Analizy Know-How - Analizy bezpieczeństwa Inteligentnych Kontraktów

Analizy bezpieczeństwa Inteligentnych Kontraktów będzie polegało na dostarczeniu kilku obszernych dokumentów w czasie trwania drugiego etapu. Prace będą polegały głównie na zebraniu wiedzy teoretycznej jak i analizie zaprojektowanych Inteligentnych Kontraktów.

Ethereum i skomplikowane programy na Blockchain są nowe i mocno eksperymentalne. Należy przygotować się na kluczowe zmiany w przestrzeni bezpieczeństwa. Nowe bugi i ryzyka bezpieczeństwa są odkrywane, w międzyczasie powstają nowe praktyki, które temu zapobiegają. W tym dokumencie są opisane praktyki, które można wykonać aby zapewnić należyte bezpieczeństwo działania inteligentnych kontraktów. Wykonywanie inteligentnych kontraktów wymaga innego podejścia programistycznego, a koszty związane z błędami mogą być wysokie, trudne czy niemożliwe w usunięciu.

A. Zebranie Know-How nt. zasad i sposobów bezpiecznego wykonywania Inteligentnych Kontraktów w celu uwolnienia funkcjonalności od potencjalnych błędów.

   Narzędzia do wizualizowania kontraktów:

1. Solidity Visual Auditor[1] - To rozszerzenie zapewnia składnię zorientowaną na bezpieczeństwo i wyróżnianie semantyczne, szczegółowy zarys klas i zaawansowane informacje o kodzie Solidity w programie Visual Studio Code.

[1] https://marketplace.visualstudio.com/items?itemName=tintinweb.solidity-visual-auditor

2. Sūrya[2] - Narzędzie użytkowe do inteligentnych systemów kontraktów, oferujące szereg efektów wizualnych i informacji o strukturze kontraktów. Obsługuje również zapytania do wykresu wywołań funkcji.
3. Solgraph[3] - Generuje wykres DOT, który wizualizuje przepływ kontroli funkcji kontraktu Solidity i wskazuje potencjalne luki w zabezpieczeniach.
4. EVMLab[4]
5. ethereum-graph-debugger[5] - Graficzny debugger EVM. Wyświetla cały wykres przepływu sterowania programem.
6. Piet[6] - aplikacja internetowa pomagająca zrozumieć inteligentne architektury umów. Oferuje graficzną reprezentację i kontrolę inteligentnych kontraktów, a także generator dokumentacji opisowej.

Statyczne i dynamiczne analizy:

1. MythX[7] - MythX to profesjonalna usługa w chmurze, która korzysta z analizy symbolicznej i fuzzingu wejściowego w celu wykrycia typowych błędów bezpieczeństwa i weryfikacji poprawności inteligentnego kodu umowy. Korzystanie z MythX wymaga klucza API z mythx.io.
2. Mythril[8] - szwajcarski scyzoryk zapewniający inteligentne bezpieczeństwo kontraktu.
3. Slither[9] - Framework analizy statycznej z detektorami dla wielu typowych problemów z Solidity. Ma możliwości śledzenia skażenia i wartości i jest napisany w języku Python.

---

[2] https://github.com/ConsenSys/surya
[3] https://github.com/fergarrui/ethereum-graph-debugger
[4] https://github.com/ethereum/evmlab
[5] https://github.com/fergarrui/ethereum-graph-debugger
[6] https://github.com/slockit/piet
[7] https://mythx.io
[8] https://github.com/ConsenSys/mythril
[9] https://github.com/trailofbits/slither

4. Contract-Library[10] - Dekompilator i narzędzie do analizy bezpieczeństwa dla wszystkich wdrożonych umów.
5. Echidna[11] - jedyny dostępny fuzzer dla oprogramowania Ethereum. Używa testowania właściwości do generowania złośliwych danych wejściowych, które łamią inteligentne umowy.
6. Manticore[12] - Narzędzie do dynamicznej analizy binarnej ze wsparciem EVM.
7. Oyente[13] - Analizowanie kodu Ethereum, aby znaleźć typowe luki w zabezpieczeniach.
8. Securify[14] - w pełni zautomatyzowany internetowy analizator statyczny dla inteligentnych umów, dostarczający raport bezpieczeństwa oparty na wzorcach podatności.
9. SmartCheck[15] - Analiza statyczna kodu źródłowego Solidity pod kątem luk w zabezpieczeniach i najlepszych praktyk.
10. Octopus[16] - narzędzie do analizy bezpieczeństwa inteligentnych kontraktów Blockchain ze wsparciem EVM i (e) WASM.
11. sFuzz[17] - Efektywny fuzzer zainspirowany przez AFL do wyszukiwania typowych luk.

[10] https://contract-library.com/
[11] https://github.com/trailofbits/echidna
[12] https://github.com/trailofbits/manticore
[13] https://github.com/melonproject/oyente
[14] https://securify.chainsecurity.com/
[15] https://tool.smartdec.net/
[16] https://github.com/quoscient/octopus
[17] https://sfuzz.github.io/

Kroki które należy zachować:

- Wyłączenie kontraktu, kiedy okaże się, że coś jest nie tak i wypracowanie efektywnej ścieżki łatania błędów, aktualizacji oprogramowania.
- Odpowiednie zarządzanie tokenami w ryzyku.
- Testowanie kontraktów cały czas zapobiegawczo, uruchomienie programu nagradzania od momentu kiedy oprogramowanie jest dostępne w sieci testowej zbliżonej do rzeczywistej.
- Utrzymywanie prostej logiki kontraktu, modularność kontraktów, utrzymanie metod kontraktów.
- Należy zachować jasność i wydajność kodu.
- Używanie blockchaina tylko w tych elementach oprogramowania, które wymagają decentralizacji.
- Sprawdzanie kontraktów pod kątem znalezienia nowych bugów, aktualizowanie oprogramowania i kompilatorów zawsze do najnowszych wersji. Ciągła analiza i stosowanie nowych technik zapobiegawczych.
- Być ostrożnym w odwoływaniu się do zewnętrznych kontraktów, które mogą zmienić przepływ informacji. Zrozumienie, że publiczne funkcje są publiczne i mogą być wywoływane przez dowolne oprogramowanie. W obecnej wersji Ethereum 1.0, nie utrzymywać prywatnych i wrażliwych danych w sieci Blockchain.
- Dbać o koszta wykonywania kontraktu, pamiętać, że uzyskanie liczby losowej jest trudne.
- Stosowanie wzorców plastycznych zwiększają złożoność i potencjalne możliwości ataków. Prostota jest szczególnie skuteczna w porównaniu ze złożonymi strukturami kontraktów.
- Utrzymywać proste, krótko żyjące kontrakty, zamiast monolitycznych skomplikowanych struktur.

Przykłady znanych ataków:

1. Reentrancy - odwoływanie się do zewnętrznych kontraktów, niesie ze sobą niebezpieczeństwo związane z przejęciem kontroli ruchu funduszy.
2. Front-Running - kiedy transakcje są widoczne w pamięci przed uruchomieniem transakcji, obserwatorzy mogą reagować zanim zostaną dodane do bloku transakcji. Wówczas atakujący może wyprzedzić akcję i odnieść zależne od systemu korzyści.
3. Zależność od Timestamp - Użycie timestampa bloku mogą być manipulowane przez osoby utrzymującą sieć. Wszystkie bezpośrednie i pośrednie użycia timestampa bloku powinny być przemyślane.
4. Ataki DoS - Atakujący może wieloma transakcjami zablokować uruchomienie konkretnej funkcjonalności przez dłuższy czas. Zablokować innym możliwość wykonania funkcji.

Dokładniejsze opisy ataków, inne znane ataki, przykłady użycia ataków, a także metody obrony przed atakami znajdziemy na stronie consensys.github.io[18].

B. Analizy bezpieczeństwa zaprojektowanych w pierwszym i drugim etapie Inteligentnych Kontraktów oraz ich interfejsów oraz Analizy bezpieczeństwa transakcji w ramach Inteligentnych Kontraktów.

Zalecenie: Za pomocą narzędzi z punktu A przeprowadzenie testów i analiz kodu kontraktów oraz wyprowadzenie wniosków. Poprawa ewentualnych błędów związanych z kontraktami. Należy wykonać przed wdrożeniem wyników w warunkach rzeczywistych.

[18] https://consensys.github.io/smart-contract-best-practices/known_attacks

C. Bezpieczne skonfigurowanie minimalistyczne (w sumie system powinien być bezpiecznym kontenerem, do którego ciężko się dobrać - taki jest cel). Bezpieczeństwo w zakresie kont użytkowników (konfiguracja, zalecenia logowania)

   Linki do kroków jakie należy zastosować:

   1. Privileged Accounts Guidelines[19]
   2. 10 Kroków do Cyberbezpieczeństwa[20]
   3. Zarządzanie bezpieczeństwem użytkowników[21]

D. Instrukcje sporządzenia bezpiecznego VPN (instrukcje - jaki system operacyjny (linux jaki), zalecenia co do narzędzi do zainstalowania, instrukcje co do konfiguracji tych narzędzi)

   Linki do kroków jakie należy zastosować:

   1. Jak skonfigurować OpenVPN[22]
   2. Rekomendacje do polepszenia bezpieczeństwa po instalacji[23]
   3. Czy OpenVPN jest bezpieczny do użytku w roku 2020[24]
   4. Zaawansowane ulepszenia bezpieczeństwa w OpenVPN[25]

---

[19] https://security.berkeley.edu/privileged-accounts-guidelines
[20] https://www.ncsc.gov.uk/collection/10-steps-to-cyber-security/the-10-steps/managing-user-privileges
[21] https://docs.oracle.com/cd/E37444_01/html/E37451/z40006791489694.html
[22] https://www.digitalocean.com/community/tutorials/how-to-set-up-an-openvpn-server-on-debian-10
[23] https://openvpn.net/vpn-server-resources/recommendations-to-improve-security-after-installation/
[24] https://www.vpnmentor.com/blog/what-is-openvpn-is-it-safe-enough-to-use
[25] https://openvpn.net/community-resources/hardening-openvpn-security/

E. Monitoring systemu i Alerty (Instrukcje). W tym szczegółowy sposób na monitoring konkretnych portów. Firewall / konfiguracja, która umożliwi blokowanie potencjalnych prób / zagrożeń.

1. Najlepsze praktyki i przewodnik do dostarczenia monitorowania serwerów [26]
2. Konfiguracja Firewall[27]

F. Podstawowe informacje na temat zabezpieczeń serwerów wirtualnych

Możliwości ataków z internetu dotykają praktycznie każdego urządzenia umieszczonego w serwerowni, podłączonego do sieci. Ataki na serwery mogą być wykonywane w różnym celu np. Kradzieży danych osobowych, wykorzystywania serwera do ataków w inne miejsca itp. Bezpieczeństwo serwerów wirtualnych wytworzonych za pomocą oprogramowania wirtualizacyjnego w chmurach może być zabezpieczone na kilka sposobów i na różnych warstwach od warstwy infrastruktury aż po oprogramowanie systemu operacyjnego.

Zalecenia:

- Odpowiednie skonfigurowanie systemu operacyjnego, zabezpieczeń użytkowników, dostępów.
- Utrzymywanie polityki bezpieczeństwa / kontroli dostępu do serwerów, polityki resetów i wymogów do haseł (zapewnia to między innymi poprawna konfiguracja oprogramowania serwerowego np. Linux czy Windows Server, jednakże dużym czynnikiem zagrażającym bezpieczeństwu jest czynnik ludzki i nie stosowanie się do polityk bezpieczeństwa.

[26] https://www.cloudradar.io/blog/guide-to-server-monitoring-best-practices
[27] https://phoenixnap.com/kb/iptables-tutorial-linux-firewall

- Odpowiednie zaprojektowanie sieci VPN, odpowiednie zabezpieczenie sieci VPN (szyfrowanie ruchu), korzystając z oprogramowania np. OpenVPN.
- Używanie dodatkowego oprogramowania zabezpieczającego takiego jak firewalle systemowe, oprogramowanie logujące zdarzenia, monitorujące sieć.
  (np. CSF[28])

Najlepszym sposobem zabezpieczenia całości będzie:

1. kopia całego VPSa zrobiona przez system wirtualizacyjny co 7 dni (wówczas jest zapewniona pełną konfiguracje i wszystkie pliki)
2. Codziennie lub częściej wykonana kopia wybranych plików systemu/bazy najlepiej na osobny serwer na macierz backup:

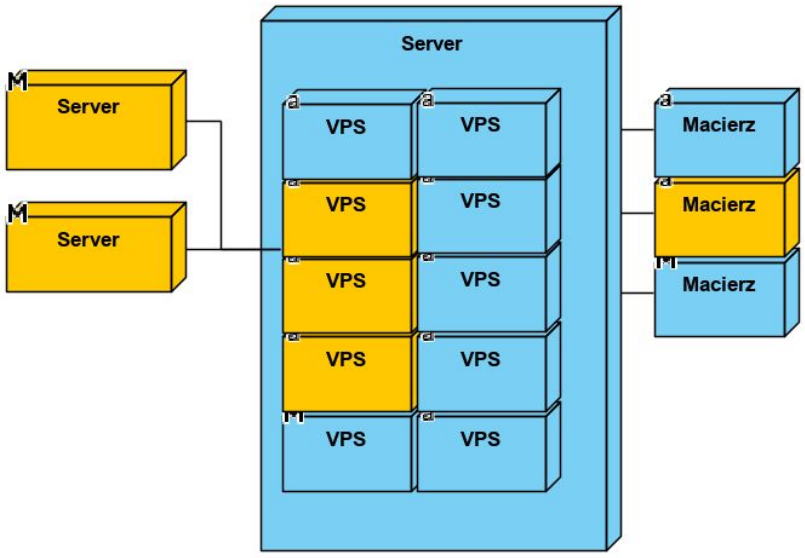
   a. zwykłe kopiowanie (mało wygodne)
   b. rsync (na zdalnego hosta np: rsync -azvP /etc/* username@remote_host:/backup/)
   c. Bacula[29]
   d. Duplicity[30] - szyfrowane backup'y
   e. BackupPC[31]

Zapewnienie sieci VPN

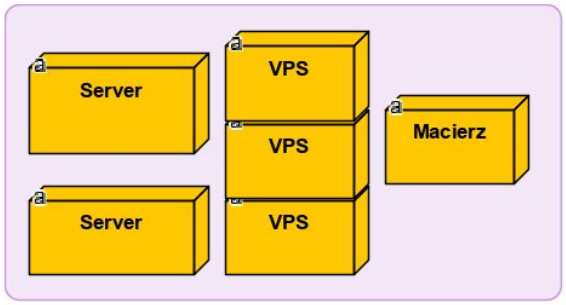
Wydzieleniem i odseparowaniem elementów infrastruktury jest zastosowanie sieci VPN, która skutecznie pozwoli odseparować komponenty platformy od pozostałych komponentów sieci w sposób bezpieczny. Dzięki zastosowaniu VPN -

[28] https://www.configserver.com/cp/csf.html
[29] http://blog.bacula.org
[30] http://duplicity.nongnu.org
[31] http://backuppc.sourceforge.net

zostaje wytworzona wirtualna sieć w której komponenty sieci zachowują się tak jakby znajdowały się w lokalnej sieci LAN, mają do siebie dostęp bezpośredni, działają w tej samej domenie / tej samej puli adresów IP, które zostały przydzielone przez sieć wirtualną. Ważne jest aby jeden z elementów - serwer wirtualnej sieci posiadał interfejs dostępu i publiczny adres IP, do którego można podłączyć domenę. Cała reszta sieci wraz z komputerami na stanowiskach pracowniczych, mogą działać w strukturze sieci VPN z kontrolowanymi dostępami do poszczególnych elementów sieci. Pozostałe elementy sieci nie mają dostępu do zasobów znajdujących się w wirtualnej sieci VPN, dlatego są bezpiecznie odseparowane. Poniższy schemat wizualizuje zachowanie sieci VPN.



Sieć VPN



Zapewnienie bezpieczeństwa danych z Baz Danych

Zapewnieniem bezpieczeństwa oraz integralności baz danych jest odpowiednie zaprojektowanie systemu backup samej bazy danych. W tym celu w zależności od typu bazy danych używa się wbudowanych mechanizmów samych systemów baz danych. Kopie 1:1 bazy danych mogą być realizowane raz na jakiś czas jednakże w przypadku zapewnienia kopii bazy danych zaleca się użycie streamingu danych oraz replikację baz danych.