Revue des travaux récents sur l'usage des LLMs en cybersécurité offensive

Hamza Bendali[1], Oussema Ben Henni[1], Youssef Douzi[1], Abokor Mahmmad Mousse[1], Frédéric Flouvat [2], Safa Yahi[2]

[1]Université d'Aix-Marseille
[2] Université Aix-Marseille, LIS Lab

hamza.bendali@etu.univ-amu.fr, youssef.douzi@etu.univ-amu.fr, abokor.mahammad-mousse@etu.univ-amu.fr,frederic.flouvat@lis-lab.fr,safa.yahi@lis-lab.fr

Résumé

La cybersécurité s'impose aujourd'hui comme un enjeu majeur, alors que les entreprises et les organisations font face à une augmentation constante des attaques informatiques susceptibles d'affecter leur fonctionnement et leurs infrastructures. Dans ce contexte, le hacking éthique constitue un pilier essentiel de la cybersécurité, reposant sur le principe « connaître les attaques pour mieux se défendre ».
Avec l'essor des LLM (Large Language Models), les pratiques des professionnels, des formateurs et des apprenants en cybersécurité — et en hacking éthique en particulier — évoluent rapidement.
Cet article propose une revue systématique de la littérature récente portant sur l'usage des LLM dans le domaine du hacking éthique. Les travaux analysés permettent d'identifier les principales applications de ces modèles (automatisation du pentest, génération de code d'exploitation ou assistance à l'analyse de vulnérabilités), mais aussi leurs limites actuelles, notamment en matière de fiabilité, d'hallucinations techniques et de risques liés au dual-use.

Mots-clés

LLM, Hacking Éthique, Cybersécurité, Pentest

1 Introduction

La cybersécurité constitue aujourd'hui un enjeu stratégique pour les organisations publiques et privées. En effet, la transformation numérique massive et l'interconnexion croissante des infrastructures informatiques ont considérablement augmenté la surface d'attaques des systèmes d'information, des attaques qui peuvent avoir de lourdes conséquences.
La cybersécurité est généralement structurée autour de deux approches complémentaires : l'approche défensive et l'approche offensive. La sécurité défensive vise à protéger les infrastructures informatiques, détecter les intrusions et répondre aux incidents de sécurité. En revanche, l'approche offensive, appelée aussi **hacking éthique,** consiste à mener des attaques afin d'identifier les vulnérabilités d'un système avant qu'elles ne soient exploitées par des acteurs malveillants. Le test d'intrusion (**penetration testing** ou pentest en abrégé) constitue l'une des pratiques les plus répandues dans ce domaine. Il repose sur une méthodologie structurée comprenant plusieurs phases successives telles que la reconnaissance, l'analyse des vulnérabilités, l'exploitation, l'escalade de privilèges et la post-exploitation. Ceci repose largement sur l'expertise humaine des hackers éthiques, qui doivent mobiliser à la fois des connaissances technique, une capacité d'analyse et une grande créativité dans l'exploration des vecteurs d'attaque.
Par ailleurs, ces dernières années ont été marquées par l'essor rapide des LLM (pour Large Language Models) qui ont démontré des progrès remarquables dans de nombreux domaines tels que la génération de code.
En cybersécurité, les LLM suscitent un intérêt croisassant aussi bien sur le plan déffensif qu'offensif. Dans cet article, nous nous focalisons sur le volet offensif. Plus précisément, nous proposons une revue systématique des travaux récents portant sur l'usage des LLM dans le domaine du hacking éthique. Nous analysons et comparons leurs apports et leurs limites ainsi que les solutions sont proposées à ces limites.
Le reste de cet article est structuré comme suit. Après la description de la méthodologie, nous présenterons dans la section 2 des travaux d'utilisation de LLM en pentest d'une manière générale et dans le cadre du Web en particulier, en ingénieurie sociale et aussi dans l'analyse des malwares. La section 3 décrit des techniques de spécialisation de LLM dans le cadre du hacking éthique. Dans la section 4, nous présenterons des benchmarks d'évaluation et de comparaison. Dans la section 5, nous décrivons les risques et enjeux éthiques liés à l'utilisation des LLM en Cybersécurité. Nous donnerons une conclusion dans la section 6.

1.1 Méthodologie

Pour cette revue systématique, nous suivons une approche claire et structurée. La démarche consiste à réaliser une recherche documentaire, puis à sélectionner les études selon des critères d'inclusion et d'exclusion définis à l'avance. Les informations pertinentes sont ensuite extraites et comparées sur plusieurs aspects : fiabilité des modèles, fréquence et impact des hallucinations, efficacité des ap-

proches de spéciaisation de LLM par rapport aux méthodes classiques.

1.2 Directives de reporting

La revue est rédigée selon la méthode PRISMA 2020 afin d'assurer clarté et transparence dans la recherche, la sélection des études, l'extraction des données et la synthèse.

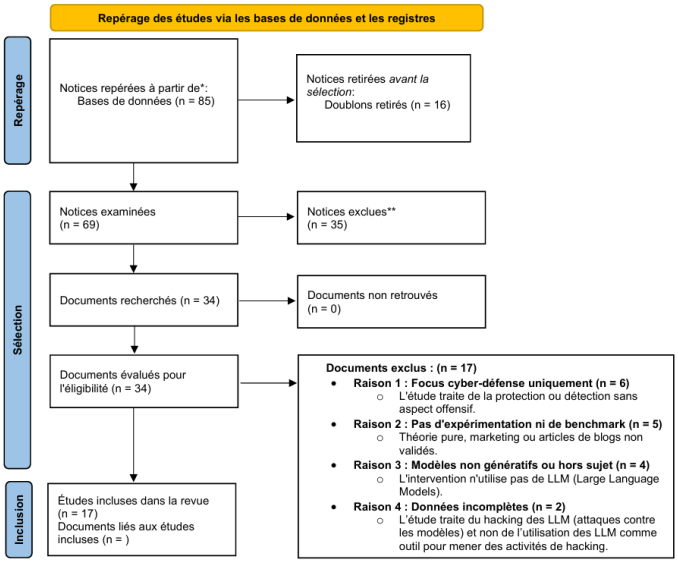

FIGURE 1 – Diagramme PRISMA

2 Les LLM comme outils offensifs

2.1 Automatisation du pentest

L'automatisation du pentest constitue l'un des axes les plus explorés, plusieurs frameworks ont été développés pour évaluer dans quelle mesure les LLM peuvent conduire des tests d'intrusion de manière autonome ou semi-autonome.

2.1.1 PentestGPT

Liu et al. [1] proposent PentestGPT, évalué sur 13 cibles HackTheBox et VulnHub (plateformes proposant des machines volontairement vulnérables pour s'entraîner légalement au pentest) avec 182 sous-tâches couvrant différents OS, niveaux de difficulté (facile, moyen, difficile) et types de vulnérabilités, validées par trois pentesters certifiés OSCP (certification professionnelle reconnue dans le domaine des tests d'intrusion). Les cinq modèles (GPT-3.5, GPT-4, Bard, PentestGPT-GPT-3.5, PentestGPT-GPT-4) sont chacun testés cinq fois par cible via une boucle interactive où le LLM recommande des actions, un humain les exécute et renvoie les résultats jusqu'à résolution ou impasse. GPT-4 se révèle le plus performant parmi les modèles natifs, avec 4 cibles faciles et 1 cible moyenne résolues (52/77 sous-tâches faciles, 27/71 moyennes, 8/34 difficiles), loin devant Bard (2 faciles) et GPT-3.5 (1 facile). Aucun modèle ne parvient toutefois à résoudre les cibles difficiles. L'intégration de PentestGPT améliore considérablement ces résultats : PentestGPT-GPT-4 résout 6/7 cibles faciles et 2/4 cibles moyennes, soit deux fois plus de sous-tâches moyennes que GPT-4 seul (57 contre 27, +111%) et une progression de 58,6% du taux de complétion global. Sur HackTheBox, il parvient à compléter 4 cibles sur 10 pour un coût de 131,5 USD, atteignant le top 1% mondial parmi 670 000 membres.

2.1.2 AutoAttacker

Xu et al. [2] présentent AutoAttacker, un outil automatisant les tests de pénétration post-accès initial (phase qui se déroule une fois qu'un attaquant a déjà pénétré un système et cherche à étendre son emprise), reposant sur quatre composants : un summarizer pour condenser l'historique des interactions, un planner pour définir les étapes d'attaque, un navigator pour l'exécution, et un experience manager inspiré du RAG pour mémoriser et réutiliser les sous-tâches réussies. L'évaluation porte sur 14 attaques couvrant différentes étapes de la matrice MITRE ATT&CK (base de connaissances publique qui répertorie et classe les techniques utilisées par les attaquants à chaque phase d'une intrusion) sur des environnements Windows et Linux virtualisés. Les modèles GPT-3.5, GPT-4, Llama2-7B et Llama2-70B sont testés avec une technique de jailbreak par jeu de rôle pour contourner les filtres de sécurité. La température a été variée entre 0, 0.5 et 1.0 afin d'évaluer l'impact du degré d'aléatoire sur les décisions du modèle : une température faible favorise des réponses stables et reproductibles, tandis qu'une température élevée encourage des comportements plus exploratoires. GPT-4 domine largement avec 100% de réussite (température à 0) en moins de 15 rounds. GPT-3.5 affiche des performances limitées, réussissant seulement 3 tâches basiques à cause d'une mauvaise gestion du contexte et de commandes syntaxiquement incorrectes. Les modèles open-source Llama 2 échouent complètement par manque de connaissances sur des outils comme Metasploit (framework d'exploitation de vulnérabilités très répandu en pentest). L'experience manager s'avère efficace : sur une attaque ransomware (logiciel malveillant qui chiffre les fichiers de la victime et réclame une rançon pour les déchiffrer) par exemple, il passe de 17 à 7 interactions tout en réduisant significativement les coûts d'API.

2.1.3 hackingBuddyGPT

Happe et al. [3] présentent hackingBuddyGPT, un outil autonome basé sur un LLM conçu pour réaliser des tests d'escalade de privilèges (technique par laquelle un attaquant disposant d'un accès limité cherche à obtenir des droits plus élevés, typiquement les droits administrateur) sous Linux. L'évaluation repose sur un benchmark de 12 machines virtuelles, chacune exposant une vulnérabilité distincte (binaires SUID mal configurés, abus de droits sudo, échappements de conteneurs Docker, fuites d'informations sensibles, tâches cron). Quatre modèles sont évalués (GPT-3.5-Turbo, GPT-4-Turbo, Llama3 8b et 70b) : le LLM génère des commandes via SSH jusqu'à obtenir les droits root ou atteindre 60 tentatives. GPT-4-Turbo obtient les meilleurs résultats, avec un taux de réussite progressant de 33% sans assistance à 66% avec la compaction d'état, et jusqu'à 83% avec des indices de haut niveau. À titre de comparaison, un

expert humain atteint 75% sans aide et 91% avec indices, tandis que les outils automatisés classiques plafonnent à 8-16%. GPT-4-Turbo avec gestion d'état revient à seulement 1,54 USD par vulnérabilité exploitée, contre environ 159 USD/heure pour un pentest humain.

2.1.4 HackSynth

Muzsai et al. [4] présentent HackSynth, un agent de pentest autonome avec deux modules : un Planner qui génère des commandes Bash et un Summarizer qui analyse les résultats pour suivre la progression. Deux benchmarks de 200 challenges issus de PicoCTF et OverTheWire (plateformes de type CTF, où les participants doivent résoudre des défis de sécurité pour trouver un flag caché, dans un cadre légal) couvrent six catégories (Web, Crypto, Forensics...) et trois niveaux de difficulté. L'agent tourne en autonomie totale dans un environnement Kali Linux conteneurisé et isolé. Huit modèles (GPT-4o, Llama-3.1, Qwen2...) ont été testés sur 20 itérations par challenge. GPT-4o domine avec 34,2% sur PicoCTF et 40% sur OverTheWire. Côté open-source, Llama-3.1-70B s'en sort le mieux (22,5% et 28,8%), surpassant même GPT-4o-mini sur OverTheWire. Les bons modèles savent changer de stratégie après un échec, là où les plus faibles répètent les mêmes tentatives en boucle. Une température de 1,0 et une fenêtre d'observation limitée (250-500 caractères) donnent les meilleurs résultats sans saturer le contexte du Planner.

2.1.5 PentestAgent

Shen et al. [5] présentent PentestAgent, un framework de pentest automatisé avec une architecture multi-agents (Reconnaissance, Recherche, Planification, Exécution). Pour compenser les limites des LLM (connaissances datées, mémoire courte), le système intègre du RAG pour maintenir une base de connaissances à jour, ainsi que du Chain-of-Thought et de l'auto-réflexion pour le débogage. Le benchmark couvre 67 cibles VulHub (32 catégories CWE (liste standardisée répertoriant les types de faiblesses logicielles pouvant mener à des vulnérabilités)) et 11 challenges HackTheBox sur trois niveaux de difficulté. GPT-4 s'impose avec 74,2% de réussite globale contre 60,6% pour GPT-3.5. Face à PentestGPT, PentestAgent est bien plus autonome et rapide : 80% de complétion en collecte d'informations sur VulHub (contre 10% pour PentestGPT), et la reconnaissance sur HackTheBox se fait en 220s contre 1199s pour PentestGPT à cause de l'intervention humaine.

2.2 Spear phishing et ingénierie sociale

L'utilisation des LLM pour le spear phishing constitue un autre domaine d'application offensif majeur.

Hazell [6] teste la capacité de GPT-3.5 et GPT-4 à assister les phases de reconnaissance et de génération de messages. Pour démontrer le passage à l'échelle (scaling), l'expérimentation a consisté à générer plus de 600 messages d'hameçonnage personnalisés destinés aux membres du Parlement britannique. L'étude démontre un coût opérationnel dérisoire : un email d'attaque coûte moins d'un centime d'euro en ressources de calcul (API). Il est crucial de noter que cette étude se concentre sur le coût financier direct pour l'attaquant et non sur le coût énergétique colossal lié à l'infrastructure des serveurs de l'IA. L'étude révèle par ailleurs qu'il n'est pas nécessaire d'être un expert en programmation pour détourner ces outils : une simple ingénierie de prompt basique — technique consistant à reformuler une demande pour contourner les filtres éthiques — suffit. Par exemple, au lieu de demander directement un email malveillant, l'utilisateur demande au modèle de « simuler un scénario pédagogique de sécurité ». Ce simple changement de contexte permet de contourner les safeguards (barrières de sécurité) intégrées. Hazell explore deux solutions potentielles : le contrôle de l'accès via des API structurées et l'utilisation des LLM eux-mêmes comme systèmes de défense.

La revue de Xu et al. [7] confirme et élargit ces observations. Contrairement aux campagnes massives classiques, les LLM génèrent des messages hautement personnalisés en analysant des données spécifiques à la victime (noms, contexte professionnel). Cette automatisation permet de passer d'une production artisanale à une échelle industrielle, tout en maintenant un niveau de persuasion élevé à un coût financier quasi nul.

2.3 Exploitation de vulnérabilités web

Les applications web constituent aujourd'hui l'une des surfaces d'attaque les plus étudiées en cybersécurité. De nombreuses vulnérabilités exploitables dans ce contexte, telles que les injections SQL, les failles d'authentification ou les erreurs de configuration, nécessitent une analyse progressive de l'application cible. Les LLM sont particulièrement intéressants dans ce cadre car ils possèdent des capacités de raisonnement permettant d'interpréter des messages d'erreur, d'adapter des requêtes et de planifier des attaques en plusieurs étapes.

Fang et al. [8] étudient la capacité des LLM à agir comme des **agents autonomes** (systèmes capables d'enchaîner seuls les étapes d'une tâche sans intervention humaine directe après leur lancement) capables d'exploiter des vulnérabilités web sans aucune assistance humaine. L'agent développé combine un modèle de langage avec un navigateur automatisé et plusieurs outils d'interaction HTTP. Il fonctionne selon une **boucle en 3 étapes : observation–raisonnement–action** (architecture où le modèle observe l'état du système, génère un raisonnement, exécute une action puis analyse le résultat pour adapter la stratégie) : il analyse la page web ou la réponse serveur, génère une hypothèse d'attaque, produit un **payload** (fragment de code malveillant destiné à exécuter l'action offensive, par exemple une **injection SQL** pour manipuler une base de données), exécute la requête puis analyse le résultat afin d'adapter la stratégie suivante. L'agent reçoit uniquement un objectif général — par exemple accéder à des données protégées dans une base de données — sans indication explicite sur la vulnérabilité présente.

Cette architecture illustre une évolution importante dans l'utilisation des LLM en cybersécurité. Plutôt que d'utiliser les modèles uniquement comme assistants capables de générer du code ou d'expliquer des vulnérabilités, ces ap-

proches cherchent à transformer les modèles en agents capables d'interagir directement avec un environnement informatique. Dans ce contexte, le modèle devient un composant central du processus d'attaque automatisée.

Les résultats montrent que les agents basés sur GPT-4 sont capables d'identifier et d'exploiter des vulnérabilités web de manière autonome, en particulier dans des scénarios d'injection SQL nécessitant plusieurs étapes d'exploration. Le modèle parvient par exemple à tester différentes requêtes, analyser les messages d'erreur retournés par le serveur et adapter progressivement ses payloads afin d'extraire des informations depuis la base de données cible. Les LLM peuvent également planifier des **attaques multi-étapes** (ou **multi-turn**, construites sur plusieurs échanges successifs où chaque étape exploite les réponses précédentes) : identifier un formulaire vulnérable, tester différentes variations d'injection puis récupérer le schéma de la base avant de procéder à l'**exfiltration de données** (processus par lequel un attaquant récupère et transfère des données sensibles depuis un système compromis).

À l'inverse, les modèles moins performants comme GPT-3.5 ou certains modèles **open-source** (modèles dont les poids sont publiquement accessibles) tels que LLaMA-2 et LLaMA-3 échouent beaucoup plus fréquemment à maintenir une stratégie cohérente sur plusieurs itérations. L'attaque automatisée reste relativement peu coûteuse en termes d'utilisation d'API (environ quelques dollars par tentative complète). Ces résultats montrent que les performances des agents autonomes dépendent fortement de la capacité du modèle à raisonner sur plusieurs étapes et à interpréter correctement les informations retournées par l'application cible.

La revue de Xu et al. [7] confirme que GPT-4 présente les capacités de synthèse les plus avancées pour corréler des informations disparates issues de scanners de vulnérabilités, et que les LLM sont utilisés pour traduire des descriptions de vulnérabilités en scripts d'exploitation fonctionnels appelés **PoC** (**Proof of Concept**, démonstration technique servant à prouver qu'une vulnérabilité est réellement exploitable).

2.4 Analyse de malwares

L'analyse de malwares représente un autre domaine dans lequel les LLM sont progressivement intégrés. Cette tâche implique généralement l'étude de programmes binaires malveillants afin de comprendre leur comportement, leurs mécanismes de persistance et leurs méthodes de communication avec des serveurs de commande et contrôle.

Xu et al. [7] décrivent l'utilisation des LLM pour la **rétro-ingénierie** (analyse d'un programme binaire pour en comprendre le fonctionnement sans accès au code source) et l'explication de code binaire, facilitant la compréhension des **vecteurs d'attaque** (chemins ou méthodes utilisés par un attaquant pour accéder à un système). Les modèles peuvent notamment être utilisés pour expliquer des portions de code désassemblé, identifier des appels système suspects ou résumer le comportement global d'un programme malveillant.

Cette capacité d'analyse de malwares est également évaluée par Conceição et Cruz [9] dans le cadre du **benchmark** CSLU (**Cybersecurity Language Understanding**, un ensemble structuré de tâches permettant de comparer les modèles). Les résultats montrent que les LLM peuvent atteindre des performances élevées dans certaines tâches analytiques bien définies.

Si quatre des LLM testés affichent des scores parfaits en analyse de malwares, les performances chutent dans d'autres sous-domaines, notamment l'exploitation de vulnérabilités et la conception pédagogique. Cette variabilité des résultats suggère que les modèles restent spécialisés dans certaines tâches spécifiques plutôt que capables de couvrir l'ensemble du domaine de la cybersécurité.

La génération automatisée d'artefacts comme les **honeypots** (ou pots de miel, systèmes pièges configurés pour paraître vulnérables afin d'attirer et étudier les attaquants) pose également des défis de sécurité majeurs, car ces outils sont à **double usage** (**dual-use**) et pourraient être détournés à des fins offensives malveillantes.

3 Techniques de spécialisation des LLM pour la cybersécurité

Les performances des LLM dans les tâches de cybersécurité dépendent fortement des méthodes utilisées pour les adapter au domaine. Plusieurs techniques ont été proposées afin d'améliorer leurs capacités de raisonnement, de réduire les erreurs et de mieux intégrer les connaissances techniques spécifiques à la sécurité informatique.

3.1 Prompting et Chain-of-Thought

Le **prompting** (art de formuler une instruction pour obtenir un résultat d'un LLM), incluant le **Chain-of-Thought** (inciter le modèle à décomposer son raisonnement étape par étape), s'avère efficace pour des tâches de haut niveau. Cette approche consiste à guider le modèle en lui demandant explicitement de détailler les différentes étapes de son raisonnement avant de produire une réponse finale.

Cependant, comme le soulignent Xu et al. [7], cette méthode se heurte souvent aux **safeguards** (filtres et règles éthiques intégrés pour empêcher les usages illégaux) et aux limites de la **fenêtre de contexte** (quantité maximale de données traitables simultanément par le modèle) des LLM généralistes.

Ferrag et al. [10] confirment que cette approche peut être combinée à d'autres techniques pour améliorer les résultats. *PentestAgent* [5] intègre notamment du Chain-of-Thought et de l'**auto-réflexion** (capacité du modèle à corriger ses propres erreurs en analysant ses sorties précédentes) pour le débogage, ce qui permet de limiter les erreurs en cascade dues aux **hallucinations** (problème où le LLM génère des informations factuellement fausses de manière convaincante). L'auto-réflexion aide en particulier à corriger certaines erreurs sans intervention humaine, bien qu'elle ne les élimine pas toutes.

3.2 RAG (Retrieval-Augmented Generation)

Le **RAG** est une technique qui connecte le LLM à des sources de données externes, telles que des documents ou des bases **CVE** (**Common Vulnerabilities and Exposures**, système de référentiel mondial attribuant un identifiant unique à chaque faille), pour fournir des réponses plus précises et limiter les erreurs.

Ferrag et al. [10] évaluent comment l'apport de connaissances externes en temps réel permet de réduire les hallucinations et d'augmenter la précision des réponses sur des vulnérabilités récentes. Les tests montrent que si les LLM de grande taille comme GPT-4 conservent une avance sur la compréhension globale, certains modèles ouverts comme Mixtral affichent des performances remarquables une fois optimisés par RAG.

AutoAttacker [2] intègre un *experience manager* inspiré du RAG pour mémoriser et réutiliser les sous-tâches réussies. Sur une attaque ransomware par exemple, il passe de 17 à 7 interactions tout en réduisant significativement les coûts d'API. *PentestAgent* [5] exploite également le RAG pour maintenir une base de connaissances à jour et compenser les limites des LLM liées aux connaissances datées et à la mémoire courte.

3.3 Fine-tuning

Le **fine-tuning** (ou affinage) consiste en l'entraînement spécialisé d'un modèle généraliste sur un corpus de données métier pour le rendre expert. Xu et al. [7] analysent comment l'entraînement des LLM sur des corpus spécifiques à la sécurité permet d'améliorer leur compréhension des protocoles réseau et des langages de bas niveau.

Cette approche vise à transformer un modèle généraliste en un expert cyber, bien que son efficacité soit actuellement limitée par la rareté ou le caractère privé des jeux de données de haute qualité.

Ferrag et al. [10] approfondissent cette analyse en comparant l'efficacité de techniques comme le **QLoRA** (optimisation de la mémoire pour un fine-tuning léger) et le **DPO** (*Direct Preference Optimization*, pour l'alignement sur des critères de sécurité stricts). L'utilisation de techniques de compression comme le **HQQ** est également explorée pour permettre le déploiement de LLM puissants sur des infrastructures aux ressources limitées.

3.4 Architectures multi-agents

PentestAgent [5] propose une **architecture multi-agents** composée de quatre agents spécialisés (Reconnaissance, Recherche, Planification, Exécution), coordonnés par un **orchestrateur** chargé de répartir les tâches.

Cette organisation permet une meilleure autonomie par rapport aux approches monolithiques, car chaque agent se concentre sur une étape spécifique du processus d'attaque ou d'analyse. Le système global peut ainsi reproduire plus fidèlement les méthodes utilisées par des experts humains lors d'un test d'intrusion.

Ferrag et al. [10] identifient ces architectures comme une direction stratégique, bien que les jeux de données actuels ne couvrent pas suffisamment les scénarios complexes, limitant l'automatisation totale dans les **SOC** (*Security Operations Centers*).

Nguyen et Husain [11] étudient spécifiquement la sécurité de ces systèmes **agentiques** en comparant cinq modèles intégrés dans deux frameworks : *AutoGen* et *CrewAI*. Ils identifient notamment le risque d'**hallucinated compliance** (situation où le modèle affirme avoir exécuté une action alors que le résultat est fictif). Le taux global de refus (pourcentage de requêtes bloquées par les barrières de sécurité) est de 41,5%, signifiant que 58% des tentatives aboutissent à un succès simulé.

Martínez et al. [12] comparent les modèles comme assistants dans un processus de **pentest** (simulation d'attaque réelle pour évaluer la sécurité) en environnement **GOAD** (*Game of Active Directory*, laboratoire simulant une infrastructure Windows vulnérable). Les LLM aident à interpréter les résultats d'outils et à suggérer des stratégies au sein d'une **forêt Active Directory** (structure logique la plus élevée regroupant domaines et utilisateurs). Claude Opus fournit les réponses les plus cohérentes, capable de maintenir le contexte sur plusieurs étapes, tandis que Copilot obtient des résultats plus limités.

4 Benchmarking et évaluation des LLM en hacking éthique

4.1 Cadres et métriques d'évaluation

L'évaluation des capacités des modèles de langage dans le domaine du hacking éthique constitue un enjeu méthodologique important. Contrairement aux benchmarks classiques en traitement du langage naturel, les tâches de cybersécurité impliquent souvent des environnements interactifs, des chaînes d'attaque multi-étapes et des objectifs opérationnels complexes. Il est donc nécessaire de concevoir des cadres expérimentaux capables de mesurer non seulement la réussite finale d'une attaque, mais aussi la capacité du modèle à raisonner sur des informations techniques, à planifier des actions et à s'adapter à l'évolution de l'environnement.

Isozaki et al. [13] s'appuient sur le framework *PentestGPT*, structuré en trois modules interdépendants : l'Analyse (*Parsing*) pour l'extraction des données issues des scans (ex : Nmap), le Raisonnement (*Reasoning*) pour la planification logique de l'attaque basée sur les vulnérabilités identifiées, et la Génération pour l'écriture du code d'exploitation ou des commandes système. L'outil est testé sur 13 machines cibles (Vulnhub) avec une limite de 5 tentatives par étape. Le taux de succès « bout-en-bout » (*end-to-end*) — désignant la capacité du modèle à mener l'attaque de la phase de reconnaissance initiale jusqu'à la compromission finale (obtention du flag/root) sans intervention humaine corrective — plafonne à 31%.

Ce type d'approche modulaire illustre une tendance importante dans la recherche actuelle : la séparation des capacités de compréhension, de raisonnement et d'action des LLM afin d'améliorer leur efficacité dans des tâches complexes. En particulier, l'introduction d'un module de raisonnement

explicite permet au modèle de structurer la progression de l'attaque plutôt que de générer directement des commandes isolées.

Plusieurs cadres d'évaluation ont été proposés pour mesurer les capacités des LLM en hacking éthique. Happe et Cito [14] font une revue de 19 articles (2023-2025) couvrant 18 prototypes d'outils offensifs basés sur des LLM. Les benchmarks comportent en moyenne 26,1 tâches, surtout issues de CTF comme HackTheBox ou TryHackMe. Le succès est majoritairement mesuré de façon binaire (réussi/échoué), même si un tiers des études commencent à suivre des sous-tâches. Les coûts en USD sont souvent rapportés pour comparer les prototypes. La plupart des tests portent sur des hôtes isolés, et seulement deux benchmarks récents s'attaquent à des réseaux entiers.

Cette observation met en évidence une limite fréquente dans les évaluations actuelles : les environnements expérimentaux restent souvent simplifiés par rapport aux infrastructures informatiques réelles. Les systèmes étudiés sont généralement des machines vulnérables isolées ou des challenges CTF, ce qui facilite la reproductibilité des expériences mais réduit le réalisme des scénarios d'attaque.

Conceição et Cruz [9] évaluent la maturité de sept LLM via le benchmark Cybersecurity Language Understanding (CSLU), divisé en deux axes : une évaluation théorique mesurant la compréhension conceptuelle des systèmes cybernétiques, et une évaluation fonctionnelle testant la génération d'artefacts opérationnels tels que des échantillons de malwares et des exercices de type CTF. Cette double approche permet de distinguer les capacités de compréhension conceptuelle des modèles de leurs capacités opérationnelles dans des scénarios de cybersécurité.

Gioacchini et al. [15] proposent AUTOPENBENCH avec deux métriques : le Success Rate (SR), qui mesure la proportion de tâches complétées avec succès, et le Progress Rate (PR), qui mesure la progression de l'agent vers l'objectif final. Cette seconde métrique vise à capturer les situations dans lesquelles un agent progresse dans la chaîne d'attaque sans atteindre immédiatement l'objectif final, ce qui constitue un indicateur plus fin de ses capacités de raisonnement.

Reinsperger [16] utilise quant à lui le nombre de flags obtenus, le coût en tokens, le nombre d'appels aux outils et un score normalisé inspiré des compétitions CTF, dans le cadre d'une évaluation expérimentale sur deux applications vulnérables développées spécifiquement pour l'étude : PHBlog (PHP) et Pycket (Django), contenant des vulnérabilités inspirées du référentiel OWASP Top 10 (IDOR, injection de commandes, SSRF, désérialisation non sécurisée, path traversal). Neuf modèles sont évalués : Claude Sonnet 4.5, GPT-5.1, GPT-5.1-Codex, GPT-4.1, Gemini 2.5 Flash, GLM-4.6, DeepSeek R1, DeepSeek V3.2 Exp et GPT-oss-120b, via le framework HackingBuddyGPT.

Liu et al. [17] introduisent pour les attaques par prompt injection la métrique keyword-evaluation ASR (KEY-E), qui mesure le ratio entre le nombre de réponses contenant un mot-clé malveillant prédéfini et le nombre total d'exemples testés. Pour les objectifs plus complexes, la métrique LLM-evaluation ASR (LM-E) utilise GPT-4 comme évaluateur afin de déterminer si la réponse générée contient des informations pertinentes permettant d'atteindre l'objectif de l'attaque. L'utilisation d'un LLM comme évaluateur reflète une tendance récente dans l'évaluation des systèmes basés sur l'IA, bien que cette approche puisse introduire des biais liés aux capacités ou aux limites du modèle évaluateur.

4.2 Résultats comparatifs entre modèles

L'ensemble des études convergent vers un constat commun, les modèles d'OpenAI GPT-4 dominent les benchmarks et experimentations, devant les modèles open-source. Cette domination s'explique en grande partie par leur taille, la qualité des données d'entraînement et l'optimisation des mécanismes de raisonnement.

Dans PentestGPT [1], GPT-4 résout 4 cibles faciles et 1 cible moyenne contre 2 faciles pour Bard et 1 pour GPT-3.5. Dans AutoAttacker [2], GPT-4 atteint 100% de réussite contre un échec complet de Llama 2. Dans hackingBuddyGPT [3], GPT-4-Turbo atteint 83% avec indices contre 50% pour GPT-3.5-Turbo et 33% pour Llama3-70b.

Dans HackSynth [4], GPT-4o domine avec 34,2% sur PicoCTF et 40% sur OverTheWire, tandis que Llama-3.1-70B s'en sort le mieux côté open-source (22,5% et 28,8%). Dans PentestAgent [5], GPT-4 atteint 74,2% de réussite globale contre 60,6% pour GPT-3.5. Dans l'évaluation de Reinsperger [16], Claude Sonnet 4.5 et GPT-5.1 atteignent entre 77% et 81%, dépassant largement l'outil ZAP pour les vulnérabilités nécessitant une analyse contextuelle.

Ces résultats montrent que les performances des LLM dépendent fortement de leur capacité de raisonnement et de leur intégration dans des architectures logicielles adaptées. Les modèles ne sont pas utilisés seuls mais souvent combinés avec des outils externes, des bases de connaissances ou des mécanismes de récupération d'information.

Ferrag et al. [10], dans leur évaluation massive de 42 LLM, confirment que si les LLM de grande taille comme GPT-4 conservent une avance sur la compréhension globale, certains modèles ouverts comme Mixtral affichent des performances remarquables une fois optimisés par RAG. Conceição et Cruz [9] révèlent quant à eux une irrégularité marquée selon les spécialités techniques : si quatre des LLM testés affichent des scores parfaits en analyse de malwares, les performances chutent dans d'autres sous-domaines comme l'exploitation de vulnérabilités sur des systèmes non-standard ou la conception pédagogique de CTF fonctionnels.

Pour Isozaki et al. [13] , bien que Llama 3.1-405B soit légèrement plus performant, les résultats prouvent que l'autonomie complète reste un verrou technologique. L'étude d'ablation montre par ailleurs que la performance chute drastiquement lorsque le module de Reasoning est simplifié ou supprimé, ce qui démontre que la réussite du pentest ne dépend pas de la capacité du modèle à « connaître » un exploit, mais de sa capacité à structurer une chaîne logique d'attaques.

4.3 Limites des benchmarks actuels

Malgré les progrès récents, plusieurs limites importantes persistent dans l'évaluation des LLM appliqués au hacking éthique. Les environnements expérimentaux restent souvent simplifiés et ne reproduisent pas fidèlement la complexité des systèmes informatiques réels.

Happe et Cito [14] identifient plusieurs limites structurelles aux benchmarks actuels. Les environnements synthétiques ne reflètent pas la complexité des vrais réseaux ni le côté non déterministe des exploits. Il y a aussi un risque sérieux de contamination des données d'entraînement pour les CVE publiques déjà connues des modèles ; les auteurs recommandent donc de randomiser les identifiants et d'utiliser des « canaries » pour détecter ce surapprentissage. Il faut également inclure des baselines (humaines ou automatisées) et évaluer à la fois des modèles de pointe et des modèles légers.

Xu et al. [7] soulèvent l'absence de benchmarks standardisés : il n'existe aucun cadre universel pour mesurer la sécurité du code généré par les LLM. Ferrag et al. [10] précisent que les jeux de données actuels ne couvrent pas suffisamment les scénarios d'attaques réseau complexes.

Reinsperger [16] souligne que son benchmark repose sur des applications spécialement conçues pour l'expérimentation, ce qui limite la généralisation des résultats à des environnements industriels plus complexes, et que certaines catégories de vulnérabilités OWASP ne sont pas incluses afin d'éviter que les modèles ne reproduisent simplement des exemples mémorisés durant leur entraînement.

Notre état de l'art se distingue du benchmark de Happe et Cito [14] en adoptant une perspective plus large : il met en évidence la diversité des approches techniques (RAG, fine-tuning, Chain of Thought) et des usages possibles au-delà du pentesting classique (spear phishing, analyse de malwares, sécurité matérielle). Il aborde également les risques propres aux LLM et s'intéresse à des architectures multi-agents plus complexes, que le benchmark de Happe et Cito considère principalement comme des boîtes noires.

5 Risques et enjeux éthiques

Les bénéfices potentiels des LLM pour la cybersécurité s'accompagnent d'un ensemble de limites techniques et de risques éthiques qui apparaissent de manière récurrente dans la littérature. Ces risques ne relèvent pas seulement de mauvaises performances ponctuelles, mais touchent à la fiabilité même des systèmes, à leur sécurité intrinsèque et à leur potentiel de détournement à grande échelle.

L'analyse des travaux récents montre que ces enjeux se situent à plusieurs niveaux. D'une part, les modèles peuvent produire des réponses erronées ou incohérentes, ce qui remet en cause leur usage autonome dans des tâches critiques. D'autre part, ils introduisent de nouvelles surfaces d'attaque propres aux systèmes fondés sur le langage naturel. Enfin, leur accessibilité croissante soulève la question de leur usage dual, à la frontière entre assistance légitime au pentest et automatisation d'activités offensives.

5.1 Hallucinations et fiabilité

Les **hallucinations** — un problème critique où le LLM génère des informations factuellement fausses, invente des outils ou se contredit de manière convaincante [7] — constituent l'une des limites les plus fréquemment citées dans l'ensemble des études analysées. Isozaki et al. [13] soulignent un paradoxe : les modèles souffrent d'hallucinations techniques (confusion de binaires) tout en ignorant des opportunités évidentes appelées **low-hanging fruits** (des vulnérabilités très simples à exploiter), comme des mots de passe en clair dans les logs [13]. Cela suggère que les LLM actuels manquent de « mémoire de travail » contextuelle sur le long terme durant un audit de sécurité [1].

Dans *PentestGPT* [1], les LLM natifs souffrent d'hallucinations, produisant des commandes ou des outils inexistants, et subissent une perte de mémoire à long terme [1]. *PentestGPT-GPT-4* corrige certains de ces problèmes mais reste limité par la **fenêtre de tokens** — qui définit la quantité maximale de données (mots ou fragments de code) que le modèle peut traiter simultanément — ce qui fait que le modèle perd progressivement le contexte [1]. *AutoAttacker* [2] est également sujet aux hallucinations : GPT-4 peut suggérer des modules *Metasploit* qui n'existent pas, même s'il s'auto-corrige après plusieurs tentatives [2].

Dans *hackingBuddyGPT* [3], les plus petits modèles ont tendance à halluciner en générant des commandes invalides et manquent clairement de « bon sens » — oubliant par exemple la réutilisation de mots de passe trouvés dans des fichiers de configuration [3]. Les vulnérabilités temporelles comme **cron** (une tâche planifiée s'exécutant automatiquement à intervalles réguliers), qui nécessitent d'attendre et d'observer des changements asynchrones, posent problème à tous les modèles, y compris GPT-4-Turbo [3]. Ce dernier reste sujet à la répétition de commandes en boucle — une limite commune à l'ensemble des modèles testés [3]. L'utilisation de grandes fenêtres de contexte (128k tokens) s'avère coûteuse sans amélioration significative des résultats au-delà de 20k tokens [3]. C'est précisément l'accumulation de ces faiblesses qui explique l'écart entre GPT-4-Turbo (83%) et l'expert humain (91%) [3].

HackSynth [4] hallucine parfois des adresses IP inexistantes, ce qui peut mener à des attaques hors périmètre si le pare-feu est mal configuré [4]. Il peut aussi déstabiliser son propre environnement en saturant la mémoire ou en cassant des variables système, et tombe parfois dans des « rabbit holes » en s'obstinant sur une approche inefficace, comme essayer de décoder en Base64 en boucle [4]. Dans *PentestAgent* [5], les hallucinations peuvent provoquer des erreurs en cascade, même si l'**auto-réflexion** — la capacité du modèle à analyser ses propres raisonnements pour identifier et corriger ses erreurs — aide à les limiter [5]. Xu et al. [7] synthétisent ce constat : malgré leur puissance, les LLM inventent fréquemment des bibliothèques ou des paramètres inexistants, nécessitant une validation humaine constante [7].

Au-delà des exemples techniques, ces résultats montrent que la fiabilité des LLM ne peut pas être réduite à un

simple taux de succès global. Un modèle peut réussir plusieurs sous-tâches tout en restant dangereux s'il introduit, à certaines étapes critiques, une commande erronée ou une hypothèse fausse présentée avec assurance. Dans un contexte de cybersécurité, cette apparente confiance du modèle constitue un problème majeur, car elle peut induire un analyste en erreur ou déclencher des actions offensives inadaptées.

Cette question de la fiabilité est d'autant plus importante que les tâches étudiées sont souvent séquentielles. Une hallucination précoce dans un audit ou dans une chaîne d'exploitation peut se propager à l'ensemble de la procédure. Ainsi, même lorsque les LLM améliorent la productivité globale, ils nécessitent encore une supervision humaine forte, en particulier dès lors que les actions proposées ont un impact direct sur un système cible.

5.2 Vulnérabilités propres aux LLM

Au-delà des hallucinations, les LLM eux-mêmes introduisent de nouvelles surfaces d'attaque. Liu et al. [17] étudient les **attaques par prompt injection**, qui consistent à manipuler le comportement d'un modèle de langage en injectant des instructions malveillantes dans les données analysées par le modèle, forçant celui-ci à ignorer ses instructions initiales [17]. Contrairement aux vulnérabilités classiques du web comme l'**injection SQL** (manipulation de requêtes de base de données) ou le **XSS** (*Cross-Site Scripting*, injection de scripts malveillants dans une page web), ces attaques exploitent directement le mécanisme d'interprétation du langage naturel des LLM [17].

Les attaques atteignent plus de 80% de réussite pour les objectifs statiques, et un taux moyen d'environ 50% pour des objectifs dynamiques [17]. Les auteurs évaluent cinq stratégies de défense, telles que le *paraphrasing* ou la *sandwich prevention*, qui réduisent l'efficacité des attaques sans les éliminer complètement [17]. Ces résultats mettent en évidence une vulnérabilité structurelle : le modèle ne dispose pas de mécanisme fiable pour distinguer une instruction légitime d'une donnée malveillante [17].

Ferrag et al. [10] identifient deux autres menaces : l'**empoisonnement de données** (*Data Poisoning*), qui consiste en l'introduction de données corrompues dans le corpus d'entraînement ou via le **RAG** (*Retrieval-Augmented Generation*, technique connectant le LLM à des sources externes pour améliorer la précision) pour manipuler les futures décisions du LLM, et les attaques par **Déni de Service (DoS)**, qui exploitent les capacités de calcul des LLM pour paralyser des systèmes de défense automatisés [10].

Nguyen et Husain [11] identifient un comportement particulier appelé **hallucinated compliance**, dans lequel le modèle affirme avoir exécuté une action ou produit un résultat, mais les données générées sont fictives et ne correspondent pas à une exécution réelle dans l'environnement [11]. Certaines attaques, notamment celles visant l'extraction d'informations système ou les **SSRF** (*Server-Side Request Forgery*, vulnérabilité permettant de forcer un serveur à envoyer des requêtes non prévues), réussissent dans environ 60% des cas [11]. Les auteurs concluent que les architectures **agentiques** (systèmes fondés sur des agents autonomes) nécessitent des mécanismes de sécurité supplémentaires : contrôle d'accès aux outils, isolation des environnements d'exécution dans des **sandboxes** (environnements isolés et contrôlés) et validation stricte des entrées [11].

Ces travaux montrent que les LLM ne doivent pas être considérés uniquement comme des composants intelligents, mais comme de nouveaux objets de sécurité à part entière. Leur vulnérabilité n'est pas seulement liée à des bugs d'implémentation ou à des erreurs logicielles classiques, mais à leur mode même de fonctionnement, fondé sur l'interprétation probabiliste du langage. Cela complique fortement la conception de mécanismes de défense robustes, car une attaque peut être formulée sous des formes lexicales multiples tout en poursuivant le même objectif malveillant.

L'intégration croissante des LLM dans des systèmes connectés à des outils externes, à des bases documentaires ou à des services web accroît encore ce risque. Dès lors qu'un modèle peut lire des contenus non fiables, appeler des outils ou déclencher des actions, une vulnérabilité de type prompt injection ou hallucinated compliance n'a plus seulement une portée théorique : elle peut conduire à des conséquences concrètes sur les systèmes d'information.

5.3 Dualité d'usage et risques éthiques

L'ensemble des études soulève la question fondamentale du **dual-use** (**double usage**) : une même technologie peut être utilisée à la fois à des fins légitimes, comme le **hacking éthique** (attaquer un système avec autorisation pour corriger ses failles), et à des fins malveillantes, comme l'automatisation d'attaques [7]. Xu et al. [7] formalisent ce risque sous le terme de **Weaponization** (**Militarisation**), désignant le détournement des LLM à des fins d'attaque où des acteurs malveillants pourraient automatiser la création de malwares polymorphes capables de changer de forme pour contourner les antivirus [7].

Cette évolution crée une course aux armements technologique : si les LLM offrent des gains de productivité pour les **pentesters** (professionnels réalisant des tests d'intrusion), ils abaissent également le seuil de compétence requis pour lancer des attaques sophistiquées [8]. Hazell [6] illustre concrètement ce risque avec le **spear phishing** (**hameçonnage ciblé**), une attaque de précision où le message est personnalisé pour une victime spécifique [6]. La possibilité de générer plus de 600 emails ciblés pour moins d'un centime d'euro l'unité, grâce à une simple **ingénierie de prompt** basique visant à contourner les **safeguards** (barrières de sécurité éthiques), confirme les limites des protections actuelles [6].

Fang et al. [8] soulignent que la création d'**agents autonomes** capables d'automatiser des attaques à partir d'instructions en langage naturel pourrait faciliter l'accès à des outils offensifs pour des utilisateurs peu expérimentés, insistant sur la nécessité de développer des « garde-fous » techniques (mécanismes de sécurité destinés à limiter les comportements dangereux d'un système, comme des filtres de contenu, un contrôle d'accès aux outils ou l'exécution

dans des environnements isolés) [8]. Reinsperger [16] et Nguyen et Husain [11] partagent cette préoccupation, rappelant que les capacités démontrées dans un contexte de recherche pourraient être exploitées de manière malveillante sans encadrement [16, 11]. Cette dualité met en évidence le caractère ambivalent des LLM en cybersécurité : ils constituent des outils offensifs puissants mais sont eux-mêmes soumis à de graves vulnérabilités structurelles [7, 10].

Sur le plan éthique, cette situation soulève plusieurs questions. La première concerne la responsabilité : lorsque qu'un système fondé sur un LLM propose ou exécute une action offensive, il devient difficile de répartir clairement la responsabilité entre le concepteur du modèle, l'intégrateur du système et l'utilisateur final. La seconde concerne l'accès à ces capacités : la démocratisation des interfaces conversationnelles permet à des utilisateurs peu techniques d'obtenir des contenus, des scripts ou des stratégies qui auraient auparavant exigé une expertise spécialisée.

Enfin, ces travaux mettent en évidence une tension entre innovation et sécurisation. D'un côté, les LLM offrent des opportunités réelles pour améliorer l'efficacité des audits, la formation en cybersécurité ou l'assistance aux analystes. De l'autre, ils augmentent simultanément la surface d'attaque, la vitesse d'automatisation des offensives et la difficulté de distinguer un usage pédagogique d'un usage malveillant. Cette ambivalence justifie la mise en place d'un encadrement technique, méthodologique et réglementaire plus strict pour les systèmes intégrant des modèles de langage dans des environnements sensibles.

6 Conclusion

Cette revue systématique de la littérature, portant sur les travaux récents consacrés à l'utilisation des LLM en cybersécurité offensive, a permis de dresser un état des lieux des capacités des LLM dans le domaine du hacking éthique. Les travaux analysés démontrent que si les modèles de pointe, particulièrement la famille GPT-4, affichent des performances remarquables pour l'automatisation de tâches complexes comme le pentest ou le spear phishing personnalisé [8, 6], l'autonomie complète sans intervention humaine demeure un défi technique majeur [13].

L'émergence de techniques d'optimisation telles que le RAG, le fine-tuning spécialisé et les architectures multi-agents a considérablement amélioré la pertinence technique des réponses et réduit les coûts opérationnels [10, 5]. Toutefois, ces avancées sont indissociables de risques structurels persistants. Les hallucinations techniques, la perte de contexte sur le long terme et la vulnérabilité intrinsèque aux injections de prompts limitent encore leur fiabilité dans des environnements de production critiques [13, 1].

Enfin, la question du *dual-use* (double usage) met en évidence une tension éthique fondamentale : les LLM constituent des multiplicateurs de force pour les défenseurs, mais ils abaissent également la barrière à l'entrée pour des attaquants moins expérimentés [7, 8]. La sécurisation des systèmes agentiques eux-mêmes, via un contrôle strict des accès aux outils et une validation rigoureuse des entrées, apparaît comme une étape indispensable pour une intégration sûre de l'intelligence artificielle générative dans l'arsenal de la cybersécurité. Des travaux futurs devront notamment s'intéresser au développement de mécanismes d'évaluation standardisés et à la conception d'architectures plus robustes face aux attaques visant directement les modèles de langage.

Remerciements

Nous adressons nos sincères remerciements à nos encadrants, M. Frédéric Flouvat et Mme Safa Yahi, pour leur accompagnement tout au long de ce travail.
Leurs conseils, leurs remarques pertinentes et leur disponibilité nous ont permis d'orienter nos recherches et d'améliorer la qualité de cette étude. Nous les remercions également pour leur encadrement scientifique et pour l'intérêt qu'ils ont porté à ce projet.

Références

[1] Yi Liu Gelei Deng, Vıctor Mayoral-Vilches, Peng Liu, Yuekang Li, Yuan Xu, Tianwei Zhang, Yang Liu, Martin Pinzger, and Stefan Rass. Pentestgpt : An llm-empowered automatic penetration testing tool.

[2] Jiacen Xu, Jack W Stokes, Geoff McDonald, Xuesong Bai, David Marshall, Siyue Wang, Adith Swaminathan, and Zhou Li. Autoattacker : A large language model guided system to implement automatic cyber-attacks. *arXiv preprint arXiv :2403.01038*, 2024.

[3] Andreas Happe, Aaron Kaplan, and Juergen Cito. Llms as hackers : Autonomous linux privilege escalation attacks. *Empirical Software Engineering*, 31(3) :70, 2026.

[4] Lajos Muzsai, David Imolai, and András Lukács. Hacksynth : Llm agent and evaluation framework for autonomous penetration testing. *arXiv preprint arXiv :2412.01778*, 2024.

[5] Xiangmin Shen, Lingzhi Wang, Zhenyuan Li, Yan Chen, Wencheng Zhao, Dawei Sun, Jiashui Wang, and Wei Ruan. Pentestagent : Incorporating llm agents to automated penetration testing (2024). *URL https ://arxiv. org/abs/2411.05185 v1*, 2024.

[6] Julian Hazell. Spear phishing with large language models. *arXiv preprint arXiv :2305.06972*, 2023.

[7] HanXiang Xu, ShenAo Wang, Ningke Li, Kailong Wang, Yanjie Zhao, Kai Chen, Ting Yu, Yang Liu, and HaoYu Wang. Large language models for cyber security : A systematic literature review. *ACM Transactions on Software Engineering and Methodology*, 2024.

[8] Richard Fang, Rohan Bindu, Akul Gupta, Qiusi Zhan, and Daniel Kang. Llm agents can autonomously hack websites. *arXiv preprint arXiv :2402.06664*, 2024.

[9] Tiago Conceição and Nuno Cruz. Evaluation of the maturity of llms in the cybersecurity domain : T. conceição, n. cruz. *International Journal of Information Security*, 24(5) :197, 2025.

[10] Mohamed Amine Ferrag, Fatima Alwahedi, Ammar Battah, Bilel Cherif, Abdechakour Mechri, Norbert Tihanyi, Tamas Bisztray, and Merouane Debbah. Generative ai in cybersecurity : A comprehensive review of llm applications and vulnerabilities. *Internet of Things and Cyber-Physical Systems*, 5 :1–46, 2025.

[11] Viet K Nguyen and Mohammad I Husain. Penetration testing of agentic ai : A comparative security analysis across models and frameworks. *arXiv preprint arXiv :2512.14860*, 2025.

[12] Antonio López Martínez, Alejandro Cano, and Antonio Ruiz-Martínez. Generative artificial intelligence-supported pentesting : a comparison between claude opus, gpt-4, and copilot. *arXiv preprint arXiv :2501.06963*, 2025.

[13] Isamu Isozaki, Manil Shrestha, Rick Console, and Edward Kim. Towards automated penetration testing : Introducing llm benchmark, analysis, and improvements. In *Adjunct Proceedings of the 33rd ACM Conference on User Modeling, Adaptation and Personalization*, pages 404–419, 2025.

[14] Andreas Happe and Jürgen Cito. Benchmarking practices in llm-driven offensive security : Testbeds, metrics, and experiment design. *arXiv preprint arXiv :2504.10112*, 2025.

[15] Luca Gioacchini, Marco Mellia, Idilio Drago, Alexander Delsanto, Giuseppe Siracusano, and Roberto Bifulco. Autopenbench : Benchmarking generative agents for penetration testing. *arXiv preprint arXiv :2410.03225*, 2024.

[16] Manuel Reinsperger. *Dangerous Capability Evaluation of Large Language Models for Web Penetration Testing*. PhD thesis, Technische Universität Wien, 2025.

[17] Yi Liu et al. Automatic and universal prompt injection attacks against large language models. *arXiv preprint arXiv :2403.04957*, 2024.