ЛЕКЦИЯ 11
Веб-қосымшалардың шабуыл бетін және осалдықтарын анықтау әдістері

Лекция мақсаты

Веб-қосымшаның шабуыл беті мен серверлік кіру нүктелері ұғымдарын түсіндіру; клиенттік JavaScript-кодтың сервермен өзара әрекеттесуін қарастыру; динамикалық және статикалық талдау әдістерінің мүмкіндіктері мен шектеулерін салыстыру; анықталған кіру нүктелерін осалдықтарды іздеуде пайдалану қағидаларын мысалдар арқылы көрсету.

Веб-қосымшаның шабуыл беті

Веб-қосымшаның шабуыл беті — сыртқы субъект серверлік функцияларға әсер ете алатын кіру нүктелерінің жиынтығы. Веб-қосымшада мұндай әсер көбіне HTTP-сұраныстар арқылы жүзеге асады. Пайдаланушы браузердегі форма, батырма немесе басқа интерфейс элементімен әрекеттескенде клиенттік бөлік серверге сұраныс жібереді, ал сервер сол сұранысқа сәйкес белгілі бір операцияны орындайды.

Шабуыл бетін анықтаудың мақсаты — тек сайттың көрінетін беттерін тізімдеу емес, сервер орындай алатын операцияларға апаратын сұраныстарды мүмкіндігінше толық анықтау. Бір веб-бет бірнеше серверлік операцияны шақыра алады, ал кейбір операциялар интерфейсте мүлде көрінбеуі мүмкін.

Мысал. Тұрғындарға арналған электрондық қызмет порталында басты бет, жеке кабинет, өтініш беру және мәртебені тексеру функциялары бар делік. Пайдаланушы интерфейсінде төрт функция көрінгенімен, JavaScript қосымша профильді жаңарту, файл жүктеу және анықтамалықтарды алу үшін бірнеше API сұранысын жібереді. Демек, нақты шабуыл беті экранда көрінетін төрт функциядан кең болады.

Серверлік кіру нүктесі

Серверлік кіру нүктесі — белгілі бір серверлік операцияны шақыратын HTTP-сұраныстардың шаблоны. Оның бір бөлігі операцияны таңдауға қатысатын тұрақты бөлік болады; көбіне URL жолы мен HTTP әдісі осыған жатады. Екінші бөлігі өзгеретін параметрлерден тұрады: query-параметрлер, сұраныс денесі немесе кейбір тақырып мәндері.

Бір операцияға әртүрлі деректермен бірнеше сұраныс жіберілуі мүмкін. Мысалы, /user/login адресіне әртүрлі логин мен пароль жіберілгенімен, сервер бір login операциясын орындайды. Ал /site/search адресіне жіберілген сұраныс басқа серверлік операцияны шақырады.

Кіру нүктесін анықтағанда URL-ді ғана жазу жеткіліксіз. HTTP әдісі, параметр атаулары, олардың орналасуы және қандай рөл немесе сессия контекстінде қолданылатыны ескеріледі. Бұл ақпарат кейін параметрлерді қауіпсіздік тұрғысынан тексеруге негіз болады.

Мысал. Тұрғынның өтініш мәртебесін алу үшін GET /api/application/status?id=... сұранысы қолданылсын. Мұнда маршрут пен әдіс тұрақты, ал id — өзгеретін бөлік. Егер сервер id мәнін пайдаланушы құқығымен дұрыс байланыстырмаса, бұл кіру нүктесі авторизацияны тексеру үшін маңызды нысанға айналады.

Клиенттік бөліктен кіру нүктелерін анықтау

Black-box жағдайында серверлік код пен конфигурация қолжетімсіз болғандықтан серверлік кіру нүктелерінің толық жиынын дәл анықтау қиын. Сондықтан практикада клиенттік бөліктен серверге жіберілуі мүмкін сұраныстарды анықтап, шабуыл бетінің жуық жиыны құрылады.

HTML-ге негізделген қарапайым интерфейсте формалар мен сілтемелердің әрекеті айқын көрінеді. JavaScript қолданылатын динамикалық интерфейсте сұраныстың URL-і, әдісі және параметрлері бағдарламалық кодта қалыптасуы мүмкін. Single Page Application архитектурасында көптеген серверлік операциялар тек JavaScript арқылы шақырылады.

Мысал. Порталдың бастапқы HTML құжатында тек бір контейнер болуы мүмкін, ал барлық мәзір мен қызметтер JavaScript орындалғаннан кейін құрылады. Мұндай жағдайда тек HTML кодын

қарау API маршруттарының басым бөлігін көрсетпейді; клиенттік JavaScript пен нақты орындалатын сұраныстарды зерттеу қажет.

Динамикалық краулинг

Динамикалық краулинг — басқарылатын браузер арқылы веб-интерфейс элементтерімен автоматты әрекеттесіп, осы әрекеттер кезінде серверге жіберілген сұраныстарды тіркеу әдісі. Ол нақты браузер ортасында орындалатындықтан заманауи JavaScript, кітапханалар және күрделі интерфейс логикасымен жұмыс істей алады.

Негізгі артықшылығы — нақты орындалған сұраныстың URL-і, әдісі, параметрлері және runtime контексті тікелей бақыланады. Бірақ әдіс интерфейстегі барлық ықтимал әрекеттерді міндетті түрде орындай алмайды. Күрделі форма, көп қадамды сценарий, белгілі дерекке тәуелді батырма немесе тек арнайы рөлге көрінетін функция қамтылмай қалуы мүмкін.

Мысал. Тұрғындар порталында «өтінішті кері қайтару» батырмасы өтініш белгілі бір күйге жеткенде ғана пайда болады. Тест дерегінде мұндай күй болмаса, краулер батырманы көрмейді және тиісті серверлік сұранысты анықтамайды. Бұл динамикалық талдаудың қамтуы нақты орындалған сценарийлерге тәуелді екенін көрсетеді.

Динамикалық талдаудың шектеулері және қолжетімсіз код

Клиенттік кодта бар функцияның интерфейс арқылы орындалмауы мүмкін. Бұған авторизацияланған аймақтар, ескі немесе әлі толық енгізілмеген функциялар, әзірлеушілерге арналған мүмкіндіктер және белгілі дерек пайда болғанда ғана іске қосылатын интерфейс элементтері жатады.

Мұндай кодты елемеу дұрыс емес. Интерфейстен жол жоқ болғанымен, серверлік операция әлі де жұмыс істеуі мүмкін. Егер сервер жағында авторизация немесе басқа бақылау қате жүзеге асса, жасырын функция қауіпсіздік мәселесіне айналады.

Мысал. Порталдың бұрынғы нұсқасында тұрғынның өтінішін экспорттайтын функция болған, кейін батырма интерфейстен алынып тасталды делік. Бірақ ескі JavaScript файлы мен серверлік маршрут сақталған. Қарапайым краулинг бұл функцияны шақырмайды, ал кодты талдау оның серверге сұраныс жасайтынын көрсетуі мүмкін.

Статикалық JavaScript талдауы

Статикалық талдау бағдарламаны нақты орындамай оның кодын зерттейді. Веб-қосымша шабуыл бетін анықтау міндетінде негізгі мақсат — JavaScript ішінен серверге сұраныс жіберетін операцияларды және сол сұраныстардың ықтимал аргументтерін анықтау.

Статикалық тәсілдің маңызды артықшылығы — орындалуы қиын немесе мүлде қолжетімсіз кодты да зерттей алуы. Бірақ JavaScript динамикалық тіл: URL бірнеше айнымалыдан құралуы, функция жанама шақырылуы, объект өрісінің аты есептелуі және код сыртқы кітапханалармен байланысуы мүмкін. Сондықтан жалпы мақсаттағы статикалық анализаторлар нақты веб-қосымшаларда масштабталу және дәлдік мәселелеріне тап болады.

Мысал. Кодта baseUrl + '/resident/' + id + '/status' түріндегі жол қалыптассын. Толық URL кодта бір жол ретінде кездеспейді. Статикалық талдау жолдың тұрақты бөліктерін және өзгеретін id параметрін байланыстырып, серверлік кіру нүктесінің құрылымын қалпына келтіруге тырысады.

JavaScript-кодтың талдауды қиындататын ерекшеліктері

Қазіргі веб-қосымшалар модульдер, класстар, жанама функция шақырулары, есептелетін объект өрістері, жолдық операциялар және ірі кітапханаларды кең қолданады. URL мен параметрлердің мәндері бірнеше функция арқылы өтіп барып сұранысқа түсуі мүмкін. Бұл статикалық талдауда деректер ағынын қадағалауды күрделендіреді.

Жалпы мақсаттағы анализатор өте дәл модель құруға тырысса, үлкен кітапханаларда есептеу уақыты мен жад шығыны шамадан тыс өсуі мүмкін. Сондықтан нақты міндетке бағытталған мамандандырылған талдау кей жағдайда тиімді: мысалы, барлық JavaScript семантикасын

модельдеудің орнына серверге сұраныс құрастыруға қатысатын жолдық мәндер мен шақыруларды іздеу.

Мысал. jQuery немесе басқа кітапхана сұраныс жіберуді бірнеше абстракция қабаты арқылы орындай алады. Тестілеушіге кітапхананың барлық ішкі күйі емес, соңында қандай URL, әдіс және параметрлер қалыптасатыны маңызды.

Динамикалық және статикалық әдістерді біріктіру

Динамикалық және статикалық талдау бір-біріне балама емес, әртүрлі әлсіз жақтарды толықтыратын әдістер. Динамикалық краулинг нақты орындалған сұраныстарды жоғары сенімділікпен береді. Статикалық талдау интерфейсте орындалмаған немесе жетуі қиын кодтан қосымша кіру нүктелерін табуға мүмкіндік береді.

Гибридті тәсілде алдымен браузер арқылы нақты сұраныстар жиналып, кейін клиенттік кодтан табылған ықтимал сұраныстармен салыстырылады. Қайталанатын нүктелер біріктіріліп, тек статикалық әдіспен табылған нүктелер бөлек тексеріледі. Бұл шабуыл бетін қамтуды арттырады.

Мысал. Динамикалық краулер 45 серверлік операцияны анықтады, ал статикалық талдау тағы 10 ықтимал операция көрсетті делік. Оның алтауы бұрынғы сұраныстармен сәйкес келді, төртеуі жаңа болды. Осы төрт нүкте қолмен тексерілгенде екеуі шынында жұмыс істейтін жасырын API екені анықталды.

Кіру нүктелерінен осалдықтарды анықтауға өту

Шабуыл бетін анықтау — осалдықты автоматты түрде тапты деген сөз емес. Ол қауіпсіздік тесті қай нүктелерді қамтуы керектігін анықтайды. Әр кіру нүктесінде өзгеретін параметрлер, аутентификация, авторизация, деректерді тексеру және сервер жауабы қарастырылады.

Black-box тестінің сапасында толықтық маңызды: серверлік операция тестке мүлде кірмесе, сол операциядағы осалдықты табу мүмкін емес. Сондықтан «осалдық табылмады» деген қорытынды тест қамтуымен бірге түсіндірілуі тиіс.

Мысал. Порталда тұрғынның профилін көру, өзгерту және құжат жүктеу үшін үш бөлек API бар делік. Егер тест тек профильді көру API-сын қамтыса, файл жүктеу немесе өзгерту функциясындағы қауіпсіздік мәселелері зерттелмеген болып қалады.

Веб-қосымшаның шабуыл бетін анықтау әдістерін салыстыру

Әдіс	Қалай жүзеге асады	Артықшылығы	Негізгі шектеуі
Динамикалық краулинг	Браузерде интерфейспен әрекеттесіп, сұраныстарды тіркейді	Нақты runtime және заманауи кодпен жұмыс	Орындалмаған кодты өткізіп алады
Статикалық талдау	JavaScript-кодты орындамай зерттейді	Қолжетімсіз кодты қамти алады	Динамикалық тіл мен кітапханалар күрделі
Белгілі серверлік БҚ бойынша іздеу	Сигнатура және нұсқа бойынша белгілі маршруттарды болжайды	Стандартты БҚ үшін пайдалы	Арнайы жасалған қолданбада әлсіз
Белсенді сұраныстарды іздеу	Сервер жауаптарын талдап ықтимал маршруттарды тексереді	Интерфейсте жоқ нүктелерді табуы мүмкін	Параметрлер мен мәндер туралы толық ақпарат бермейді
Гибридті тәсіл	Динамикалық және статикалық нәтижелерді біріктіреді	Қамтуды арттырады	Нәтижені сәйкестендіру және растау қажет

Өзара байланыстар схемасы



Сурет 1. Клиенттік бөліктен серверлік шабуыл бетін анықтаудың жалпы логикасы

Схема веб-қосымшаның клиенттік бөлігіндегі әрекеттер мен JavaScript-код серверге HTTP-сұраныстарын қалыптастыратынын көрсетеді. Осы сұраныстар серверлік операцияларды

шақырады. Кіру нүктелерінің жиыны неғұрлым толық анықталса, осалдықтарды тексерудің қамтуы соғұрлым жоғары болады.

Практикалық кешенді мысал

Тұрғындарға мемлекеттік қызмет көрсететін веб-порталды қарастырайық. Порталда аутентификация, жеке кабинет, өтініш беру, құжат жүктеу және өтініш мәртебесін тексеру функциялары бар. Динамикалық краулинг кезінде қарапайым тұрғын рөлімен орындалатын сұраныстар тіркеледі. Нәтижесінде login, profile, application және status API нүктелері анықталады.

Кейін клиенттік JavaScript статикалық түрде зерттеледі. Кодта әкімшілік функцияға, бұрын қолданылған экспорт функциясына және тек белгілі күйдегі өтініш үшін іске қосылатын маршрутқа сұраныс қалыптастыратын бөліктер табылды делік. Бұл маршруттардың кейбірі динамикалық тестте шақырылмаған. Олар серверде шынымен жұмыс істей ме, қандай аутентификация мен авторизация талап етеді және қандай параметрлер қабылдайды деген сұрақтар бөлек тексеріледі.

Осылайша веб-қосымшаның шабуыл бетін анықтау интерфейсті жай шолудан тұрмайды. Клиенттік код, нақты HTTP-сұраныстар, пайдаланушы рөлдері және серверлік операциялар бір жүйе ретінде қарастырылады. Нәтижесінде осалдықтарды іздеу үшін неғұрлым толық және негізделген тест кеңістігі қалыптасады.

Қорытынды

Веб-қосымшалардың шабуыл бетін анықтауда негізгі объект — серверлік кіру нүктесі. Black-box жағдайда олардың толық жиыны тікелей белгісіз болғандықтан, клиенттік бөліктен серверге жіберілетін сұраныстар зерттеледі. Динамикалық краулинг нақты орындалған сұраныстарды сенімді анықтайды, бірақ орындалуы қиын немесе қолжетімсіз кодты өткізіп алуы мүмкін. Статикалық JavaScript талдауы осы шектеуді толықтыра алады, алайда динамикалық тілдің ерекшеліктері мен заманауи кітапханалар талдауды күрделендіреді. Сондықтан практикалық тұрғыдан әдістерді біріктіру шабуыл бетін қамтуды арттырады.

Бақылау сұрақтары

1. Веб-қосымшаның шабуыл беті дегеніміз не?
2. Серверлік кіру нүктесі қалай анықталады?
3. HTTP-сұраныстың тұрақты және өзгеретін бөліктерінің айырмашылығы қандай?
4. Black-box тестінде кіру нүктелерін толық анықтау неге қиын?
5. Динамикалық краулинг қалай жүзеге асады?
6. Динамикалық краулинг қандай кіру нүктелерін өткізіп алуы мүмкін?
7. Статикалық JavaScript талдауының мақсаты қандай?
8. JavaScript-тің қандай ерекшеліктері статикалық талдауды қиындатады?
9. Динамикалық және статикалық әдістерді біріктіру не береді?
10. Шабуыл бетін анықтау осалдықтарды тестілеудің толықтығына қалай әсер етеді?

Әдебиеттер тізімі:

1. Сигалов Д.А. Методы выявления поверхности атаки веб-приложений при помощи анализа клиентского JavaScript-кода: диссертация на соискание ученой степени кандидата технических наук. — Москва: МГУ имени М.В. Ломоносова, 2024.

2. Бегаев А.Н., Бегаев С.Н., Федотов В.А. Тестирование на проникновение: учебно-методическое пособие. — Санкт-Петербург : Университет ИТМО, 2018.