące funkcjonalność związaną z zarządzaniem wspólną bazą osób, warstwą bezpieczeństwa, szeroko pojętą dydaktyką i sferą naukowo-badawczą [148].

Są również mniejsze, o charakterze nieewidencyjnym, jak np. komponent odpowiedzialny za powiadomienia mailowe i smsowe, czy inny wspomagający generację wszelkiego rodzaju wydruków [149].

Głównym źródłem danych jest relacyjna baza PostgreSQL [150] (również dostępna na licencji open source). Oprócz niej występuje wiele innych typów źródeł: bazy relacyjne, zasoby plikowe, repozytoria dokumentów cyfrowych. Wolumen przechowywanych i przetwarzanych danych jest bardzo duży – sama główna baza SQL w chwili obecnej zajmuje ponad 100 GB. Operowanie na takim dużym wolumenie informacji jest sporym wyzwaniem w kontekście zapewnienia bezpieczeństwa zarówno w sferze kontroli dostępu (*ang. security*), jak i zapewnienia ciągłości działania (*ang. safety*).

Komunikacja systemów wydziałowych i dziedzinowych z platformą centralną jest możliwa na kilka różnych sposobów. Podstawowym punktem kontaktowym jest bramka API obsługująca żądania w protokołach XML-RPC, JSON-RPC oraz wołania REST. Ponadto możliwa jest komunikacja protokołami EJB Remoting i WS SOAP. Taka szeroka gama możliwych sposobów komunikacji pozwala na sprawną integrację wielu systemów z centralną platformą. Warto nadmienić, że system Moja PG, będący głównym interfejsem użytkownika do tej platformy, z punktu widzenia platformy jest również traktowany jako system kliencki. Zatem wszystkie żądania systemów klienckich (a więc systemów zintegrowanych, jak i systemu Moja PG) przechodzą przez tą samą warstwę bezpieczeństwa.



Rysunek 22 Technologia platformy IP[2]

System Moja PG to kontener portletów z silnikiem JBoss Portal. Poszczególne moduły interfejsu są oddzielnymi, niezależnymi od siebie portletami. Wykonane są również w technologii Java Enterprise Edition z wykorzystaniem Java Server Faces. Jak w każdym systemie internetowym intensywnie wykorzystywany jest JavaScript i technologia AJAX. Zadanie to ułatwia wykorzystanie gamy gotowych komponentów dedykowanych temu rozwiązaniu stanowiących również własne rozwinięcia biblioteki RichFaces.

Rysunek 22 ilustruje zastosowane technologie w platformie IP^2. Warto zauważyć, że wszystkie rozwiązania dostępne są na licencji open source. Dzięki zastosowaniu niezależnych od technologii standardów komunikacji możliwa jest integracja z systemami własnościowymi, zrealizowanymi w innych technologiach. Również powiązania pomiędzy poszczególnymi elementami platformy (CPL PG, portal Moja PG, serwer logiki biznesowej, baza danych) pozwalają na wymianę poszczególnych komponentów na zrealizowane w innej technologii bez większych nakładów pracy – przykładowo dotyczyć może wymiany silnika bazy danych na oprogramowanie dostarczane przez firmę Oracle.

4.2 Architektura rozpatrywanego systemu

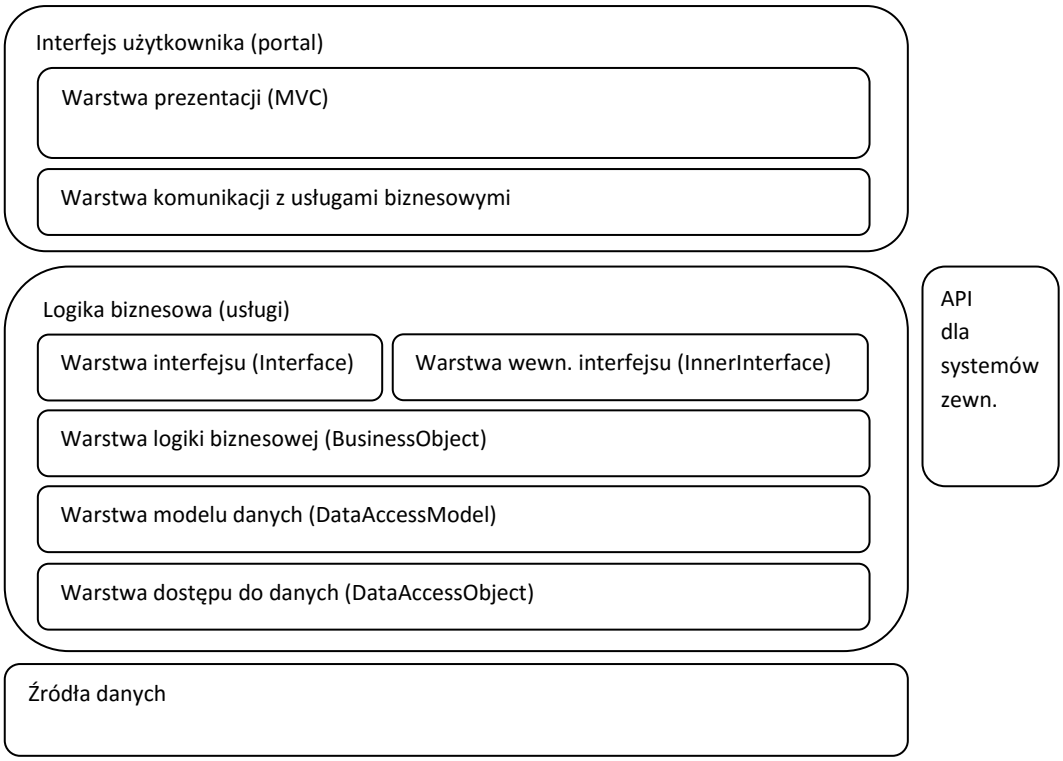
Platforma IP^2 została zaprojektowana jako system rozproszony, gdzie rozproszenie to jest realizowane w kilku wymiarach. Pierwszym z nich jest rozdzielenie interfejsu użytkownika od logiki biznesowej (usług) i zlokalizowanie ich na różnych serwerach. Na osobnych serwerach znajdują się również źródła danych. Dodatkowo każdy z tych komponentów może podlegać dalszemu podziałowi na niezależne serwery. Dodatkowo platforma integruje wiele innych systemów poprzez bogate API oraz dostęp do wielu zewnętrznych źródeł danych.

Każdy komponent architektury taki jak interfejs użytkownika, logika biznesowa, API, baza danych zaprojektowany został wielowarstwowo (patrz Rysunek 23). Interfejs użytkownika (portal) w warstwie prezentacji jest zgodny ze wzorcem MVC (*ang. model-view-controller*)[151] oraz posiada dobrze wydzieloną warstwę komunikacji z usługami logiki biznesowej.

Logika biznesowa bazuje na modelu usługowym przy czym pewne usługi udostępniane są przez Beany EJB lokalnie (pomiędzy usługami), inne zdalnie poprzez protokół EJB Remoting (na potrzeby interfejsu użytkownika i innych usług „klienckich"), a jeszcze inne zdalnie poprzez API i wykorzystanie protokołów uniezależniających od technologii (JSON-RPC, XML-RPC, SOAP).

Ze względu na wielkość platformy i jej złożoność logika biznesowa została podzielona na współpracujące ze sobą moduły obejmujące funkcjonalnością różne dokładnie określone obszary tematyczne, takie jak: dydaktyka, współpraca i innowacje, badania i rozwój [145][152]. Każdy z modułów logiki biznesowej posiada również wydzielone warstwy: uwierzytelniania i kontroli dostępu do usług, interfejsu (Interface), wewnętrznego interfejsu (InnerInterface), logiki biznesowej (BusinessObject), modelu danych (DataAccessModel) oraz

dostępu do danych (DataAccessObject). Podejście takie sprzyja utrzymaniu względnego porządku w kodzie systemu oraz stosunkową łatwość wymiany technologii poszczególnych elementów platformy przy minimalnej ingerencji w pozostałe. Rysunek 23 prezentuje podstawowe warstwy platformy IP^2. Stanowi ona bazę systemu internetowego Moja PG.



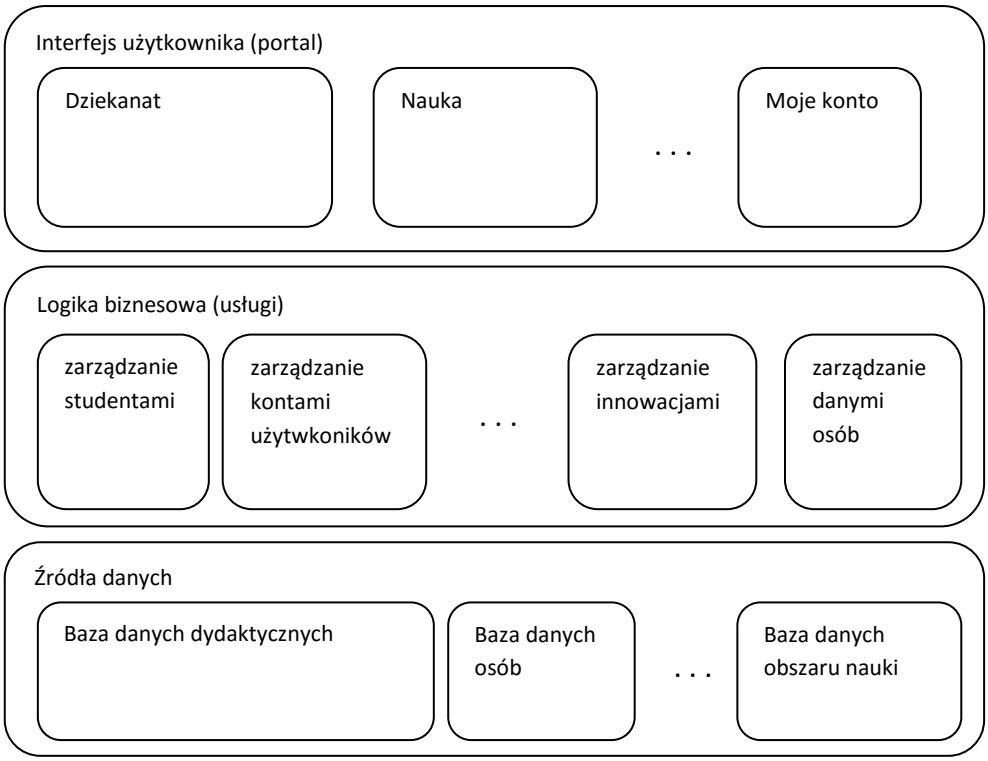
Rysunek 23 Model warstwowy platformy IP^2

Rysunek 24 z kolei przedstawia zasadę modularyzacji poszczególnych elementów platformy. Modularyzacja ta jest obecna we wszystkich trzech głównych warstwach platformy. Z punktu widzenia użytkownika najbardziej widoczna jest modularność interfejsu użytkownika – poszczególne moduły to odrębne aplikacje uruchamiane w ramach jednego portalu powiązane ze sobą jedynie sesją użytkownika. Oczywiście jednocześnie może być kreowane wiele takich sesji na różnych serwerach.

Modularyzacja w logice biznesowej polega na wyróżnieniu, jak już wcześniej wspomniano, dwóch kategorii usług: podstawowe świadczące usługi dla systemów klienckich (w tym portalu), bądź na rzecz innych usług usług (usługi pomocnicze). Moduły te można pogrupować ze względu na ich charakter na: ewidencyjne zarządzające obszarem funkcjonalnym oraz narzędziowe wspierające pierwsze (np. zapewniające komunikację e-mail, GSM lub generujące wydruki).

Również w warstwie danych zaznacza się organizację modułową. Realizowane jest to w dwojaki sposób: poprzez tworzenie odrębnych baz danych, albo poprzez podział danych

na mniejsze obszary (schemy). W tym drugim przypadku ziarnistość odpowiada ziarnistości modułów logiki biznesowej – każdy z nich posiada własny obszar danych, który tylko on może modyfikować, a pozostałe moduły tylko ew. czytać. Rozwiązanie takie zapewnia większą spójność danych w ramach jednego modułu.

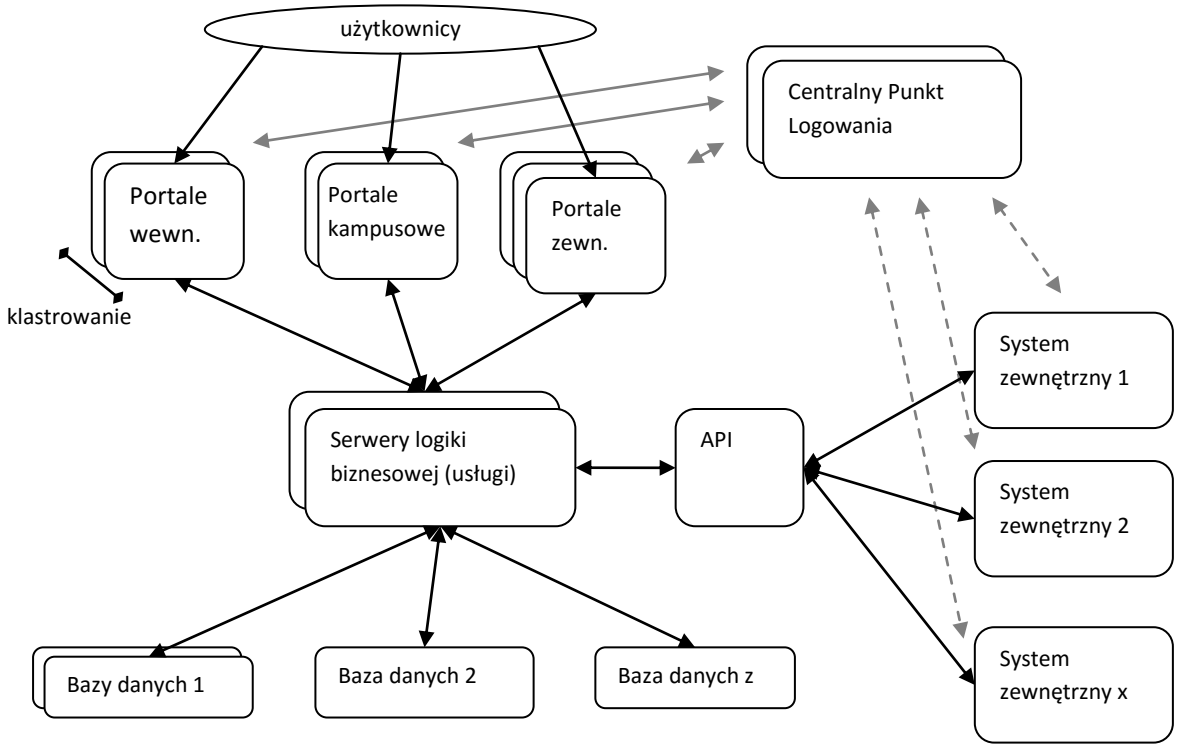


Rysunek 24 Wyszczególnienie podstawowych modułów platformy IP²

Architektura platformy IP² wykorzystuje opisane w rozdziale 2.3.2 wyniesione uwierzytelnianie i mieszany (dwustopniowy) model zarządzania uprawnieniami. W takim modelu zostało zaprojektowane zarządzanie uprawnieniami zarówno w platformie IP² jak i systemach powiązanych.

Rysunek 25 przedstawia główne elementy platformy. Są to 3 portale umieszczone w różnych sieciach (wydzielona sieć wewnętrzna, sieć uczelniana i Internet), serwer logiki biznesowej, bazy danych oraz API dla systemów zewnętrznych. Portale odpowiedzialne są jedynie za budowę interfejsu użytkownika – jest to system internetowy, więc wymaga stworzenia widoku użytkownika po stronie serwera. Zaszyto w nich również scenariusze wołań usług biznesowych, tak aby osiągnąć oczekiwany przez użytkowników efekt ich działań w systemie. Jednak wszystkie procesy biznesowe oraz obróbka żądań i danych odbywa się dopiero w serwerze logiki biznesowej. Analizując w ten sposób architekturę można przyjąć, że zarówno portale, jak i wszystkie systemy zewnętrzne łączące się za pomocą API, pełnią rolę kliencką wobec usług świadczonych przez serwer logiki biznesowej.

Poszczególne elementy platformy podlegają klastrowaniu (oznaczone na rysunku poprzez powielenie danego elementu). Dzięki skalowaniu poziomemu uzyskiwana jest w pierwszej kolejności niezawodność działania – upadek jednego z węzłów nie wpływa na pracę innych, które automatycznie przejmują obsługę żądań przychodzących do uszkodzonego węzła. Bliźniacze węzły najczęściej zlokalizowane są na różnych fizycznych maszynach co zwiększa niezawodność na wypadek awarii fizycznej jednej z maszyn. Efektem uzyskanym przy okazji jest zwiększenie wydajności całości.



Rysunek 25 Uproszczony model rozproszenia i skalowania poziomego poszczególnych elementów platformy IP[2]

4.3 Algorytm działania warstwy bezpieczeństwa uwzględniającej kontekst

Algorytm działania warstwy bezpieczeństwa uwzględniającego kontekst i poziom zaufania do użytkownika został zaprezentowany poniżej w postaci obiektowego kodu Java. Przedstawiono algorytm zarówno dla tradycyjnego modelu RBAC (linia 21), jak i dla rozszerzonego modelu TCoRBAC (linia 52). Model TCoRBAC zawiera w sobie model CoRBAC. Przytoczono tylko klasy najistotniejsze pod kątem algorytmu działania. Pozostałe klasy oraz przykładowa implementacja klas abstrakcyjnych zaprezentowano w kolejnych punktach.

Najistotniejszą klasą jest *SecurityInterceptor* zabezpieczający każde wołanie metod oferowanych przez usługi. Wykorzystuje on usługę bezpieczeństwa *SecurityService* (przedstawioną za pomocą interfejsu *ISecurityService*), która odpowiada za uruchamianie odpowiednich scenariuszy bezpieczeństwa (ss). Kolejnym elementem jest *SecurityConfiguration* (przedstawiony za pomocą interfejsu *ISecurityConfiguration*), który realizuje konfigurację uprawnień zgodną z modelem RBAC (powiązanie użytkownika z rolami i uprawnieniami oraz powiązanie usług z uprawnieniami). Ostatnim elementem jest usługa *ContextService* operująca na kontekście (*Context*) składającym się z parametrów kontekstu (*ContextParameter*). To w tej usłudze realizowana jest analiza bieżącego kontekstu c_x oraz wyznaczanie aktualnego poziomu zaufania do użytkownika tl_x. Bardziej szczegółową analizę można przeprowadzić analizując komentarze w kodzie.

SecurityInterceptor

```
1   package pl.lubomski.corbac;
2
3   import java.util.HashSet;
4   import java.util.List;
5   import java.util.Set;
6
7   public class SecurityInterceptor {
8
9           private ISecurityService securityService;
10          private ISecurityConfiguration securityConfiguration;
11          private ContextService contextService;
12
13          /**
14           * Metoda sprawdza, czy pozwolić na dostęp zgodnie z modelem RBAC
15           *
16           * @param req
17           *            Żądanie użytkownika zawierające jego sesję oraz usługę, do
18           *            której chce uzyskać dostęp
19           * @return true - dostęp udzielony, false - dostęp zabroniony
20           */
21          public boolean canGrantAccessRBAC(Request req) {
22                  //wyznacz zbiór uprawnień użytkownika P(u)
23                  Set<Permission> pu = this.securityConfiguration
24                                  .getUserPermissions(req.getUserSession().getUser());
25                  //wyznacz zbiór uprawnień usługi P(s)
26                  Set<Permission> ps = this.securityConfiguration
27                                  .getServicePermissions(req.getService());
28                  //wyznacz część wspólną zbiorów P(u) i P(s)
29                  Set<Permission> resultPerm = new HashSet<Permission>();
30                  for (Permission p : ps) {
31                          if (pu.contains(p)) {
32                                  resultPerm.add(p);
33                          }
34                  }
35                  //
36                  //sprawdzenie, czy zostało co najmniej jedno wspólne uprawnienie
37                  if (resultPerm.size() > 0) {
38                          return true;
39                  } else {
40                          return false;
41                  }
42          }
43
44          /**
45           * Metoda sprawdza, czy pozwolić na dostęp zgodnie z modelem TCoRBAC
46           *
47           * @param req
48           *            Żądanie użytkownika zawierające jego sesję oraz usługę, do
49           *            której chce uzyskać dostęp
50           * @return true - dostęp udzielony, false - dostęp zabroniony
51           */
```

```
52          public boolean canGrantAccessTCoRBAC(Request req) {
53                  //wyznacz zbiór uprawnień użytkownika P(u)
54                  Set<Permission> pu = this.securityConfiguration
55                                  .getUserPermissions(req.getUserSession().getUser());
56                  //wyznacz zbiór uprawnień usługi P(s)
57                  Set<Permission> ps = this.securityConfiguration
58                                  .getServicePermissions(req.getService());
59                  //wyznacz część wspólną zbiorów P(u) i P(s)
60                  Set<Permission> resultPerm = new HashSet<Permission>();
61                  for (Permission p : ps) {
62                          if (pu.contains(p)) {
63                                  resultPerm.add(p);
64                          }
65                  }
66                  //wyznacz aktualny kontekst dla żądania
67                  Set<ContextParameterValue> currentContext = this.contextService
68                                  .determineCurrentContext(req);
69                  //wyznacz poziom zaufania (TL) dla żądania
70                  int reqTL = this.contextService
71                                  .getContextTrustLevel(currentContext);
72                  //wyznacz uprawnienia dopuszczalne w danym kontekście
73                  Set<Permission> pc = this.contextService.getContextPermissions(
74                                  currentContext, reqTL);
75                  //
76                  //wyznacz część wspólną zbiorów resultPerm i P(c)
77                  for (Permission p : resultPerm) {
78                          if (!pc.contains(p)) {
79                                  resultPerm.remove(p);
80                          }
81                  }
82                  //
83                  //sprawdzenie, czy zostało co najmniej jedno wspólne uprawnienie
84                  if (resultPerm.size() > 0) {
85                          List<Action> actionLog = req.getUserSession()
86                                          .getActionLog();
87                          boolean result = true;
88                          if (actionLog.size() == 0) {
89                                  //pierwsze żądanie użytkownika
90                                  result = this.securityService
91                                                  .performInitialScenario(reqTL);
92                          } else {
93                                  //nie pierwsze żądanie użytkownika
94                                  int prevTrustLevel = actionLog
95                                                  .get(actionLog.size() - 1).getTrustLevel();
96                                  if (prevTrustLevel != reqTL) {
97                                          result = this.securityService.performScenario(
98                                                          prevTrustLevel, reqTL);
99                                  }
100                         }
101                         //czy scenariusz bezpieczeństwa został wykonany poprawnie?
102                         if (result) {
103                                 //odkładamy w logu wykonywanych akcji
104                                 req.getUserSession().addActionToLog(
105                                                 new Action(req.getService(), reqTL));
106                                 return true;
107                         }
108                 }
109                 return false;
110         }
111 }
```

ISecurityConfiguration

```
1   package pl.lubomski.corbac;
2
3   import java.util.Set;
4
5   public interface ISecurityConfiguration {
6
7           /**
8            * Metoda wyznaczająca zbiór uprawnień przypisanych w konfiguracji do
9            * użytkownika na podstawie powiązania U-R-P lub uproszczonego U-P
10           *
11           * @param user
12           *            Konkretny użytkownik U
```

```
13          * @return Zbiór uprawnień przypisanych użytkownikowi
14          */
15         public Set<Permission> getUserPermissions(User user);
16
17         /**
18          * Metoda wyznaczająca zbiór uprawnień przypisanych w konfiguracji do usługi
19          * na podstawie powiązania P-S
20          *
21          * @param s
22          *            Konkretna usługa S
23          * @return Zbiór uprawnień przypisanych usłudze
24          */
25         public Set<Permission> getServicePermissions(Service s);
26 }
27
```

ContextService

```
1  package pl.lubomski.corbac;
2
3  import java.util.Map;
4  import java.util.Set;
5
6  public class ContextService {
7
8          /**
9           * Kontekst
10          */
11         private Context context;
12         /**
13          * Konfiguracja macierzy MPCN w zakresie przypadków dokładnie określnych
14          * (zbiory uprawnień dla poszczególnych wartości kontekstu reprezentowanego
15          * jako zbiór wartości parametrów kontekstu)
16          */
17         private Map<Set<ContextParameterValue>, Set<Permission>> contextPermissionConfig;
18         /**
19          * Konfiguracja macierzy MPCN w zakresie przypadków przybliżanych (zbiory
20          * uprawnień dla poszczególnych TL)
21          */
22         private Map<Integer, Set<Permission>> contextTrustLevelLConfig;
23
24         public Set<ContextParameterValue> determineCurrentContext(
25                         Request request) {
26                 return this.context.determineCurrentContext(request);
27         }
28
29         public Set<Permission> getContextPermissions(
30                         Set<ContextParameterValue> currentContext,
31                         int currentTrustLevel) {
32                 if (this.contextPermissionConfig.containsKey(currentContext)) {
33                         //przypadek dokładnie określony
34                         return this.contextPermissionConfig.get(currentContext);
35                 } else {
36                         //przypadek "przybliżany" za pomocą TL
37                         return this.contextTrustLevelLConfig.get(currentTrustLevel);
38                 }
39         }
40
41         public int getContextTrustLevel(
42                         Set<ContextParameterValue> currentContext) {
43                 //inicjalnie najniższy
44                 int trustLevel = 1;
45                 //wyznaczamy wartość minimalną
46                 for (ContextParameterValue value : currentContext) {
47                         if (value.getTrustLevel() < trustLevel) {
48                                 trustLevel = value.getTrustLevel();
49                         }
50                 }
51                 return trustLevel;
52         }
53 }
```

Context

```
1  package pl.lubomski.corbac;
2
3  import java.util.HashSet;
4  import java.util.Set;
5
6  public class Context {
7
8          private Set<ContextParameter> contextParameters;
9
10         public Set<ContextParameter> getContextParameters() {
11                 return this.contextParameters;
12         }
13
14         public Set<ContextParameterValue> determineCurrentContext(Request request) {
15                 Set<ContextParameterValue> result = new HashSet<ContextParameterValue>();
16                 for (ContextParameter cp : this.contextParameters) {
17                         result.add(cp.computeCurrentValue(request));
18                 }
19                 return result;
20         }
21 }
22
```

ContextParameter

```
1  package pl.lubomski.corbac;
2
3  import java.util.Set;
4
5  public abstract class ContextParameter {
6
7          /**
8           * Nazwa parametru kontekstu
9           */
10         private String name;
11
12         public ContextParameter() {
13                 this.fillAllPossibleValues();
14         }
15
16         public abstract void fillAllPossibleValues();
17
18         /**
19          * Abstrakcyjna metoda, która zwraca zbiór wszystkich możliwych wartości
20          * danego parametru kontekstu
21          *
22          * @return Zbiór możliwych wartości danego parametru kontekstu
23          */
24         public abstract Set<ContextParameterValue> getAllPossibleValues();
25
26         /**
27          * Abstrakcyjna metoda, która wyznacza dla danego parametru kontekstu CP
28          * jego aktualną wartość na podstawie żądania użytkownika oraz innych
29          * przesłanek
30          *
31          * @param request
32          *            Żądanie użytkownika
33          * @return Bieżąca wartość parametru kontekstu
34          */
35         public abstract ContextParameterValue computeCurrentValue(
36                         Request request);
37
38         public String getName() {
39                 return this.name;
40         }
41 }
```

ContextParameterValue

```
1  package pl.lubomski.corbac;
2
3  public class ContextParameterValue {
4
5          /**
6           * Nazwa wartości parametru kontekstu
7           */
8          private final String name;
9          /**
10          * Poziom bezpieczeństwa przypisany do tej wartości parametru kontekstu
11          */
12         private final int trustLevel;
13
14         public ContextParameterValue(String name, int trustLevel) {
15                 this.name = name;
16                 this.trustLevel = trustLevel;
17         }
18
19         public String getName() {
20                 return this.name;
21         }
22
23         public int getTrustLevel() {
24                 return this.trustLevel;
25         }
26 }
```

ISecurityService

```
1  package pl.lubomski.corbac;
2
3  public interface ISecurityService {
4
5          /**
6           * Usługa bezpieczeństwa (Ss) wykonywana przy pierwszym żądaniu użytkownika
7           * w ramach sesji. Usługa realizuje konkretny scenariusz bezpieczeństwa (SS)
8           * w zależności od poziomu bezpieczeństwa bieżącego kontekstu.
9           *
10          * Np. wprowadzenie kodu CAPTCHA
11          *
12          * @param trustLevel
13          *            Poziom bezpieczeństwa wliczony z bieżącego kontekstu
14          * @return true - prawidłowo wykonany scenariusz (prawidłowa odpowiedź),
15          *         false - błędnie
16          */
17         public boolean performInitialScenario(int trustLevel);
18
19         /**
20          * Usługa bezpieczeństwa (Ss) wykonywana kolejnego żądaniu użytkownika w
21          * ramach sesji, jeżeli zmienił się poziom bezpieczeństwa od czasu
22          * poprzedniego żądania. Usługa realizuje konkretny scenariusz
23          * bezpieczeństwa (SS) w zależności od poprzedniego i bieżacego poziomu
24          * bezpieczeństwa bieżącego kontekstu.
25          *
26          * Np. wprowadzenie kodu CAPTCHA
27          *
28          * @param previousTrustLevel
29          *            Poziom bezpieczeństwa wliczony z bieżącego kontekstu podczas
30          *            poprzedniego żądania
31          * @param trustLevel
32          *            Poziom bezpieczeństwa wliczony z bieżącego kontekstu
33          * @return true - prawidłowo wykonany scenariusz (prawidłowa odpowiedź),
34          *         false - błędnie
35          */
36         public boolean performScenario(int previousTrustLevel,
37                         int trustLevel);
38 }
```

4.4 Analiza bieżącego kontekstu

Metoda pozyskiwania kontekstu dekomponuje się na niezależne zadania pozyskania wartości poszczególnych parametrów kontekstu (CP). Zależenie od charakteru i dziedziny poszczególnych parametrów kontekstu można wskazać następujące źródła kontekstu:

- żądania użytkownika (*ang. user request*) – nagłówki HTTP (Referer, Cookie, User-Agent, …),
- parametry połączenia (np. adres IP),
- czasy wystąpowania zdarzeń,
- stany systemu.

Najszerszym źródłem informacji dotyczących kontekstu w przypadku systemów internetowych są nagłówki protokołu HTTP. Są to metadane przesyłanego żądania (*ang. request*) dołączane przez przeglądarkę WWW użytkownika. Kompletną listę standardowych nagłówków HTTP definiują standard RFC 2616 [153] oraz RFC 4229 [154] z późniejszymi uzupełnieniami.

Istotnym jest fakt, że dane te mogą być modyfikowane przez np. wtyczki do przeglądarki. W związku z tym należy bardzo ostrożnie interpretować te informacje, ponieważ system bezpieczeństwa może być świadomie oszukiwany przez zaawansowanego klienta/użytkownika. Warto zastosować dodatkowe mechanizmy weryfikacji przesyłanych informacji, np. szerokorozumiana poprawność identyfikatora sesji, czy zestawić je z innymi, niemożliwymi do modyfikacji po stronie użytkownika parametrami (np. z adresem IP klienta).

Analiza parametrów połączenia może być realizowana na różnych warstwach modelu OSI [155][156][157]. Najczęściej analizowany jest adres IP źródła żądania oraz dodatkowe parametry przesyłane w treści żądania (niekoniecznie jako nagłówki protokołu HTTP), jak np. aktualne dane geolokalizacyjne pobrane z modułu GPS urządzenia, siła i BSSID dostępnych sieci Wi-Fi, specjalne tokeny uwierzytelniające czy też pozwalające na śledzenie aktywności użytkownika.

Istnieje grupa parametrów kontekstu, których wartości można wyznaczyć niezależnie od żądania użytkownika. Przykładem takiego parametru kontekstu jest czas wystąpienia żądania. Jest on pobierany w sposób autonomiczny z zegara systemowego, a następnie na jego podstawie jest wyznaczana wartość parametru kontekstu (odpowiednia kwantyzacja) – patrz pkt. 3.3. Czas może być analizowany również w połączeniu z żądaniami użytkownika, np. odstęp czasu od ostatniego żądania tego użytkownika, lub również w połączeniu z historią akcji wykonywanych w systemie, np. czy jest to standardowa godzina, gdy użytkownik wykonuje tą akcję w systemie?

Osobną grupą parametrów kontekstu są parametry związane z bieżącym stanem systemu. Obejmują one np. bieżące obciążenie systemu, tryb pracy (normalna, serwisowa, awaria),

poziom rozproszenia klastra, itp. Wartości tych parametrów w większości muszą być wyznaczane w oparciu o wbudowane w system mechanizmy monitorujące.

4.5 Implementacja kontekstowych mechanizmów bezpieczeństwa

W kolejnych punktach opisano mechanizmy bezpieczeństwa (o różnym stopniu skomplikowania) uwzględniające kontekst, które zostały wdrożone w platformie IP^2. Zlokalizowane są one w różnych miejscach architektury platformy. Mechanizmy te przeważnie wykazują „wrażliwość" tylko na jeden parametr kontekstu (CP). Oznacza to, że pozostałe parametry kontekstu w danym mechanizmie nie wpływają na wynik – macierz M_{PC} zwiera 1 niezależnie od aktualnych wartości tych parametrów kontekstu. Jest to wersja uproszczona w stosunku do zaproponowanego modelu ze względu na łatwiejszą implementację, jednak podlega ona iteracyjnemu rozszerzaniu o kolejne parametry kontekstu. Tabela 15 prezentuje zestawienie analizowanych parametrów kontekstu oraz ich wartości.

Tabela 15 Analizowane parametry kontekstu i ich wartości

Parametr kontekstu	Wartości parametru
logiczna lokalizacja użytkownika	wydzielona sieć wewnętrzna, sieć uczelniana, sieć zewnętrzna (Internet)
czas wystąpienia żądania użytkownika	poszczególne dni tygodnia: poniedziałek, wtorek, …
tryb pracy systemu	praca normalna, prace serwisowe
adres IP usługi/systemu klienckiego	prawidłowe wartości adresu IPv4
moduł logiki biznesowej	studentManager, personManager, dictManager, securityManager, …
dostawca i wersja używanej przeglądarki	dostępne wersje przeglądarek internetowych
scenariusz działań użytkownika	sekwencja akcji w systemie (profil użytkownika)
moment czasu wystąpienia żądania	prawidłowa data i godzina (z dokładnością do milisekund)

Szczegółowy opis mechanizmów bezpieczeństwa wykorzystujących te parametry oraz kod ich implementacji został opisany w kolejnych punktach. Zapisy zostały tak uogólnione, żeby nie zdradzać szczegółów mających istotny wpływ na bezpieczeństwo działającego systemu. Jednak przyjęty poziom szczegółowości powinien w pełni wystarczyć, żeby zrozumieć zasadę działania omawianych mechanizmów kontekstowych.

4.5.1 Logiczna lokalizacja użytkownika

Analizowany parametr kontekstu przyjmuje następującą postać:

Parametr kontekstu	logiczna lokalizacja użytkownika
Wartości parametru	wydzielona sieć wewnętrzna, sieć uczelniana, sieć zewnętrzna (Internet)

Identyfikacja logicznej lokalizacji użytkownika jest jednym z podstawowych kontekstowych mechanizmów bezpieczeństwa zaimplementowanych w platformie IP^2. Zgodnie z opisem architektury zaprezentowanym w rozdziale 4.2 użytkownik może korzystać z platformy poprzez jeden z 3 portali Moja PG zlokalizowanych w różnych sieciach. Zastosowano podział na 3 logiczne sieci: wewnętrzna sieć wydzielona, sieć uczelniana (politechniczna), sieć zewnętrzna (Internet). Na podstawie sieci, z której korzysta użytkownik, wyznaczana jest jego logiczna lokalizacja. W znaczącym stopniu lokalizacja ta pokrywa się z lokalizacją fizyczną z pewnym wyjątkiem: możliwości zastosowania połączeń VPN w dostępie do sieci uczelnianej „z zewnątrz".

Mechanizm został tak skonfigurowany, aby sieci te tworzyły hierarchię od najbardziej zaufanej do najmniej [35]. Im sieć jest mniej zaufana, tym mniej uprawnień z nią powiązanych, a więc tym mniej operacji można z niej wykonać. W praktyce z Internetu możliwy jest dostęp dla studentów i podstawowych innych danych w trybie tylko do odczytu. Z sieci uczelnianej dodatkowo możliwe jest wykonanie większości operacji wykonywanych przez pracowników. Operacje krytyczne pod kątem bezpieczeństwa systemu (administracja) oraz modyfikacji najistotniejszych i w pewnym sensie wrażliwych danych przechowywanych w systemie możliwe są jedynie z wydzielonej sieci wewnętrznej.

Poniżej przedstawiono w postaci kodu Java implementację parametru kontekstu logicznej lokalizacji użytkownika. Klasa ta implementuje abstrakcyjną klasę *ContextParameter* przytaczaną w punkcie 4.3.

CPLogicalUserLocalization

```
1   package pl.lubomski.corbac.impl;
2
3   import java.net.InetAddress;
4   import java.util.HashSet;
5   import java.util.Hashtable;
6   import java.util.Set;
7   import pl.lubomski.corbac.ContextParameter;
8   import pl.lubomski.corbac.ContextParameterValue;
9   import pl.lubomski.corbac.Request;
10
11  public class CPLogicalUserLocalization extends ContextParameter {
12
13          private enum Zone {
14                  INTERNAL, CAMPUS, EXTERNAL
15          }
16          private Hashtable<Zone, ContextParameterValue> values;
17
18          @Override
19          public ContextParameterValue computeCurrentValue(Request request) {
20                  InetAddress remoteIPAddress = request.getRemoteIPAddress();
21                  //
22                  if (remoteIPAddress.getHostAddress().startsWith("10.0.0.")) {
23                          return this.values.get(Zone.INTERNAL);
```

```
24                  } else if (remoteIPAddress.getHostAddress().startsWith(
25                                  "153.19.")) {
26                          return this.values.get(Zone.CAMPUS);
27                  } else {
28                          return this.values.get(Zone.EXTERNAL);
29                  }
30          }
31
32          @Override
33          public void fillAllPossibleValues() {
34                  this.values = new Hashtable<>();
35                  this.values.put(Zone.INTERNAL, new ContextParameterValue(
36                                  "sieć wewnętrzna", 3));
37                  this.values.put(Zone.CAMPUS, new ContextParameterValue(
38                                  "sieć uczelniana", 2));
39                  this.values.put(Zone.EXTERNAL, new ContextParameterValue(
40                                  "internet", 1));
41          }
42
43          @Override
44          public Set<ContextParameterValue> getAllPossibleValues() {
45                  return new HashSet<ContextParameterValue>(this.values.values());
46          }
47  }
```

4.5.2 Czas wystąpienia żądania

Analizowany parametr kontekstu jest następującej postaci:

Parametr kontekstu	czas wystąpienia żądania użytkownika
Wartości parametru	poszczególne dni tygodnia: poniedziałek, wtorek, …

Kolejnym najczęściej weryfikowanym parametrem kontekstu analizowanym przez systemu z kontekstową kontrolą dostępu jest czas wystąpienia żądania użytkownika. Czas ten jest kwantyfikowany do poszczególnych dni tygodnia, godzin roboczych i wolnych, itp. W przetaczanym przypadku zdecydowano o kwantyzacji czasu z dokładnością do dni tygodnia. Rozwiązanie jest bardzo uproszczone, ponieważ nie uwzględnia dni wolnych od pracy (święta) oraz innych uwarunkowań wynikających z kalendarza.

Mechanizm został tak skonfigurowany, aby pełny dostęp do systemu był w dni robocze (poniedziałek – piątek). W sobotę występuje pewne ograniczenie dostępnej funkcjonalności, natomiast w niedzielę zostają dostępne praktycznie tylko pojedyncze usługi, w większości działające w trybie „tylko do odczytu”.

Poniżej przedstawiono w postaci kodu Java implementację parametru kontekstu czasu wystąpienia żądania użytkownika. Klasa ta implementuje abstrakcyjną klasę *ContextParameter* przytaczaną w punkcie 4.3.

CPDayOfWeek

```
1   package pl.lubomski.corbac.impl;
2
3   import java.util.Calendar;
4   import java.util.HashSet;
5   import java.util.Hashtable;
6   import java.util.Set;
7   import pl.lubomski.corbac.ContextParameter;
8   import pl.lubomski.corbac.ContextParameterValue;
```

```
9   import pl.lubomski.corbac.Request;
10
11  public class CPDayOfWeek extends ContextParameter {
12
13          private Hashtable<Integer, ContextParameterValue> values;
14
15          @Override
16          public ContextParameterValue computeCurrentValue(Request request) {
17                  int dayOfWeek = Calendar.getInstance()
18                                  .get(Calendar.DAY_OF_WEEK);
19                  //
20                  return this.values.get(dayOfWeek);
21          }
22
23          @Override
24          public void fillAllPossibleValues() {
25                  this.values = new Hashtable<>();
26                  this.values.put(Calendar.MONDAY, new ContextParameterValue(
27                                  "MONDAY", 3));
28                  this.values.put(Calendar.TUESDAY, new ContextParameterValue(
29                                  "TUESDAY", 3));
30                  this.values.put(Calendar.WEDNESDAY, new ContextParameterValue(
31                                  "WEDNESDAY", 3));
32                  this.values.put(Calendar.THURSDAY, new ContextParameterValue(
33                                  "THURSDAY", 3));
34                  this.values.put(Calendar.FRIDAY, new ContextParameterValue(
35                                  "FRIDAY", 3));
36                  this.values.put(Calendar.SATURDAY, new ContextParameterValue(
37                                  "SATURDAY", 2));
38                  this.values.put(Calendar.SUNDAY, new ContextParameterValue(
39                                  "SUNDAY", 1));
40          }
41
42          @Override
43          public Set<ContextParameterValue> getAllPossibleValues() {
44                  return new HashSet<ContextParameterValue>(this.values.values());
45          }
46  }
47
```

4.5.3 Tryb pracy systemu

Analizowany parametr kontekstu przyjmuje postać:

Parametr kontekstu	tryb pracy systemu
Wartości parametru	praca normalna, prace serwisowe

Systemy charakteryzujące się dużą liczbą aktywnych użytkowników (w przypadku portalu Moja PG jest to liczba około 30 000) pracujące w trybie 24/7/365 praktycznie nie mogłyby podlegać zatrzymaniom. Jednak szczególnie w przypadku systemów podlegających intensywnemu rozwojowi potrzebne są okienka serwisowe na przeprowadzenie prac konserwacyjnych, np. wgranie nowszej wersji oprogramowania.

Opracowany mechanizm pozawala na zaplanowanie okienka serwisowego. Następnie na 10 minut przed planowanym wyłączeniem systemu wyświetla komunikat wszystkim zalogowanym użytkownikom o zbliżającej się przerwie i potrzebie zapisania efektów pracy. W momencie rozpoczęcia prac serwisowych wszyscy użytkownicy zostają wylogowani i niemożliwe jest ponowne zalogowanie do końca prac konserwacyjnych. Nie jest to jednak mechanizm całkowitego odcięcia użytkowników – uprawnieni użytkownicy mogą mimo

blokady zalogować się do systemu, aby wykonać prace serwisowe, czy zweryfikować poprawność uruchomienia systemu po wgraniu nowszej wersji oprogramowania.

Głównym celem wprowadzenia tego mechanizmu była troska o użytkownika, aby był lepiej poinformowany o zbliżającej się niedostępności i zabezpieczeniu przed utratą efektów pracy. Właśnie taka komunikacja wpłynęła na zwiększenie zaufania użytkowników do systemu, ponieważ z ich punktu widzenia podniosła się niezawodność pracy – system nie „ulega awariom", a „podlega planowanym przestojom", które nota bene odbywają się poza godzinami największego obciążenia.

Poniżej przedstawiono w postaci kodu Java implementację parametru kontekstu trybu pracy systemu. Klasa ta implementuje abstrakcyjną klasę *ContextParameter* przytaczaną w punkcie 4.3.

CPSystemState

```
1  package pl.lubomski.corbac.impl;
2
3  import java.util.HashSet;
4  import java.util.Hashtable;
5  import java.util.Set;
6  import pl.lubomski.corbac.ContextParameter;
7  import pl.lubomski.corbac.ContextParameterValue;
8  import pl.lubomski.corbac.Request;
9  import pl.lubomski.corbac.impl.SystemService.State;
10
11 public class CPSystemState extends ContextParameter {
12
13         private Hashtable<State, ContextParameterValue> values;
14
15         @Override
16         public ContextParameterValue computeCurrentValue(Request request) {
17                 State state = SystemService.getCurrentState();
18                 //
19                 return this.values.get(state);
20         }
21
22         @Override
23         public void fillAllPossibleValues() {
24                 this.values = new Hashtable<>();
25                 this.values.put(State.NORMAL, new ContextParameterValue(
26                                 "praca normalna", 3));
27                 this.values.put(State.MAINTENANCE, new ContextParameterValue(
28                                 "prace serwisowe", 1));
29         }
30
31         @Override
32         public Set<ContextParameterValue> getAllPossibleValues() {
33                 return new HashSet<ContextParameterValue>(this.values.values());
34         }
35 }
```

4.5.4 Weryfikacja adresu IP klienta API

Analizowany parametr kontekstu przyjmuje postać:

Parametr kontekstu	adres IP usługi/systemu klienckiego
Wartości parametru	prawidłowe wartości adresu IPv4

Uwierzytelnianie w API dostępowym do platformy IP^2, z którego korzystają systemu integrujące się z centralnym systemem, opiera się o weryfikację certyfikatu, jakim przedstawia się system kliencki [56][158]. Certyfikat taki w sekcji **X509v3 extensions** zawiera dodatkowe pola, np.:

```
X509v3 Subject Alternative Name:
                DNS:zak.eti.pg.gda.pl, IP
Address:153.19.55.230
```

W polach tych jest wskazany adres DNS oraz adres IP systemu klienckiego w stosunku do API. Certyfikaty te są podpisywane przez uczelniane CA (*ang. Certification Authority*).

Mechanizm bezpieczeństwa przyznaje uprawnienia do korzystania z API tylko wtedy, gdy adres IP klienta przychodzący w żądaniu (*ang. request*) jest identyczny z adresem IP zapisanym w certyfikacie. Dodatkowo weryfikowana jest nazwa DNS przypisana do danego adresu IP. W ten sposób niemożliwe jest wykorzystanie certyfikatu uwierzytelniającego przez inny system (na innym serwerze) niż zostało to określone w momencie wydawania dostępu. Mechanizm zabezpiecza dodatkowo przed zmianą charakteru usługi klienckiej (przynajmniej takiej, który zmienia nazwę DNS).

Algorytm implementacji tego parametru kontekstu jest bardzo podobny do parametru logicznej lokalizacji użytkownika. W tym przypadku również jest analizowany adres IP użytkownika. W pozostałych poniższych przykładach również algorytmy implementacji będą bardzo analogiczne.

4.5.5 Kontrola usług przy dostępie do źródła danych

Analizowany parametr kontekstu przyjmuje postać:

Parametr kontekstu	moduł logiki biznesowej
Wartości parametru	studentManager, personManager, dictManager, securityManager, …

Ze względu na rozproszenie poszczególnych elementów platformy na różne serwery fizyczne wymagane było wprowadzenie uwierzytelniania do poszczególnych jej elementów. Zagadnienie to dotyczy również dostępu do bazy danych SQL – dostęp możliwy jest po uprzednim uwierzytelnieniu i nadaniu uprawnień do poszczególnych tabel, sekwencji, itd. Ponieważ wolumen danych jest duży (aktualnie przeszło 100 GB w ponad 500 tabelach

bazodanowych) i pokrywa wiele obszarów funkcjonalnych, został on podzielony na osobne schematy bazodanowe odpowiadające obszarom funkcjonalnym. Stworzono również użytkowników bazodanowych odpowiadających poszczególnym modułom logiki biznesowej. Konfiguracja uprawnień do tabel bazodanowych została tak wprowadzona, żeby tylko jeden użytkownik (jeden moduł logiki biznesowej) mógł pisać do danej tabeli. Moduł ten jest traktowany jako „właściciel" danego obszaru danych. Ze względów wydajnościowych możliwy natomiast jest odczyt na poziomie bazy danych z tabel „należących" do innych obszarów.

Rozwiązanie takie zapewnia spójność danych wewnątrz obszarów. Jednocześnie, w systemie charakteryzującym się dużą liczbą operacji odczytu, nie wpływa negatywnie na wydajność, ponieważ możliwe jest realizowanie łączenia danych w jednym zapytaniu SQL z różnych obszarów.

4.5.6 Weryfikacja zgodności przeglądarki

Analizowany parametr kontekstu przyjmuje postać:

Parametr kontekstu	dostawca i wersja używanej przeglądarki
Wartości parametru	dostępne wersje przeglądarek internetowych

Jak już wcześniej wspomniano systemy internetowe charakteryzują się dużą przenośnością dzięki dostępowi przez przeglądarkę WWW. Jednocześnie intensywne wykorzystanie technologii AJAX i JavaScript wymaga dużej zgodności środowiska uruchomieniowego (więc przeglądarki) ze standardami HTML i XHTML. Z tym jednak starsze wersje przeglądarek mają poważne problemy. Skutkować może to słabszą wydajnością interfejsu użytkownika lub niedziałaniem pewnych funkcjonalności, które przeglądarka nie potrafi prawidłowo obsłużyć. Dodatkowo takie „archaiczne" przeglądarki posiadają liczne luki bezpieczeństwa usuwane dopiero w kolejnych wersjach.

Mając na celu zapewnienie pracy systemu charakteryzującej się niezawodnością na zakładanym poziomie (również niską awaryjnością i odpowiednią wydajnością interfejsu użytkownika) wprowadzono mechanizm weryfikacji używanej przez użytkownika przeglądarki. Mechanizm ten opiera się o analizę nagłówka HTTP **User-Agent**. Na podstawie przesłanego w żądaniu nagłówka wyznaczany jest dostawca i wersja użytej przeglądarki, a następnie porównywana z listą wersji dopuszczonych do użytku stworzonej na podstawie przeprowadzonych wcześniej testów. Użytkownik korzystający z przeglądarki niedopuszczonej do użytku otrzymuje stosowny komunikat z prośbą o skorzystanie z autoryzowanej wersji przeglądarki.

Analogicznie do mechanizmu weryfikującego tryb pracy systemu, zabezpieczenie to ma na celu zapewnienie zakładanego poziomu niezawodności działania systemu (sfera safety) –

w tym przypadku interfejsu użytkownika. Jednocześnie przekłada się to na wzrost zaufania użytkownika do systemu.

4.5.7 Zabezpieczenia przed atakami typu brute-force

Analizowane parametry kontekstu są następującej postaci:

Parametr kontekstu	scenariusz działań użytkownika
Wartości parametru	sekwencja akcji w systemie (profil użytkownika)

Parametr kontekstu	adres IP usługi/systemu klienckiego
Wartości parametru	prawidłowe wartości adresu IPv4

Parametr kontekstu	moment czasu wystąpienia żądania
Wartości parametru	prawidłowa data i godzina (z dokładnością do milisekund)

Wprowadzono dwa różne mechanizmy zabezpieczeń przed atakami siłowymi (*ang. brute-force*) mającymi na celu zgadnięcie danych uwierzytelniających. Jeden z mechanizmów wprowadzono w Centralnym Punkcie Logowania CPL PG. Każde żądanie uwierzytelniania podlega weryfikacji, czy z danego adresu IP w zadanym przedziale czasu od bieżącego momentu wstecz (ostatnie x minut) nie został przekroczony próg 3 prób uwierzytelniania zakończonych niepowodzeniem. Jeżeli wspomniany próg został przekroczony, to przerywana jest dalsza obsługa żądania. Dodatkowo zwiększany jest okres czasu blokady danego adresu IP. Na interfejsie użytkownika dodatkowo unieaktywniany jest przycisk logowania na czas trwania blokady.

Drugi z mechanizmów zastosowano w systemie eRekrutacja PG. W tym przypadku również zliczane są próby uwierzytelniania zakończone niepowodzeniem dla danego adresu IP klienta. W odróżnieniu od mechanizmu zastosowanego w CPL PG, w tym przypadku po 3 błędnych próbach z danego adresu IP użytkownik musi dodatkowo przepisać kod CAPTCHA [159] (w przypadku pierwszych 3 prób kod taki nie jest wyświetlany i nie jest wymagany). Dopiero pierwsze prawidłowe zalogowanie z danego adresu IP powoduje wyzerowanie licznika.

Rozdział 5. Ocena porównawcza platformy IP²

Pomiar bezpieczeństwa systemów informatycznych sprowadza się do obliczenia podstawowych wskaźników na podstawie zaproponowanego modelu w rozdziale 3 przy wykorzystaniu dostępnych technik pomiarowych. Głównym podejściem jest metoda audytów, która sprawdza kolejno wykorzystane mechanizmy bezpieczeństwa i porównuje je z dobrymi praktykami (*ang. best practices*), w tym także dobrze opracowanymi standardami. Przykładem takich norm są:

- brytyjska BS 7799-2
- międzynarodowa ISO/IEC 27001

Dodatkowo audyty mogą zawierać część techniczną, w skład której wchodzą testy penetracyjne. Bazują one na listach znanych typów podatności i próbują znaleźć je w testowanym systemie [96]. Innym podejściem jest wybranie jak najbardziej obiektywnych kryteriów ilościowych, na podstawie których będzie można wykazać różnice w bezpieczeństwie i przewagę poszczególnych rozwiązań – tzw. porównanie różnicowe.

Poniżej dokonano próby doboru takich kryteriów, oceny na ich podstawie tego samego systemu z bezkontekstową i kontekstową kontrolą bezpieczeństwa.

5.1 Ocena jakościowa poziomu bezpieczeństwa i użyteczności systemu

Na potrzeby analizy bezpieczeństwa i użyteczności rozpatrzono następujące kryteria:

- Przezroczystość dla użytkownika – mechanizmy bezpieczeństwa powinny wymuszać jak najmniej dodatkowych operacji wykonywanych przez użytkownika, które mają na celu podniesienie poziomu bezpieczeństwa.
- Zawężenie dostępnych operacji do niezbędnego minimum – jak już wcześniej sygnalizowano – użytkownik powinien mieć w danym momencie uprawnienia dokładnie tylko te, które są niezbędne do realizacji jego zadań. Każde dodatkowe w danym momencie „zbędne" uprawnienie jest niepotrzebnym punktem słabości w przypadku wykrycia podatności.
- Łatwość konfiguracji – im bardziej skomplikowana jest konfiguracja, tym bardziej podatna na błędy, które przekładają się na potencjalne luki w zabezpieczeniach.
- Prostota użycia – system powinien w jak największym stopniu automatyzować pracę administratorów, dzięki czemu unika się ludzkich pomyłek i uzyskuje wyższy poziom bezpieczeństwa.

- Dostępność systemu – z punktu widzenia użytkownika najwygodniej, gdy system w pełnej funkcjonalności dostępny jest z dowolnego miejsca w dowolnym czasie.

Biorąc pod uwagę powyższe kryteria można w sposób intuicyjny ocenić dwa proponowane rozwiązania w skali 1 do 5, gdzie 1 oznacza wartość najniższą, natomiast 5 – wartość najwyższą. Ponieważ jest to analiza porównawcza, jako wartość wyjściową dla każdego kryterium przyjęto wartość 3. Ocena porównawcza systemów z kontekstowymi i bezkontekstowymi mechanizmami bezpieczeństwa dla poszczególnych kryteriów przedstawia się następująco:

- Właściwie zaprojektowana warstwa bezpieczeństwa jest w takim samym stopniu przezroczysta zarówno dla systemu bezkontekstowego, jak i kontekstowego, przy czym w zastosowanie mechanizmów dostosowujących liczba dodatkowych weryfikacji (scenariusze bezpieczeństwa ss) na podstawie zaufania do użytkownika (*tl*) sprawia, że mechanizmy te są zdecydowanie mniej uciążliwe dla większości użytkowników (patrz rozdział 3.4). Dlatego system bezkontekstowy oceniono na wyjściowe 3 punkty, natomiast kontekstowemu przyznano 4 punkty.
- Zawężanie dostępnych operacji do niezbędnego minimum jest podstawowym założeniem mechanizmów bezpieczeństwa systemów kontekstowych (patrz Rysunek 21 w rozdziale 3.4), stąd system kontekstowy otrzymuje maksymalne 5 punktów. W przypadku systemu bezkontekstowego nie ma możliwości dynamicznego zawężania, więc ocena jest poniżej wyjściowej. Przyznano 2 punkty, ponieważ istnieje możliwość statycznego podziału dostępnych dla użytkownika operacji.
- Niewątpliwie złożoność konfiguracji mechanizmów systemów kontekstowych powoduje, że są one trudniejsze w konfiguracji (patrz rozdział 3.3). Warto jednak zauważyć, że zmiany w takiej konfiguracji są niezwykle rzadkie, wynikające przeważnie z rozwoju systemu, co nie zaniża tak bardzo oceny w tym przypadku. Dla systemy kontekstowego przyznano zatem 3 punkty. W przypadku systemu bezkontekstowego konfiguracja taka jest zdecydowanie prostsza ze względu na wykorzystanie intuicyjnego mechanizmu ról, jednak w przypadku bardzo dużych systemów wymaga również uwagi, stąd 4 punkty.
- Poświęcając czas na właściwą konfigurację mechanizmów kontekstowych zyskuje się pewną inteligencję systemu – na podstawie różnych przesłanek (kontekst) system będzie w stanie stosować różne scenariusze bezpieczeństwa (patrz rozdział 3.4), przez co prostota użycia wzrasta, stąd ocena ponad przeciętna wynosząca 4 punkty. Dla systemu bezkontekstowego, który nie wykazuje w tym zakresie usprawnień przyznano wyjściowe 3 punkty.
- Pełna dostępność systemu stoi w sprzeczności z założeniami ograniczania dostępu do krytycznych operacji (patrz Rysunek 21 w rozdziale 3.4). W takim rozumieniu dostępności system bezkontekstowy jest bardziej przyjazny użytkownikowi, stąd 4 punkty, natomiast system kontekstowy otrzymałby ocenę poniżej wyjściowej. Należy jednak zwrócić uwagę na inny aspekt dostępności w przypadku systemów

kontekstowych – oddzielając użytkowników korzystających z różnych strefy (sieci) i portali, separujemy ich od siebie, więc zbytnie obciążenie wygenerowane przez jednych nie blokuje dostępu dla tych zlokalizowanych w innej strefie. Fakt ten kompensuje ograniczenia wynikające z udostępniania niektórych funkcjonalności jedynie w określonych lokalizacjach. Wynikowo więc dla systemu kontekstowego przyznano 3 punkty.

Tabela 16 prezentuje zestawienie porównawcze oceny bezpieczeństwa systemów bez i z kontekstową kontrolą uprawnień.

Tabela 16 Zestawienie oceny bezpieczeństwa systemu z i bez kontekstowej kontroli uprawnień

Kryterium	System kontekstowy	System bezkontekstowy
przezroczystość dla użytkownika	4	3
zawężenie dostępnych operacji do niezbędnego minimum	5	2
łatwość konfiguracji	3	4
prostota użycia	4	3
dostępność systemu	3	4
Suma	**19**	**16**

Z powyższego porównania wynika, że oba rozwiązania są lepsze lub gorsze zależnie od analizowanego kryterium, jednak w całościowej ocenie przeważa system z zaimplementowanymi kontekstowymi mechanizmami bezpieczeństwa. Jest to zgodne z intuicyjnym podejściem – im więcej dodatkowych, ale poprawnych zabezpieczeń zostanie wprowadzonych, tym system będzie bezpieczniejszy. Jednak trzeba zadbać o to, żeby odbywało się to w sposób kontrolowany i nie komplikowało znacząco całości.

5.2 Audyt bezpieczeństwa platformy

W rozdziale 2.5.3 wskazano, że jednym z możliwych sposobów oceny bezpieczeństwa jest analiza ryzyka podatności (*ang. vulnerabilities*) występujących w systemie. Podatności można analizować w oparciu o jedną z dostępnych baz wymienionych w rozdziale 2.5 lub wyniki przeprowadzonego audytu bezpieczeństwa zawierającego testy penetracyjne.

Platforma IP^2 poddana została audytom bezpieczeństwa przeprowadzonym przez zewnętrzną firmę audytorską. Podczas każdego audytu sprawdzano ponad 1000 potencjalnych podatności. Otrzymane raporty podlegały analizie ryzyka wykrytych podatności. Poniżej zaprezentowano wyniki takiej oceny przeprowadzonej dla 12 najistotniejszych podatności zgłoszonych w pierwszym raporcie. Analiza została przeprowadzona zgodnie z wytycznymi *Common Vulnerability Scoring System Version 2.0* [111] z wykorzystaniem kalkulatora dostarczonego przez National Institute of Standards and

Technology [112]. Dla każdej wykrytej podatności określono wektor CVSS v2 oraz wyliczona jego wartość numeryczną zgodnie ze wzorem 2.1 przedstawionym w rozdziale 2.5.3.

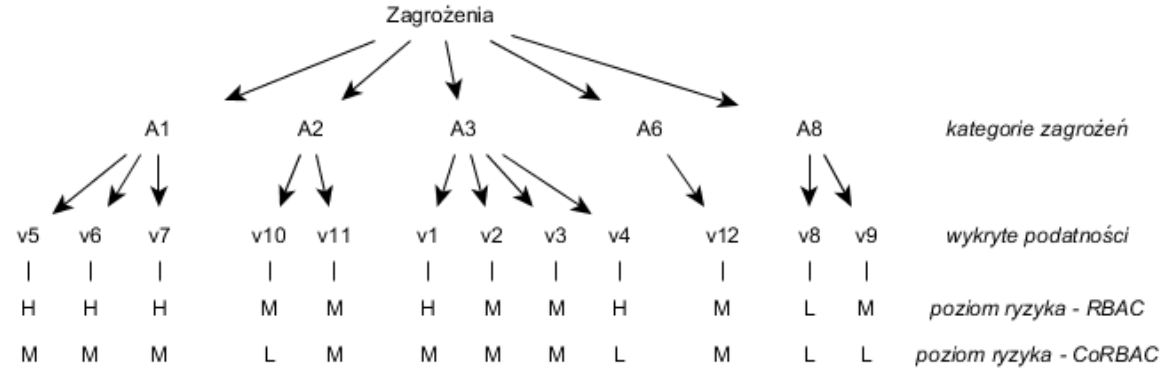
W poniższej analizie skupiono się tylko na pierwszej grupie metryk CVSS określających podstawowe cechy podatności niezmienne w czasie. Ma to na celu uniezależnienie analizy od uwarunkowań środowiskowych organizacji oraz zmienności w czasie życia podatności (większość została już usunięta poprzez wprowadzenie odpowiednich poprawek w systemie).

Tabela 17 przedstawia wyniki przeprowadzonej analizy ryzyka dla poszczególnych wykrytych podatności. Wskazano również jakie powinny być wyniki, w przypadku wyłączenia zaimplementowanych kontekstowych mechanizmów zabezpieczeń.

Tabela 17 Wyniki analizy ryzyka CVSS v2 dla wykrytych podatności

Id	Krótki opis	System z kontrolą RBAC		System z kontrolą CoRBAC	
		Wektor CVSS v2	Wynik	Wektor CVSS v2	Wynik
v1	Cross-Site Scripting – vulnerability 1	(AV:N/AC:M/Au:S/C:C/I:C/A:N)	7,9	(AV:N/AC:M/Au:S/C:P/I:P/A:N)	4,9
v2	Cross-Site Scripting – vulnerability 2	(AV:N/AC:L/Au:S/C:P/I:P/A:N)	5,5	(AV:A/AC:L/Au:S/C:P/I:P/A:N)	4,1
v3	Cross-Site Scripting – vulnerability 3	(AV:N/AC:L/Au:N/C:P/I:P/A:N)	6,4	(AV:N/AC:L/Au:N/C:N/I:P/A:N)	5,0
v4	Cross-Site Scripting – vulnerability 4	(AV:N/AC:M/Au:S/C:C/I:C/A:N)	7,9	(AV:A/AC:M/Au:S/C:P/I:P/A:N)	3,8
v5	SQL Injection – vulnerability 1	(AV:N/AC:H/Au:S/C:C/I:C/A:C)	7,1	(AV:A/AC:H/Au:S/C:C/I:C/A:C)	6,5
v6	SQL Injection – vulnerability 2	(AV:N/AC:H/Au:S/C:C/I:C/A:C)	7,1	(AV:A/AC:H/Au:S/C:C/I:C/A:C)	6,5
v7	SQL Injection – vulnerability 3	(AV:N/AC:H/Au:S/C:C/I:C/A:C)	7,1	(AV:A/AC:H/Au:S/C:C/I:C/A:C)	6,5
v8	CSRF – vulnerability 1	(AV:N/AC:M/Au:S/C:N/I:P/A:N)	3,5	(AV:N/AC:M/Au:S/C:N/I:P/A:N)	3,5
v9	CSRF – vulnerability 2	(AV:N/AC:M/Au:S/C:N/I:C/A:N)	6,3	(AV:A/AC:M/Au:S/C:N/I:P/A:N)	2,3
v10	session ID – vulnerability 1	(AV:N/AC:H/Au:S/C:C/I:C/A:N)	6,6	(AV:N/AC:H/Au:S/C:P/I:P/A:N)	3,6
v11	session ID – vulnerability 2	(AV:N/AC:H/Au:S/C:C/I:C/A:N)	6,6	(AV:A/AC:H/Au:S/C:C/I:C/A:N)	5,9
v12	Password reset	(AV:N/AC:L/Au:N/C:P/I:N/A:N)	5,0	(AV:N/AC:L/Au:N/C:P/I:N/A:N)	5,0

CVSS v2 definiuje trzy zakresy poziomu ryzyka podatności: niski (0 – 4,0), średni [4,0 – 7,0) oraz wysoki [7,0 – 10,0]. Zgodnie z tą interpretacją pokolorowano wyniki w tabeli – odpowiednio kolorami: zielonym, pomarańczowym i czerwonym. Rysunek 26 przedstawia topologię wykrytych podatności (w nawiązaniu do pkt. 2.5.2).



Rysunek 26 Topologia wykrytych podatności z określeniem ich ryzyka

Wyraźnie zauważalne jest obniżenie poziomu ryzyka wykrytych podatności dzięki zastosowaniu mechanizmów bezpieczeństwa wykorzystujących kontekst. Wszystkie podatności mieszczą się w przedziale średnim i niskim, natomiast dla systemu bezkontekstowego osiągają one zdecydowanie wyższe wartości.

Jest to powiązane z wrażliwością metryk CVSS v2 na dwa typy kontekstu: logiczna lokalizacja użytkownika (sieć, z której się łączy) oraz powiązanie z danymi (zakres danych, do których użytkownik ma dostęp i może modyfikować). Zasadnym byłoby rozszerzenie wrażliwości kryteriów oceny ryzyka na pozostałe typy kontekstu, a więc moment czasu wykonania operacji, bieżący stan systemu, historię działań użytkownika oraz cel tych działań. Pozwoliłoby to na dokładniejszą ocenę ryzyka związanego z wykrytą podatnością. Wymaga to modyfikacji zarówno metryk CVSS, jak i wzorów wyliczających wartość numeryczną ryzyka.

Zgodnie ze wzorem 2.5 w rozdziale 2.5.3 całościowy poziom zaufania do systemu λ wynosi odpowiednio: **0,9909** dla systemu z tradycyjną kontrolą bezpieczeństwa, oraz **0,9928** dla systemu z kontekstową kontrolą bezpieczeństwa. Widać wyraźnie, że system z kontekstowo zorientowaną kontrolą dostępu posiada wyższy poziom zaufania (λ). Na podstawie przytoczonych danych z rzeczywistego systemu wykazano, że kontekstowo zorientowana warstwa bezpieczeństwa wpływa pozytywnie, na podniesienie poziomu bezpieczeństwa całego systemu, **co potwierdza słuszność pierwszej tezy rozprawy.**

Powyższa analiza została przygotowana i przedstawiona do publikacji o międzynarodowym zasięgu [160].

5.3 Analiza profili użytkowników

W ramach prac doświadczalnych przeprowadzono analizę i budowę statystycznych profili działań użytkowników z uwzględnieniem poszczególnych parametrów kontekstu. Na bazie tak przygotowanych parametrów kontekstu można budować poszczególne kontekstowe

mechanizmy bezpieczeństwa. Jako baza badań posłużyły logi bezpieczeństwa platformy IP[2], które pierwotnie służyły jako narzędzie do ew. analiz powłamaniowych. Poddano analizie aktywność ponad 46 000 użytkowników na przekroju 4 lat.

Na wybór poszczególnych analizowanych parametrów kontekstu wpływ miały:

- dostępność poszczególnych danych, na podstawie których można przeprowadzić analizę,
- w miarę możliwości próba pokrycia wszystkich typów kontekstu celem wskazania najbardziej przydatnych w proponowanym modelu,
- weryfikacja przydatności korelacji danych kontekstowych z innymi danymi ewidencyjnymi zbieranymi w systemie.

W ten sposób wytypowano następujące parametry kontekstu:

- CP_1 – logiczną lokalizację użytkownika – zbiór składający się z następujących elementów: cp_{11} = sieć wewnętrzna, cp_{12} = sieć uczelniana, cp_{13} = sieć zewnętrzna (internet),
- CP_2 – dzień tygodnia, w którym użytkownicy pracowali z systemem – zbiór następujących elementów: cp_{21} = dzień powszedni, cp_{22} = sobota, cp_{23} = niedziela.

Zatem w analizowanym przypadku: $z = 2$, $|CP_1| = 3$, $|CP_2| = 3$, $|C| = 9$.

Przykładowy kontekst: $c_1 = (cp_{11}, cp_{21})$ = (sieć wewnętrzna, dzień powszedni).

Dokonana analiza polegała na wykryciu statystycznie nietypowego zachowania użytkownika. Przyjęto dwa założenia – dwa limity częstotliwości: 1% i 5% aktywności użytkownika w kontekście c. Poniżej pierwszego wspomnianego limitu aktywność jest bardzo podejrzana, więc powiązano ją z najniższym poziomem zaufania do użytkownika tl1. W związku z tym, zgodnie z modelem TCoRBAC (patrz punkt 3.4), użytkownik musi potwierdzić w zdecydowany sposób, że jest tym właściwym. Ten mechanizm bezpieczeństwa jest najbardziej niewygodny dla użytkownika (τ_T jest największe). Pomiędzy limitami 1% i 5% użytkownikowi jest przypisywany poziom zaufania tl2. Rozważono jeszcze jeden limit wynoszący 10%. Założono, że aktywności użytkownika wykonywane częściej niż raz na dziesięć razy w danym poziomie kontekście c nie są wystarczająco podejrzane ($\tau_T = 0$). W ten sposób przyjęto cztery poziomy zaufania do użytkownika tl1 do tl4, gdzie tl1 – najniższy, tl4 – najwyższy.

Przyjęto również pewne uproszczenie: podzielono interakcje użytkownika na mniejsze scenariusze odpowiadające sesjom od zalogowania do wylogowania. W ten sposób analizowano jedynie akcje logowania do systemu.

Analiza przeprowadzona została w dwóch aspektach: jak dużo użytkowników podlegało podejrzeniom oraz ile razy poziom zaufania (tl) był obniżany – innymi słowy jak często mechanizmy bezpieczeństwa musiały być uruchamiane.

Każdy profil użytkownika był wyznaczany w następujący sposób:

1. Dla każdego użytkownika u, dla każdego jego x-tego żądania dostępu została wyznaczana metryka częstotliwości określająca, jaki procent wcześniejszych żądań był wykonywany w określonym kontekście o wartości c_x (określonej lokalizacji i dniu tygodnia); brane pod uwagę było tylko 100 poprzedzających żądań danego użytkownika u:

$$\mathrm{freq}\left(\mathrm{access\_request}_{\mathrm{u,x,c_x}}\right) = \frac{\sum_{\mathrm{s=w}}^{\mathrm{x-1}} \mathrm{access\_request}_{\mathrm{u,s,c_x}}}{\sum_{\mathrm{s=w}}^{\mathrm{x-1}} \mathrm{access\_request}_{\mathrm{u,s}}} * 100\%, \quad (5.1)$$

   gdzie:

   $access\_request_{u,x,c_x}$ – x-te żądanie dostępu użytkownika u, mające miejsce w określonym kontekście o wartości c_x,

$$w = \begin{cases} x - 100, & x > 100 \\ 1, & x \leq 100 \end{cases}.$$

2. Jeżeli częstotliwość *freq* była mniejsza niż przyjęty limit, wtedy żądanie dostępu oznaczane było jako podejrzane i poziom zaufania do użytkownika (tl) był obniżany. W konsekwencji tego był uruchamiany bardziej restrykcyjny mechanizm bezpieczeństwa.
3. Pierwsze dziesięć żądań użytkownika nie zmieniało poziomu zaufania do użytkownika (tl).

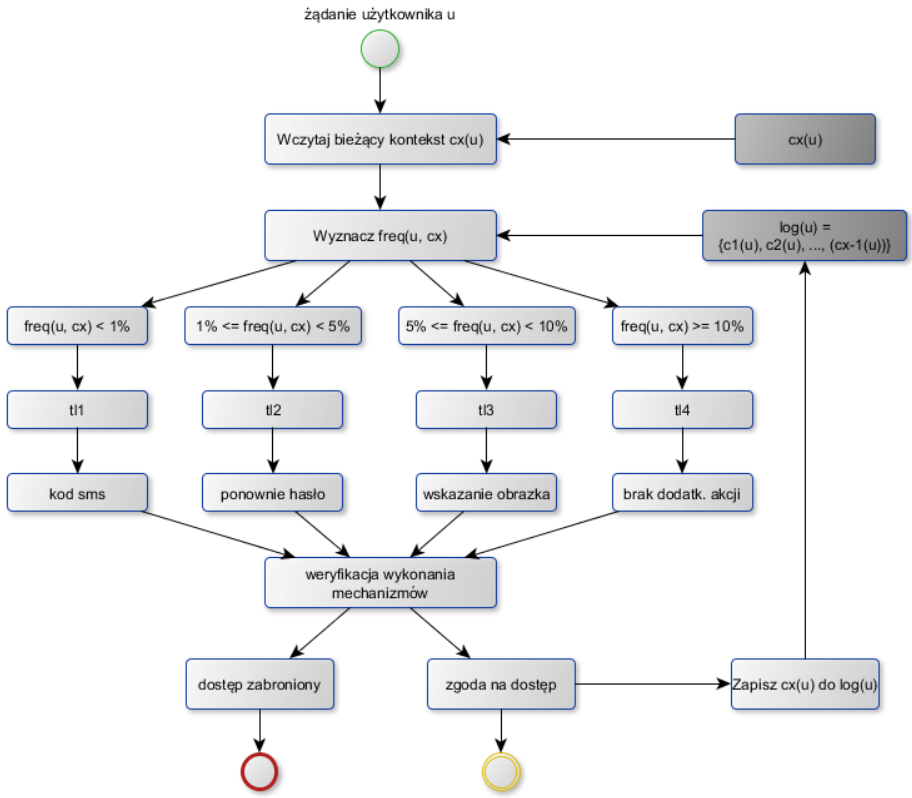
Dla lepszego zobrazowania przedstawiono ten proces na przykładzie zaprezentowanym w Tabela 18. Tabela ta prezentuje log żądań użytkownika u ze wskazaniem kontekstu każdego żądania (c_x). Dla uproszczenia przyjęto tylko jeden limit częstotliwości: poniżej 20%. Limit ten służy tylko i wyłącznie do zobrazowania procesu budowania profilu użytkownika. Sam profil jest konsekwentnie aktualizowany po każdym pozytywnie zrealizowanym żądaniu użytkownika.

Tabela 18 Przykładowa sekwencja żądań użytkownika u dla x od 1 do 13

x	c_x	freq (%)	limit „poniżej 20%”
1	c_1	0	tak (pominięte)
2	c_1	100	nie
3	c_1	100	nie
4	c_2	0	tak (pominięte)
5	c_1	75	nie
6	c_1	80	nie
7	c_1	83,3	nie
8	c_1	85,7	nie
9	c_1	87,5	nie
10	c_1	88,9	nie

11	c_2	10	tak
12	c_2	18,2	tak
13	c_2	25	nie

Przeanalizujmy kilka wierszy z przytoczonej tabeli. W pierwszym wierszu widać pierwsze żądanie użytkownika, któremu towarzyszy kontekst c_1. Nie było wcześniej żadnych żądań, więc częstotliwość wynosi 0% wcześniejszych żądań. To oczywiście przekracza limit „poniżej 20%", ale przypadek ten jest pominięty ze względu na regułę, że pierwsze 10 żądań nie zmienia poziomu zaufania. W wierszu 3 widać żądanie związane z kontekstem c_1 oraz częstotliwość wcześniejszych dwóch żądań związanych z kontekstem c_1 wynoszącą 100%. Przypadek ten nie przekracza limitu „poniżej 20%". W wierszu 4 ma miejsce pierwsze żądanie, któremu towarzyszy kontekst c_2. Częstotliwość żądań związanych z kontekstem c_2 wynosi 0% dla trzech wcześniejszych żądań użytkownika. To ponownie przekracza limit „poniżej 20%", ale również jest pomijane ze względu na regułę pierwszych dziesięciu żądań. Wiersz 11 przedstawia pierwszy interesujący przypadek. Jest to drugi raz, kiedy żądaniu użytkownika towarzyszy kontekst c_2. Żądania towarzyszące tej wartości kontekstu stanowią 10% wcześniejszych żądań użytkownika. To oczywiście przekracza limit „poniżej 20%" i powoduje uruchomienie bardziej restrykcyjnego scenariusza bezpieczeństwa. W wierszu 13 żądanie związane z kontekstem c_2 miało miejsce po raz czwarty. Żądania, którym towarzyszy kontekst c_2 stanowią 25% wcześniejszych dwunastu żądań, co nie powoduje już przekroczenia limitu „poniżej 20%".



Rysunek 27 Algorytm weryfikacji dostępu zgodnego z modelem TCoRBAC dla zadanych mechanizmów bezpieczeństwa

Rysunek 27 prezentuje algorytm weryfikacji dostępu zgodnego z modelem TCoRBAC dla zadanych mechanizmów bezpieczeństwa. Przeanalizowano log każdego użytkownika oddzielnie. Dla każdego żądania każdego użytkownika wskazano wartość bieżącego kontekstu, parametr częstotliwości ($freq_i$) oraz budowano i aktualizowano profil jego działań wskazując aktualny poziom zaufania (tl_i). Następnie zsumowano liczbę żądań mających miejsce w poszczególnych poziomach zaufania tl1 do tl4. Tabela 19 prezentuje wynik powyższej analizy. Liczby żądań użytkowników zostały pogrupowane do przedziałów (m, n], dla których użytkownicy zostali sklasyfikowani od m do n razy w poziomie zaufania tl_i. Dla każdego poziomu zaufania (tl_i) wskazano wartość liczbową i procentową użytkowników, którzy byli sklasyfikowani w danym poziomie zaufania.

Tabela 19 Liczba różnych użytkowników, którzy zostali sklasyfikowani od m do n razy w poziomie zaufania tl_i

Liczba żądań pojedynczego użytkownika	tl1		tl2		tl3		tl4	
	Liczba użytk.	% użytk.	Liczba użytk.	% użytk.	Liczba użytk.	% użytk.	Liczba użytk.	% użytk.
0	12265	26,64	15037	32,66	10727	23,30	4922	10,70
(0 – 5]	33164	72,04	19261	41,84	12085	26,25	1655	3,60
(5 – 10]	513	1,11	7296	15,85	6418	13,94	1463	3,18
(10 – 20]	72	0,16	3498	7,60	7824	17,00	2660	5,78
(20 – 30]	15	0,03	643	1,40	4111	8,93	2324	5,05
(30 – 40]	3	0,01	176	0,38	2207	4,79	2105	4,57
(40 – 50]	0	0,00	66	0,14	1155	2,51	1832	3,98
(50 – 100]	2	0,01	51	0,11	1376	2,99	7250	15,75
powyżej 100	0	0,00	6	0,01	131	0,28	21823	47,41

Podsumowując powyższą analizę około 70% użytkowników było rozważanych jako podejrzanych. Jednakże miało to miejsce w większości przypadków od 1 do 5 razy dla pojedynczego użytkownika, kiedy musiał on wykonać najbardziej restrykcyjny i uciążliwy scenariusz bezpieczeństwa (odpowiadający wejściu do stanu o najniższym poziomie zaufania do użytkownika tl1). Tak więc nie był on mocno uciążliwy dla niego w okresie 4 lat. Przeanalizowano również ekstremalne przypadki, gdy wymagana liczba pracochłonnych dla użytkownika scenariuszy bezpieczeństwa (ss) wynosiła powyżej 100. Byli to ludzie, którzy przez długi czas pracowali w jednym miejscu na pozycji administracyjnej wykorzystując sieć wewnętrzną, a następnie zmienili swoje stanowisko i miejsce pracy, co spowodowało zmianę sieci, z której korzystali.

Na podstawie powyższej analizy wykazano, że zastosowanie wzmocnionej kontroli bezpieczeństwa obejmującej aż ok. 70% użytkowników było prawie niewidoczne dla pojedynczego użytkownika – maksymalnie 10 razy na przestrzeni 4 lat. Można więc przyjąć, że bezpieczeństwo systemu (β) wzrosło, a użyteczność systemu (ε) z punktu widzenia użytkownika praktycznie nie zmalała.

Celem wykazania powyższego stwierdzenia dotyczącego użyteczności (ε) przeprowadzono jeszcze jedną analizę. Tabela 20 prezentuje dla wyżej analizowanego przypadku średnią liczbę żądań użytkowników związanych z poszczególnymi tl_i. Dodatkowo w wyniku normalizacji wyliczono częstotliwość tych żądań dla poszczególnych tl_i.

Tabela 20 Średnia liczba i częstotliwość żądań użytkowników w poszczególnych tl_i

	tl1	tl2	tl3	tl4
Średnia liczba żądań w poszczególnych tl_i	1,4745	3,9947	11,6607	153,7474
Częstotliwość żądań w poszczególnych tl_i	0,0086	0,0234	0,0682	0,8998

Zdefiniowano również przykładowy prosty scenariusz działań użytkownika w internetowym systemie usługowym. Tabela 21 prezentuje poszczególne żądania scenariusza wraz z czasem jego wykonania przez użytkownika. Dodatkowo wskazano kategorię tego czasu przypisując odpowiadające mu τ. Na podstawie własnych obserwacji pracy użytkowników oraz konsultacji z testerami zajmującymi się testami wydajnościowymi systemów, przyjęto, że średni czas interakcji po stronie użytkownika wynosi 1 sekunda w przypadku, gdy musi tylko kliknąć myszką oraz 3 sekundy, gdy musi wprowadzić jakieś dane z klawiatury, np. login i hasło.

Tabela 21 Przykładowy prosty scenariusz działań użytkownika w internetowym systemie usługowym

Nr żądania	Treść żądania	Zaangażowanie użytkownika (czas)	τ
1	Logowanie do systemu	3 sek.	τ_P
2	Wyliczenie bieżącego kontekstu c_x	≈ 0 sek.	τ_C
3	Wyliczenie uprawnień z uwzględnieniem c_x	≈ 0 sek.	τ_C
4	Weryfikacja uprawnień	≈ 0 sek.	τ_P
5	Wykonanie scenariusza bezpieczeństwa ss	0 / 1 / 3 / 5 sek.	τ_T
6	Wybranie aplikacji „student”	1 sek.	τ_B
7	Wybór przedmiotu	1 sek.	τ_B
8	Odczyt oceny	1 sek.	τ_B
9	Wylogowanie się	1 sek.	τ_P

Operacje po stronie systemu (np. weryfikacja uprawnień) z punktu widzenia użytkownika są pomijalne. Przyjęto cztery przykładowe scenariusze bezpieczeństwa ss wynikające z zaufania do użytkownika:

- tl4 – użytkownik w pełni zaufany – brak dodatkowych akcji (czas 0 sek.)
- tl3 – potwierdzenie tożsamości poprzez wskazanie 1 z 10 obrazków (czas 1 sek.)
- tl2 – ponowne wprowadzenie hasła lub innej danej osobowej (czas 3 sek.)
- tl1 – wprowadzenie sms-kodu (czas 5 sek.)

Na podstawie powyższego przykładowego scenariusza oraz wzoru 1.1 wyliczono $\varepsilon(ISS_{RBAC})$ uwzględniające tylko τ_B i τ_P, $\varepsilon(ISS_{CoRBAC})$ uwzględniające dodatkowo τ_C oraz $\varepsilon(ISS_{TCoRBAC})$, dla

którego uwzględniono dodatkowo τ_T, a całościowe czasy żądań dla poszczególnych tl_i przemnożono przez ich częstotliwość (patrz Tabela 20). Wyniki prezentują się następująco:

$$\varepsilon(ISS_{RBAC}) = \frac{\boldsymbol{\tau_B}}{\boldsymbol{\tau_B}+\boldsymbol{\tau_P}} = \frac{3}{3+4} \approx 0{,}4286$$

$$\varepsilon(ISS_{CoRBAC}) = \frac{\boldsymbol{\tau_B}}{\boldsymbol{\tau_B}+\boldsymbol{\tau_P}+\boldsymbol{\tau_C}} = \frac{3}{3+4+0} \approx 0{,}4286$$

$$\varepsilon(ISS_{TCoRBAC}) = \frac{\boldsymbol{\tau_B}}{\boldsymbol{\tau_B}+\boldsymbol{\tau_P}+\boldsymbol{\tau_C}+\boldsymbol{\tau_T}} = \frac{3}{3+4+0+0} \cdot 0{,}8998 + \frac{3}{3+4+0+1} \cdot 0{,}0682 + \frac{3}{3+4+0+3} \cdot 0{,}0234 + \frac{3}{3+4+0+5} \cdot 0{,}0086 \approx 0{,}4204$$

Użyteczność ε przyjmuje wartości od bliskich 0 do 1 (w przypadku braku kontroli bezpieczeństwa), a w praktyce do $\varepsilon(ISS_{RBAC})$. Wartość ε(ISS) rośnie i zbliża się do 1 wraz ze wzrostem stopnia złożoności scenariusza działań użytkowych, a więc wzrostem τ_B. Jednak przyjmując takie samo τ_B dla poszczególnych modeli otrzymujemy:

$\varepsilon(ISS_{RBAC}) \approx \varepsilon(ISS_{CoRBAC}) \approx \varepsilon(ISS_{TCoRBAC})$**.**

Ten wynik potwierdza słuszność drugiej tezy rozprawy.

Warto zanotować kilka uwag do powyższej analizy. Użytkownicy, którzy zdecydowanie zmieniali kontekst, byli poddawani bardzo restrykcyjnym scenariuszom bezpieczeństwa. Miało to miejsce stosunkowo rzadko dla tych użytkowników. Powodowało to, że mogli oni postrzegać system jako bardziej bezpieczny (w nawiązaniu do wyników pracy H. Crawford i K. Renaud [161]). Odbywało się to bez straty na ich ocenie dotyczącej wygody używania systemu.

Dodatkowo kontekstowe mechanizmy bezpieczeństwa nie są w stanie wychwycić sytuacji, gdy atakujący wpasuje się dokładnie w profil działania użytkownika (taki sam kontekst działania). Jest to jednak bardzo mało prawdopodobne, szczególnie gdy będzie analizowany bardziej złożony kontekst składający się z większej ilości parametrów kontekstu. Podejście takie jest wskazane również dlatego, że redukuje liczbę błędów klasyfikacji (ang. „false positive" i „false negative").

Zauważono również, że nie ma znaczącej różnicy w wynikach analizy przy zastosowaniu ramki 100 ostatnich żądań użytkownika do budowania profilu użytkownika oraz przypadku, gdy analizie podlegała cała wcześniejsza historia żądań użytkownika.

Powyższa analiza została zaprezentowana w publikacji o międzynarodowym zasięgu [143].

Rozdział 6. Uwagi i wnioski

W niniejszej rozprawie dokonano analizy usługowych systemów internetowych pod względem bezpieczeństwa. Przedstawiono autorskie uniwersalne modele bezpieczeństwa CoRBAC oraz jego rozszerzenie TCoRBAC, będące rozwinięciem tradycyjnego modelu RBAC. Uwzględniono analizę szeroko rozumianego kontekstu funkcjonowania systemu oraz poziom zaufania do użytkownika. Opracowano metodę wyznaczania dwóch parametrów: zaufania do systemu (λ) będącego tożsamym z poziomem bezpieczeństwa (β) oraz użyteczność systemu (ε). Analiza bezpieczeństwa została wykonana na podstawie analizy ryzyka podatności wykrytych przez audyt bezpieczeństwa, natomiast użyteczność na podstawie analizy logów działań użytkowników w systemie Moja PG rozwijanym i używanym na Politechnice Gdańskiej od ponad 4 lat.

W toku prac nad rozprawą zdefiniowano formalny model kontekstu oraz jego wpływ na model uprawnień (model CoRBAC). Kontekst został zdefiniowany jako zbiór wartości odpowiednich parametrów. Powoduje on dynamiczne zawężanie przydzielanych uprawnień w zależności od bieżącej sytuacji w systemie. Zaprezentowano również rozwinięcie tego modelu o algorytm kontroli dostępu uwzględniający zaufanie do użytkownika (model TCoRBAC). Przedstawiono oryginalne metody pozyskania i analizy bieżącego kontekstu. Zaproponowane modele w pewnym sensie odpowiadają modelom cytowanych w rozdziale 1.3. CoRBAC odróżnia się jednak od pozostałych tym, że wiąże kontekst nie z rolami, a z uprawnieniami, co pozwala na bardziej dokładną kontrolę bezpieczeństwa. Działa on na zasadzie „filtra", który zawęża zbiór uprawnień użytkownika z wykorzystaniem funkcji f_c i macierzy M_{PC}, w odróżnieniu od procedur aktywacji ról definiowanych w cytowanych pracach. Z kolei najbliższy modelowi TCoRBAC jest model X-GTRBAC [20], który sugeruje jedynie możliwość wprowadzenia dodatkowo zaufania do użytkownika podczas autoryzacji na podstawie analizy profilu jego działalności, ale nie podaje konkretnego rozwiązania. Tym samym cytowane prace nie uwzględniają również wpływu mechanizmów bezpieczeństwa na poziom bezpieczeństwa oraz na użyteczność systemu. Tak więc zaproponowane w niniejszej rozprawie modele wyróżniają się przede wszystkim całościowym, systematycznym ujęciem zagadnienia i wyjściem poza etap koncepcji. W konsekwencji pozwoliły na przedstawienie konkretnych rozwiązań oraz ich implementację w rzeczywistym systemie uczelnianym.

Uwzględnienie zaufania do użytkowników wymaga obserwacji ich zachowań w odpowiednim przedziale czasu przy sprecyzowaniu obserwowanych parametrów. Dotyczy to parametrów kontekstu związanych z użytkownikiem (takie jak lokalizacja, czas, wykorzystane urządzenie, powiązanie z danymi), a nie tych związanych z samym systemem (np. bieżące obciążenie, tryb pracy, itp.). Z analizy danych systemu Moja PG wynika, że można, przy uwzględnieniu charakterystycznych zmian zachowań użytkowników, ograniczyć liczbę poziomów zaufania

do czterech. W konsekwencji zdefiniowano cztery różne mechanizmy zabezpieczeń, które mogą być wykorzystane w zależności od poziomu zaufania systemu do użytkownika. Co więcej obserwując użytkownika można ograniczyć się do śledzenia nie więcej niż 100 ostatnich jego żądań. Zwiększenie liczby poziomów zaufania, ani śledzenie większej liczby żądań, nie zwiększa w sposób znaczący bezpieczeństwa systemu. Oczywiście wielkości te mogą być różne dla różnych typów systemów i kategorii użytkowników.

Na podstawie zaproponowanych modeli zaimplementowano reprezentacyjne mechanizmy bezpieczeństwa zgodne z przyjętą polityką bezpieczeństwa. Wykorzystano je w systemie Moja PG pracującym na platformie IP^2, który jest rozwijany i używany na Politechnice Gdańskiej. System ten obsługuje ponad 46 000 użytkowników, a wolumen przetwarzanych danych przekracza znacząco 100 GB. Warto przy tym podkreślić, że proponowane mechanizmy są implementowane w internetowym systemie o charakterze usługowym, co więcej są one realizowane jako nowe usługi nazywane usługami bezpieczeństwa. Tym samym w swojej strukturze i sposobie wywoływania nie różnią się od usług biznesowych. Takie jednorodne podejście ułatwia projektowanie systemu. Warte rozważenia jest też wprowadzenie specjalnych usług bezpieczeństwa (*ang. honeypot*) mających na celu tylko wykrycie prób nieautoryzowanego dostępu do systemu.

Przebadano właściwości proponowanych rozwiązań w oparciu o przyjęte modele, jak i też o wyniki audytów oraz dane eksperymentalne gromadzone podczas eksploatacji systemu Moja PG. Na tej podstawie wykazano zauważalny wzrost poziomu bezpieczeństwa (β) systemu wykorzystującego modele CoRBAC i TCoRBAC w porównaniu do systemu wykorzystującego model RBAC. Co więcej jest to możliwe bez znaczącego spadku użyteczności systemu (ε). To zdecydowanie potwierdza przewagę takiego rozwiązania w praktyce. Tym samym stanowi potwierdzenie słuszności przyjętych tez rozprawy.

Niniejsza rozprawa nie kończy definitywnie prac nad zagadnieniem kontekstowej kontroli bezpieczeństwa w internetowych systemach usługowych. Obszar ten, z uwagi na ważność problematyki, podlega intensywnym badaniom przez różne ośrodki naukowe oraz takie korporacje jak Google czy Facebook. Zapewne część osiągnięć trzymana jest w ścisłej tajemnicy.

Wraz ze wzrostem złożoności kontekstu wzrasta rozmiar analizowanych danych, co w konsekwencji wydłuża czas poświęcony na kontrolę bezpieczeństwa. Tym samym zmniejsza się użyteczność nadzorowanego systemu. Interesującym podejściem jest nie uwzględnianie pełnego kontekstu przy każdym żądaniu użytkownika, ale wybór w danym momencie czasu tylko jego jednego fragmentu (1-2 parametrów). W takim przypadku zgodnie z wykazanymi tezami akceptowalna użyteczność systemu pozostaje zachowana. Pojawia się jednak ciekawy problem, jak w czasie dobierać właściwe fragmenty kontekstu. Odpowiednia strategia wyboru może zapewnić również zwiększenie bezpieczeństwa systemu.

Rozprawę i proponowane w niej rozwiązania należy traktować jako istotny głos w dyskusji, co zostało podkreślone na międzynarodowym forum specjalistów bezpieczeństwa komputerowego [113]. Szczególnie obszar wpływu wprowadzanych zabezpieczeń na użyteczność jest bardzo trudny do przeanalizowania, ponieważ dotyczy przede wszystkim sfery miękkiej związanej z percepcją użytkownika. Dodatkowo wpływ na niego ma wiele czynników, w tym pomijane w niniejszej rozprawie bieżące obciążenie systemu jak i cała sfera zapewnienia ciągłości działania (rozprawa ogranicza się jedynie do wpływu kontroli dostępu na użyteczność). W tym kierunku prowadzone będą dalsze badania.

Spis rysunków

Spis tabel

Bibliografia

[1] P. P. Maglio, S. Srinivasan, J. T. Kreulen, and J. Spohrer, "Service systems, service scientists, SSME, and innovation," *Communications of the ACM*, vol. 49, no. 7, p. 81, Jul. 2006.

[2] W. Li, H. Wan, X. Ren, and S. Li, "A Refined RBAC Model for Cloud Computing," in *IEEE/ACIS International Conference on Computer and Information Science*, 2012, pp. 43–48.

[3] V. Chaurasiya, P. Dhyani, and S. Munot, "Linux Highly Available (HA) Fault-Tolerant Servers," *10th International Conference on Information Technology (ICIT 2007)*, 2007.

[4] T. Mather, S. Kumaraswamy, and S. Latif, *Cloud Security and Privacy. An Enterprise Perspective on Risks and Compliance*. O'Reilly, 2009.

[5] S. Fehr, "Flexible networks for better security," *Network Security*, vol. 2013, no. 3, pp. 17–20, Mar. 2013.

[6] G. Hogben, "A privacy enhancing identity management framework using the semantic web." nakł. aut., Gdańsk, 2009.

[7] C. K. Georgiadis, I. Mavridis, G. Pangalos, and R. K. Thomas, "Flexible team-based access control using contexts," in *Proceedings of the sixth ACM symposium on Access control models and technologies - SACMAT '01*, 2001, pp. 21–27.

[8] K.-D. Lee, M. Y. Nam, K.-Y. Chung, Y.-H. Lee, and U.-G. Kang, "Context and profile based cascade classifier for efficient people detection and safety care system," *Multimedia Tools and Applications*, vol. 63, no. 1, pp. 27–44, Apr. 2012.

[9] J. M. Stanton, K. R. Stam, P. Mastrangelo, and J. Jolton, "Analysis of end user security behaviors," *Computers & Security*, vol. 24, no. 2, pp. 124–133, Mar. 2005.

[10] S. P. S. Pahnila, M. S. M. Siponen, and A. M. A. Mahmood, "Employees' Behavior towards IS Security Policy Compliance," *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*, 2007.

[11] S. Furnell, "Usability versus complexity – striking the balance in end-user security," *Network Security*, vol. 2010, no. 12, pp. 13–17, Dec. 2010.

[12] W. H. DeLone and E. R. McLean, "Information Systems Success: The Quest for the Dependent

Variable," *Information Systems Research*, vol. 3, no. 1, pp. 60–95, Mar. 1992.

[13] J. H. Saltzer and M. D. Schroeder, "The protection of information in computer systems," *Proceedings of the IEEE*, vol. 63, no. 9, pp. 1278–1308, 1975.

[14] R. Sandhu, D. Ferraiolo, and R. Kuhn, "The NIST model for role-based access control," in *Proceedings of the fifth ACM workshop on Role-based access control - RBAC '00*, 2000, pp. 47–63.

[15] D. F. Ferraiolo, D. R. Kuhn, and R. Chandramouli, *Role-Based Access Control*, Second. Artech House, 2007.

[16] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," *Computer*, vol. 29, no. 2, pp. 38–47, 1996.

[17] M. Strembeck and G. Neumann, "An integrated approach to engineer and enforce context constraints in RBAC environments," *ACM Transactions on Information and System Security*, vol. 7, no. 3, pp. 392–427, Aug. 2004.

[18] X. Feng, X. Jun, H. Hao, and X. Li, "Context-Aware Role-Based Access Control Model for Web Services," in *Grid and Cooperative Computing - GCC 2004 Workshops SE - 54*, vol. 3252, H. Jin, Y. Pan, N. Xiao, and J. Sun, Eds. Springer Berlin Heidelberg, 2004, pp. 430–436.

[19] S. Haibo and H. Fan, "A context-aware role-based access control model for Web services," *IEEE International Conference on e-Business Engineering (ICEBE'05)*, pp. 220–223, 2005.

[20] R. Bhatti, E. Bertino, and A. Ghafoor, "A Trust-Based Context-Aware Access Control Model for Web-Services," *Distributed and Parallel Databases*, vol. 18, no. 1, pp. 83–105, Jul. 2005.

[21] J. W. Woo, M. J. Hwang, C. G. Lee, and H. Y. Youn, "Dynamic Role-Based Access Control with Trust-Satisfaction and Reputation for Multi-agent System," *2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops*, pp. 1121–1126, 2010.

[22] A. Gupta, M. S. Kirkpatrick, and E. Bertino, "A formal proximity model for RBAC systems," *Computers & Security*, Sep. 2013.

[23] X. H. Le, T. Doll, M. Barbosu, A. Luque, and D. Wang, "An enhancement of the Role-Based Access Control model to facilitate information access management in context of team collaboration and workflow," *Journal of Biomedical Informatics*, vol. 45, pp. 1084–1107, 2012.

[24] S. Gostojić, G. Sladić, B. Milosavljević, and Z. Konjović, "Context-Sensitive Access Control Model for Government Services," *Journal of Organizational Computing and Electronic*

Commerce, vol. 22, no. 2, pp. 184–213, Apr. 2012.

[25] P. Damián-Reyes, J. Favela, and J. Contreras-Castillo, "Uncertainty Management in Context-Aware Applications: Increasing Usability and User Trust," *Wireless Personal Communications*, vol. 56, no. 1, pp. 37–53, Dec. 2009.

[26] Politechnika Gdańska, "Moja PG," 2013. [Online]. Available: https://moja.pg.gda.pl.

[27] Oracle, "Java Platform, Enterprise Edition (Java EE)," 2013. [Online]. Available: http://www.oracle.com/technetwork/java/javaee/overview/index.html.

[28] M. Bishop, *Introduction to Computer Security*. Addison-Wesley Professional, 2004.

[29] S. Hernan, S. Lambert, T. Ostwald, and A. Shostack, "Uncover Security Design Flaws Using The STRIDE Approach," *Microsoft MSDN Magazine*, 2006.

[30] M. Anisetti, C. A. Ardagna, E. Damiani, and F. Saonara, "A test-based security certification scheme for web services," *ACM Transactions on the Web*, vol. 7, no. 2, pp. 1–41, May 2013.

[31] "The STRIDE Threat Model," *Commerce Server*, 2002. [Online]. Available: http://msdn.microsoft.com/en-us/library/ee823878(v=cs.20).aspx.

[32] G. Disterer, "ISO/IEC 27000, 27001 and 27002 for Information Security Management," *Journal of Information Security*, vol. 04, no. 02, pp. 92–100, 2013.

[33] ISO 27001, *Information Technology, Security Techniques, Information Security Management Systems, Requirements*. Geneve: International Organization for Standardization ISO, 2005.

[34] M. T. Siponen, "Secure-system design methods: evolution and future directions," *IT Professional*, vol. 8, no. 3, pp. 40–44, Jan. 2006.

[35] H. Krawczyk and P. Lubomski, "Generalized access control in hierarchical computer network," in *Zeszyty naukowe Wydziału Elektroniki, Telekomunikacji i Informatyki Politechniki Gdańskiej*, vol. 18, 2010, pp. 217–222.

[36] P. J. Windley, *Digital Identity*. O'Reilly, 2005.

[37] C. W. Thompson and D. R. Thompson, "Identity Management," *IEEE Internet Computing*, vol. 11, no. 3, pp. 82–85, May 2007.

[38] M. Benantar, *Access Control Systems. Security, Identity Management and Trust Models*. Springer-Verlag, 2006.

[39] T. Erl, *SOA Principles of Service Design*. SOA Systems Inc., 2007.

[40] W. Christopher, *Ajax. Bezpieczne aplikacje internetowe*. Helion Wydawnictwo, 2007.

[41] Google, "GoogleDocs," 2014. [Online]. Available: http://www.google.com/google-d-s/intl/pl/tour1.html.

[42] Microsoft, "Office 365," 2014. [Online]. Available: http://office.microsoft.com/pl-PL/.

[43] Adobe, "Adobe Photoshop," 2014. [Online]. Available: http://www.photoshop.com/tools.

[44] R. Zhang, F. Giunchiglia, B. Crispo, and L. Song, "Relation-Based Access Control: An Access Control Model for Context-Aware Computing Environment," *Wireless Personal Communications*, vol. 55, no. 1, pp. 5–17, Aug. 2009.

[45] N. Dimmock, A. Belokosztolszki, D. Eyers, J. Bacon, and K. Moody, "Using trust and risk in role-based access control policies," in *Proceedings of the ninth ACM symposium on Access control models and technologies - SACMAT '04*, 2004, p. 156.

[46] F. B. F. Shaikh and S. Haider, "Security threats in cloud computing," *2011 International Conference for Internet Technology and Secured Transactions*, pp. 214–219, 2011.

[47] K. J. Knapp, R. Franklin Morris, T. E. Marshall, and T. A. Byrd, "Information security policy: An organizational-level process model," *Computers & Security*, vol. 28. pp. 493–508, 2009.

[48] H. Krawczyk and J. Proficz, "Podstawowe metody integracji aplikacji trójwarstwowych," in *Od modelu do wdrożenia - kierunki badań i zastosowań inżynierii oprogramowania*, W. Dąbrowski and A. Stasiak, Eds. Wydawnictwo Komunikacji i Łączności, 2009, pp. 103–115.

[49] P. Lubomski, "Architektura zintegrowanego środowiska usług wspomagających funkcjonowanie uczelni," in *Perspektywy Rozwoju e-Uczelni w Kontekście Globalnej Informatyzacji; - e-uczelnia, konferencja krajowa.*, 2009, pp. 165–170.

[50] G. Goth, "Single Sign-on and Social Networks," *IEEE Distributed Systems Online*, vol. 9, no. 12, pp. 1–1, Dec. 2008.

[51] S. De Capitani Di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, G. Psaila, and P. Samarati, "Integrating trust management and access control in data-intensive Web applications," *ACM Transactions on the Web*, vol. 6, no. 2, pp. 1–43, May 2012.

[52] OpenID Foundation, "OpenID," 2013. [Online]. Available: http://openid.net/.

[53] OAuth Community, “OAuth,” 2013. [Online]. Available: http://oauth.net/.

[54] Jasig Community, “Central Authentication Service,” 2013. [Online]. Available: http://www.jasig.org/cas.

[55] Ministerstwo Administracji i Cyfryzacji, “ePUAP - elektroniczna Platforma Usług Administracji Publicznej,” 2013. [Online]. Available: http://epuap.gov.pl/.

[56] L. Harn and J. Ren, “Generalized Digital Certificate for User Authentication and Key Establishment for Secure Communications,” *IEEE Transactions on Wireless Communications*, vol. 10, no. 7, pp. 2372–2379, Jul. 2011.

[57] M. Myers and H. Tschofenig, “Online Certificate Status Protocol (OCSP) Extensions to IKEv2,” *IETF*, 2007. [Online]. Available: http://tools.ietf.org/html/rfc4806.

[58] B. W. Lampson, “Protection,” in *Proc. 5th Princeton Conf. on Information Sciences and Systems*, 1971, pp. 18–24.

[59] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman, “Protection in operating systems,” *Communications of the ACM*, vol. 19, no. 8, pp. 461–471, Aug. 1976.

[60] D. E. Denning, “A lattice model of secure information flow,” *Communications of the ACM*, vol. 19, no. 5, pp. 236–243, May 1976.

[61] R. S. Sandhu, “Lattice-based access control models,” *Computer*, vol. 26, no. 11, pp. 9–19, Nov. 1993.

[62] D. E. Bell, “Looking Back at the Bell-La Padula Model,” in *21st Annual Computer Security Applications Conference (ACSAC’05)*, pp. 337–351.

[63] K. Biba, “Integrity considerations for secure computer systems,” 1977.

[64] D. F. C. Brewer and M. J. Nash, “The Chinese Wall security policy,” *Proceedings. 1989 IEEE Symposium on Security and Privacy*, 1989.

[65] D. F. Ferraiolo and D. R. Kuhn, “Role-Based Access Controls,” in *15th National Computer Security Conference*, 1992, pp. 554–563.

[66] E. Bertino, “RBAC models — concepts and trends,” *Computers & Security*, vol. 22, no. 6, pp. 511–514, Sep. 2003.

[67] E. Bertino, L. Martino, F. Paci, and A. Squicciarini, *Security for Web Services and Service-Oriented Architectures*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[68] D. M'Raihi, M. Bellare, F. Hoornaert, D. Naccache, and O. Ranen, "HOTP: An HMAC-Based One-Time Password Algorithm," *IETF*, 2005. [Online]. Available: https://tools.ietf.org/html/rfc4226.

[69] A. Herzog and N. Shahmehri, "An evaluation of Java application containers according to security requirements," in *Proceedings of the Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE*, 2005, vol. 2005, pp. 178–183.

[70] A. Freier, P. Karlton, and P. Kocher, "The Secure Sockets Layer (SSL) Protocol Version 3.0," *IETF*, 2011. [Online]. Available: http://tools.ietf.org/html/rfc6101.

[71] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol. Version 1.2," *IETF*, 2008. [Online]. Available: http://tools.ietf.org/html/rfc5246.

[72] T. Imamura, B. Dillaway, and E. Simon, "XML Encryption Syntax and Processing," *W3C Recommendation*, 2002. [Online]. Available: http://www.w3.org/TR/xmlenc-core/.

[73] M. Bartel, J. Boyer, B. Fox, B. LaMacchia, and E. Simon, "XML Signature Syntax and Processing (Second Edition)," *W3C Recommendation*, 2008. [Online]. Available: http://www.w3.org/TR/xmldsig-core/.

[74] S. Cantor, J. Kemp, and E. Maler, "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0," *OASIS Security Services Technical Committee*, 2004. [Online]. Available: http://xml.coverpages.org/SSTC-SAMLCoreV20Draft17-7750.pdf.

[75] L. Sliman, F. Biennier, and Y. Badr, "A security policy framework for context-aware and user preferences in e-services," *Journal of Systems Architecture*, vol. 55, pp. 275–288, 2009.

[76] N. Nurseitov, M. Paulson, R. Reynolds, and C. Izurieta, "Comparison of JSON and XML Data Interchange Formats: A Case Study," *Scenario*, vol. 59715, pp. 157–162, 2009.

[77] B. Lin, Y. Chen, X. Chen, and Y. Yu, "Comparison between JSON and XML in Applications Based on AJAX," *2012 International Conference on Computer Science and Service System*, pp. 1174–1177, 2012.

[78] JSON-RPC Working Group, "JSON-RPC 2.0 Specification," 2013. [Online]. Available: http://www.jsonrpc.org/specification.

[79] C. Samsel, P. Heiniz, and K. Krempels, "Web Service to JSON-RPC Transformation," in *Proceedings of the 8th International Joint Conference on Software Technologies*, 2013, pp.

214–219.

[80] L. Richardson and S. Ruby, *RESTful Web Services*. 2008.

[81] S. Schreier, “Modeling RESTful applications,” *Proceedings of the Second International Workshop on RESTful Design WSREST 11*, pp. 15–21, 2011.

[82] S. Nakajima and T. Tamai, “Formal specification and analysis of JAAS framework,” *Proceedings of the 2006 international workshop on Software engineering for secure systems - SESS ’06*, p. 59, 2006.

[83] N. Leavitt, “Mobile Security: Finally a Serious Problem?,” *Computer*, vol. 44, no. 6, pp. 11–14, Jun. 2011.

[84] M. S. Kirkpatrick and E. Bertino, “Enforcing spatial constraints for mobile RBAC systems,” in *SACMAT ’10 Proceedings of the 15th ACM symposium on Access control models and technologies*, 2010, pp. 99–108.

[85] F. Hansen and V. Oleshchuk, “SRBAC: A spatial role-based access control model for mobile systems,” in *Proceedings of the 7th Nordic Workshop on Secure IT Systems (NORDSEC’03)*, 2003, pp. 129–141.

[86] C. Miller, “Mobile Attacks and Defense,” *IEEE Security & Privacy Magazine*, vol. 9, no. 4, pp. 68–70, Jul. 2011.

[87] A. Drozd, *Zabezpieczenie danych osobowych*. Presscom, 2008.

[88] P. Lubomski, “Wyzwania bezpieczeństwa nowoczesnych platform nauczania zdalnego,” *EduAkcja. Magazyn edukacji elektronicznej*, vol. 9, no. 1, pp. 80–89, 2015.

[89] Software Engineering Institute, “US-CERT Vulnerability Notes,” *Carnegie Mellon University*, 2014. [Online]. Available: https://www.kb.cert.org/vuls/.

[90] CVE Community, “Common Vulnerabilities and Exposures,” *The MITRE Corporation*, 2014. [Online]. Available: http://cve.mitre.org/.

[91] “National Vulnerability Database Version 2.2,” *National Institute of Standards and Technology*, 2014. [Online]. Available: https://nvd.nist.gov/.

[92] “SecurityFocus Vulnerability Database and BugTraq mail list,” *SecurityFocus*, 2014. [Online]. Available: http://www.securityfocus.com/vulnerabilities.

[93] “Open Sourced Vulnerability Database,” *Open Sourced Vulnerability Database (OSVDB)*, 2014. [Online]. Available: http://osvdb.org/.

[94] CWE Community, “Common Weakness Enumeration,” *The MITRE Corporation*, 2014. [Online]. Available: https://cwe.mitre.org/.

[95] “OWASP Top Ten Project,” 2015. [Online]. Available: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project.

[96] M. Meucci and A. Muller, *OWASP Testing Guide v4*. OWASP Foundation, 2014.

[97] L. Lowis and R. Accorsi, “On a Classification Approach for SOA Vulnerabilities,” in *2009 33rd Annual IEEE International Computer Software and Applications Conference*, 2009, pp. 439–444.

[98] M. Jensen, *Analysis of Attacks and Defenses in the Context of Web Services*. 2011.

[99] “Creative Commons Licenses,” 2015. [Online]. Available: http://creativecommons.org/licenses/.

[100] M. S. Lund, B. Solhaug, and K. Stølen, “Evolution in Relation to Risk and Trust Management,” *Computer*, vol. 43, no. 5, pp. 49–55, May 2010.

[101] J. Clinch, “ITIL v3 and information security,” *White Paper*, pp. 1 – 40, 2009.

[102] G. Hinson, “Seven myths about information security metrics,” *ISSA Journal*, 2006.

[103] J. R. Hauser and G. M. Katz, “Metrics: you are what you measure!,” *European Management Journal*, no. 4, pp. 517–528, 1998.

[104] S. C. Payne, *A Guide to Security Metrics*. SANS Security Essentials GSEC Practical Assignment, 2006.

[105] “Microsoft Security Response Center Security Bulletin Severity Rating System,” *Microsoft Developer Network*, 2002. [Online]. Available: http://msdn.microsoft.com/en-us/library/bb720758.aspx.

[106] “US-CERT Vulnerability Metric,” *National Institute of Standards and Technology*, 2014. [Online]. Available: www.kb.cert.org/vuls/html/fieldhelp#metric.

[107] J. D. Meier, A. Mackman, M. Dunner, S. Vasireddy, R. Escamilla, and A. Murukan, “Improving Web Application Security: Threats and Countermeasures,” *Microsoft patterns & practices*,

2015.

[108] P. Mell, K. Scarfone, and S. Romanosky, “Common Vulnerability Scoring System,” *IEEE Security and Privacy Magazine*, vol. 4, no. 6, pp. 85–89, Nov. 2006.

[109] “NVD Common Vulnerability Scoring System Support v2,” *National Institute of Standards and Technology*, 2007. [Online]. Available: http://nvd.nist.gov/cvss.cfm.

[110] P. Mell, K. A. Kent, and S. Romanosky, *The common vulnerability scoring system (CVSS) and its applicability to federal agency systems*. US Department of Commerce, National Institute of Standards and Technology, 2007.

[111] P. Mell, K. Scarfone, and S. Romanosky, “A Complete Guide to the Common Vulnerability Scoring System Version 2.0,” *FIRST.org, Inc.*, 2014. [Online]. Available: http://www.first.org/cvss/cvss-guide.

[112] “Common Vulnerability Scoring System Version 2 Calculator,” *National Institute of Standards and Technology*, 2014. [Online]. Available: https://nvd.nist.gov/cvss.cfm?calculator&version=2.

[113] P. Lubomski, “Context in Security of Distributed e-Service Environments,” in *Proceedings of the Chip to Cloud Security Forum 2014*.

[114] A. Ricci, M. Viroli, and A. Omicini, “An RBAC Approach for Securing Access Control in a MAS Coordination Infrastructure,” in *1st International Workshop “Safety and Security in MultiAgent Systems” (SASEMAS 2004)*, 2004, pp. 110–124.

[115] F. Cuppens and N. Cuppens-Boulahia, “Modeling contextual security policies,” *International Journal of Information Security*, vol. 7, no. 4, pp. 285–305, Nov. 2007.

[116] Z. Maamar, D. Benslimane, and N. C. Narendra, “What can context do for web services?,” *Communications of the ACM*, vol. 49, no. 12, pp. 98–103, Dec. 2006.

[117] R. Mayrhofer, H. R. Schmidtke, and S. Sigg, “Security and trust in context-aware applications,” *Personal and Ubiquitous Computing*, Nov. 2012.

[118] M. S. Kirkpatrick, M. L. Damiani, and E. Bertino, “Prox-RBAC: a proximity-based spatially aware RBAC,” in *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS ’11*, 2011, p. 339.

[119] M. L. Damiani, E. Bertino, B. Catania, and P. Perlasca, “GEO-RBAC,” *ACM Transactions on Information and System Security*, vol. 10, no. 1, p. 2–es, Feb. 2007.

[120] S. Aich, S. Sural, and A. K. Majumdar, “STARBAC: Spatiotemporal Role Based Access Control,” in *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS*, Springer Berlin Heidelberg, 2007, pp. 1567–1582.

[121] E. Bertino and M. S. Kirkpatrick, “Location-based access control systems for mobile users,” in *Proceedings of the 4th ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS - SPRINGL ’11*, 2011, p. 49.

[122] M. F. F. Khan and K. Sakamura, “Context-aware access control for clinical information systems,” in *2012 International Conference on Innovations in Information Technology (IIT)*, 2012, pp. 123–128.

[123] A. Baumgrass, “Deriving Current State RBAC Models from Event Logs,” *2011 Sixth International Conference on Availability, Reliability and Security*, pp. 667–672, 2011.

[124] M. Miettinen and N. Asokan, “Towards security policy decisions based on context profiling,” in *Proceedings of the 3rd ACM workshop on Artificial intelligence and security - AISec ’10*, 2010, p. 19.

[125] A. E. Abdallah and H. Takabi, “Integrating Delegation with the Formal Core RBAC Model,” *2008 The Fourth International Conference on Information Assurance and Security*, pp. 33–36, Sep. 2008.

[126] A. Gupta, M. Miettinen, N. Asokan, and M. Nagy, “Intuitive Security Policy Configuration in Mobile Devices Using Context Profiling,” in *Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Confernece on Social Computing (SocialCom)*, 2012, pp. 471–480.

[127] P. A. Zandbergen, “Accuracy of iPhone Locations: A Comparison of Assisted GPS, WiFi and Cellular Positioning,” *Transactions in GIS*, vol. 13, pp. 5–25, Jun. 2009.

[128] S. Cawood and M. Fiala, *Augmented Reality: A Practical Guide*. Pragmatic Bookshelf, 2008.

[129] Wikimedia Commons, “Wikitude - location-based Augmented Reality explained,” 2012. [Online]. Available: http://commons.wikimedia.org/wiki/File:Wikitude_explained.png.

[130] S. Meek, G. Priestnall, M. Sharples, and J. Goulding, “Mobile capture of remote points of interest using line of sight modelling,” *Computers & Geosciences*, vol. 52, pp. 334–344, Mar. 2013.

[131] E. Macías, H. Abdelfatah, A. Suárez, and A. Cánovas, “Full Geo-localized Mobile Video in Android Mobile Telephones,” *Network Protocols and Algorithms*, vol. 3, no. 1, Apr. 2011.

[132] M. Conti, V. T. N. Nguyen, and B. Crispo, “CRePE: Context-related Policy Enforcement for Android,” *ISC’10 Proceedings of the 13th international conference on Information security*, pp. 331–345, 2010.

[133] F. Paci, M. Mecella, M. Ouzzani, and E. Bertino, “ACConv -- An Access Control Model for Conversational Web Services,” *ACM Transactions on the Web*, vol. 5, no. 3, pp. 1–33, Jul. 2011.

[134] H. Mouratidis and J. Jurjens, “From goal-driven security requirements engineering to secure design,” *International Journal of Intelligent Systems*, vol. 25, no. 8, pp. 813–840, Jun. 2010.

[135] H. Mouratidis and P. Giorgini, “Integrating Security and Software Engineering: An Introduction,” in *Integrating Security and Software Engineering: Advances and Future Visions*, Hershey, PA, USA: Idea Group Publishing, 2006, pp. 1–15.

[136] E. Yu, L. Liu, and J. Mylopoulos, “A Social Ontology for Integrating Security and Software Engineering,” in *Integrating Security and Software Engineering: Advances and Future Visions*, Hershey, PA, USA, 2006, pp. 70–106.

[137] O. Etzion, Y. Magid, E. Rabinovich, I. Skarbovsky, and N. Zolotorevsky, “Context Aware Computing and its utilization in event-based systems,” *Context*, vol. 4, pp. 270–281, 2010.

[138] M. J. Covington, P. Fogla, Z. Z. Z. Zhan, and M. Ahamad, “A context-aware security architecture for emerging applications,” *18th Annual Computer Security Applications Conference, 2002. Proceedings.*, 2002.

[139] D. Kulkarni and A. Tripathi, “Context-aware role-based access control in pervasive computing systems,” in *Proceedings of the 13th ACM symposium on Access control models and technologies - SACMAT ’08*, 2008, p. 113.

[140] P. McDaniel, “On context in authorization policy,” *SACMAT*, pp. 80–89, 2003.

[141] S. Schefer-Wenzl and M. Strembeck, “Modeling Context-Aware RBAC Models for Business Processes in Ubiquitous Computing Environments,” in *2012 Third FTRA International Conference on Mobile, Ubiquitous, and Intelligent Computing*, 2012, pp. 126–131.

[142] V. Franqueira and R. Wieringa, “Role-Based Access Control in Retrospect,” *Computer*, vol. 45, no. 6, pp. 81–88, Jun. 2012.

[143] H. Krawczyk and P. Lubomski, “User Trust Levels and Their Impact on System Security and Usability,” in *Communications in Computer and Information Science*, Springer International Publishing, 2015, pp. 82–91.

[144] K. Scarfone and P. Mell, “Guide to Intrusion Detection and Prevention Systems (IDPS) Recommendations of the National Institute of Standards and Technology,” *NIST Special Publication*, p. 94, 2007.

[145] P. Pszczoliński and H. Krawczyk, “Ujednolicony opis zasobów uczelnianych,” in *Zeszyty Naukowe Wydziału ETI Politechniki Gdańskiej. Technologie Informacyjne*, Zeszyty Naukowe Wydziału ETI Politechniki Gdańskiej. Technologie Informacyjne, 2009, pp. 151–159.

[146] A. Rek and H. Krawczyk, “Wykorzystanie technologii portletów do budowy usług uczelnianych,” *Zeszyty Naukowe Wydziału ETI Politechniki Gdańskiej. Technologie Informacyjne*, vol. 17, no. 7, pp. 161–171, 2009.

[147] A. Rek and H. Krawczyk, “Methodology for developing Web-Based applications from reusable components using open source tools,” *Zeszyty Naukowe Wydziału ETI Politechniki Gdańskiej. Technologie Informacyjne*, vol. 18, no. 8, pp. 211–216, 2010.

[148] T. Dziubich, P. Lubomski, and A. Mizgier, “Architektura portalu zarządzania informacjami dydaktycznymi,” in *Zeszyty naukowe Wydziału Elektroniki, Telekomunikacji i Informatyki Politechniki Gdańskiej*, vol. 16, 2008, pp. 539–544.

[149] H. Krawczyk and P. Lubomski, “Pączkowanie – metoda rozwoju interoperacyjnych komponentów dla systemów rozproszonych,” in *Inżynieria oprogramowania w procesach integracji systemów informatycznych*, 2010, vol. 8, pp. 241–248.

[150] The PostgreSQL Global Development Group, “PostgreSQL,” 2013. [Online]. Available: http://www.postgresql.org/.

[151] M. Brambilla and A. Origgi, “MVC-Webflow: An AJAX Tool for Online Modeling of MVC-2 Web Applications,” in *2008 Eighth International Conference on Web Engineering*, 2008, pp. 344–349.

[152] P. Pszczoliński and H. Krawczyk, “Unified and flexible way to the organizations resources,” *Zeszyty Naukowe Wydziału ETI Politechniki Gdańskiej. Technologie Informacyjne*, vol. 19, no. 8, pp. 359–364, 2010.

[153] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, “Hypertext Transfer Protocol -- HTTP/1.1,” *IETF*, 1999. [Online]. Available: http://tools.ietf.org/html/rfc2616.

[154] M. Nottingham and J. Mogul, “HTTP Header Field Registrations,” *IETF*, 2005. [Online]. Available: http://tools.ietf.org/html/rfc4229.

[155] H. Zimmermann, “OSI Reference Model--The ISO Model of Architecture for Open Systems Interconnection,” *IEEE Transactions on Communications*, vol. 28, no. 4, pp. 425–432, Apr.

1980.

[156] Y. Liu and D. B. Hoang, “OSI RPC model and protocol,” *Computer Communications*, vol. 17, no. 1, pp. 53–66, Jan. 1994.

[157] Y. Li, D. Li, W. Cui, and R. Zhang, “Research based on OSI model,” in *2011 IEEE 3rd International Conference on Communication Software and Networks*, 2011, pp. 554–557.

[158] D. Khader, L. Chen, and J. H. Davenport, *Cryptography and Coding*, vol. 5921. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[159] E. Bursztein, M. Martin, and J. Mitchell, “Text-based CAPTCHA strengths and weaknesses,” in *Proceedings of the 18th ACM conference on Computer and communications security - CCS ’11*, 2011, vol. 2011, p. 125.

[160] P. Lubomski and H. Krawczyk, “Practical evaluation of security mechanisms of Internet systems (w recenzji),” *IEEE Security & Privacy Magazine*.

[161] H. Crawford and K. Renaud, “Understanding user perceptions of transparent authentication on a mobile device,” *Journal of Trust Management*, vol. 1, 2014.